Author:
Neve, James O

Title:
Advancing the field of content-based and collaborative filtering reciprocal recommender systems

Advancing the field of content-based and collaborative filtering reciprocal recommender systems

By

JAMES NEVE

Department of Engineering Mathematics
UNIVERSITY OF BRISTOL

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of DOCTOR OF PHILOSOPHY in the Faculty of Engineering.

DECEMBER 2021

Word count: fifty two thousand four hundred and fifty six

Abstract

Recommender systems are personalisation tools that predict a user's preference for an item. They are used on services where users have to choose between a large number of options, such as shopping services like *Amazon* and movie websites such as *Netflix*. Even with search functions, the choice can often be overwhelming, and recommender systems provide users easy access to the items that are best suited to them, usually based on their history of previous preferences. Algorithmically, recommender systems often generate a score between 0 and 1 which represents how much a user will like an item. Items with high scores can then be recomended.

Reciprocal recommender systems are a more complex subtype of recommender system designed for services where the objective is to recommend people to each other, such as online dating, social and recruitment services. They are considered complex because the recommendation must be based on a bidirectional preference relation: it is important that both the person being recommended and the person viewing the recommendations are satisfied.

In spite of the relatively interesting algorithmic challenge their complexity presents, reciprocal recommender systems have been overlooked in the literature, with a rich variety of research concentrating on user-item recommendation, and very few techniques for reciprocal recommendation. The purpose of this PhD is to contribute new methods and ideas to reciprocal recommendation, to advance the field with modern techniques currently being used in user-item recommendation, and to develop novel algorithms unique to reciprocal recommendation.

This thesis is divided into three main sections: content-based filtering, collaborative filtering and hybrid filtering, to correspond to the main subdivisions of recommender systems. Each of these sections contributes new techniques to that field within the context of reciprocal recommendation. This includes both adaptations of existing algorithms and entirely novel methods.

All of these methods are tested against large datasets from industry, including data from a popular online dating service, and from a social recipe-sharing website. Their success over and above the current state of the art demonstrates the value of these new techniques, and provides a base of modern techniques that can be further improved upon by researchers in this field.

Dedication and acknowledgements

I would like to express my sincere gratitude to my supervisor Dr Ryan McConville, whose expertise in machine learning greatly contributed to my ability to develop new and interesting ways to recommend people to each other. I truly appreciate him making time for me every week, even during the busiest periods, for helping me with everything from maths to ethics, and for astute TV and pizza recommendations.

I would also like to express my gratitude to the numerous other academics who helped me along the way both at the University of Bristol and at conferences, with progress reviews and ideas for improvements to papers or algorithms. In particular, I am grateful to Professor Weiru Liu for helping guide the direction of my research, and for her thoughts on my thesis.

I acknowledge the academic contributions made by my former supervisor Dr Palomares to the sections related to papers where he is listed as an author.

I'd like to thank my family: my wife Konatsu, my parents Peta and John and my sister Alice for keeping me sane during a pandemic PhD. Without their support, this thesis would have been a much more difficult and much less enjoyable endeavour.

Finally, I'd like to thank my fellow PhD students and especially my friend Ercan Ezin for all sorts of interesting discussions and perspectives on both my algorithms and life in general.

Author's declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: ..... DATE: .....

TABLE OF CONTENTS

List of Tables

List of Figures

Figure **Page**

CHAPTER 1

INTRODUCTION

Recommender Systems (RSs) are personalisation tools that are used to help users of services find what they are looking for. Conventional RSs recommend items to users. *Reciprocal Recommender Systems* (RRSs) are a subset of recommender systems that recommend users to other users. They are used on social services, online dating and job recruitment platforms. RRSs have received comparatively little attention compared to item recommendation. This thesis describes a number of original contributions to the field of Reciprocal Recommendation that significantly advance the field, many of which are based on fundamentally different technologies from conventional recommendation.

=======

1.1 Recommender Systems

Recommender Systems were popularised by services such as *Amazon*[1] and *Netflix*[2]. Out of a desire to increase user engagement and facilitate their choice between potentially tens of millions of products on the site, RSs developed profiles of users based on their explicit preferences and implicit preferences derived from behaviour on the site. These preference profiles could then be used to recommend products to users. These systems effectiveness could easily be measured, assessed and improved: if users clicked on or purchased items they were recommended, the interaction was considered a success.

Recommender systems typically make recommendations by generating a preference relation between a user and an item. This is a score representing how much the system estimates a user will like an item, and generally lies between 0 and 1. Elementary RSs score a number of

[1] https://www.amazon.com
[2] https://www.netflix.com

candidate items, and either recommend the items with the highest scores, or re-rank search results based on these scores. More sophisticated systems might also take into account factors such as whether or not the user has seen the item before, and serendipity (whether the user is unlikely to come across the item in the course of their normal behaviour) [49].

1.1.1 Recommender Systems Classification

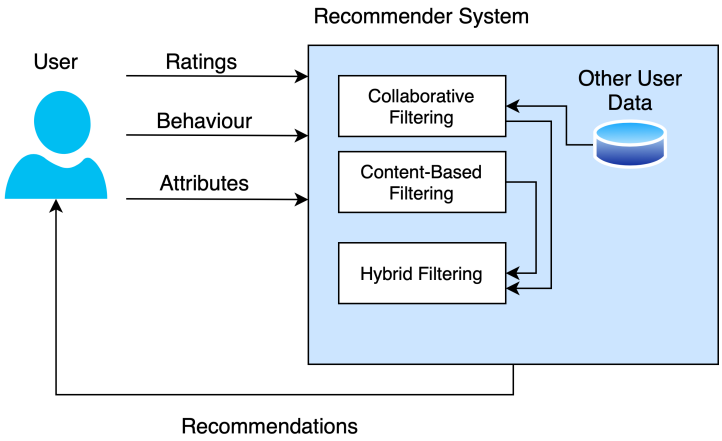


Figure 1.1: Recommender Systems Visualisation

The general process by which recommendations are made is visualised in Figure 1.1. A RS takes data from the user, which is generally (but not limited to) some combination of the user's ratings for previous items, which may be expressed as explicit or implicit preferences, their behaviour and their own attributes. The recommender system uses these, often in combination with data from other users, to calculate scores for candidate recommendations. These recommendations are usually ranked, and then displayed to the user. As the system accumulates more data about a user, the recommendations generally become more accurate.

As shown in the diagram, there are three main categories of Recommender System acknowledged in the literature [5]: *Content-Based Systems*, *Collaborative Filtering Systems* and *Hybrid Systems*. These three categories are also used to classify reciprocal systems, and are each discussed briefly in this section.

Content-based systems use explicit or implicit preference expressions by users to establish profiles of users describing their preferences for properties of items. These profiles are then used to make recommendations of items that fit these preference profiles. The profiles are often built from users' behaviour. For example, on a shopping service a user Alice purchases four items from the *Gardening* subcategory and two from the *DIY* subcategory. A content-based system might infer that Alice likes home improvement, and subsequently recommend her items related to this. Content-based systems have the advantage of being relatively simple to design and are often efficient to run.

Collaborative filtering systems use correlations between users to make recommendations, based on identifying similarities between the user viewing the recommendations and other users who have expressed similar preferences. For instance, on a streaming platform, two users Bob and Charlie have both watched *The Matrix*, *Mission Impossible* and *Die Hard*. After Alice watches *The Matrix* and *Mission Impossible*, a collaborative filtering system might recommend *Die Hard* to her based on her similarity to Bob and Charlie. There are a great many factors in user choice; in the case of movies, people might make choices based on genre, preferred actors, family situation and so on. Collaborative filtering has the advantage of being able to take account of these factors without having to make the potentially faulty assumptions underlying content-based systems, and as a result generally outperform them [39]. However, they do suffer from the *Cold Start Problem* [76], where a new user is not able to receive effective recommendations because similarity coefficients cannot be effectively calculated based on very little data.

Hybrid systems attempt to combine the advantages of content-based and collaborative filtering. *Burke* describes a number of ways of doing this [29]. For example, in *Weighted* systems the preference relation is based on a weighted average of the output of the content-based and collaborative filtering algorithm. *Switching* systems will use the result of a content-based or collaborative filtering algorithm depending on the context, often used to mitigate the Cold-Start Problem by using a collaborative filtering system after sufficient behaviour data has been generated. Hybrid systems tend to perform the best of the three categories based on competitions on large datasets such as the *Netflix Prize Challenge* [22]. However, their design and implementation is also more complex and the potential gains are sometimes minor over a straightforward collaborative filtering implementation.

Besides these three main categories, there are a number of other minor categories of RS for more specific situations. For example, *Knowledge-Based Recommender Systems* derive recommendations from general trends in user demographics, and *Context-Sensitive Recommender Systems* tailor their recommendations to fluctuations in user behaviour over contexts such as time and location. As reciprocal recommendation is still in a relative infancy compared to user-item recommendation, these situation-specific subcategories are not discussed in this thesis.

1.1.2 Recommender System Challenges

This section describes challenges and common problems with all recommender systems, whether user-item or reciprocal. These terms are used throughout this thesis and are therefore defined here.

Data Sparsity [15] is a common feature of RS datasets. A popular shopping service such as Amazon[3] might have millions of users and tens of thousands of products, and each user is likely to express an opinion about a small subset of them. Effective recommender systems algorithms must be able to calculate recommendations based on vast quantities of very sparse data. This

[3] http://www.amazon.com

can be challenging, as collaborative filtering algorithms often interpret user preference for items as a matrix [23], which causes practical problems if stored and operated on naively in memory.

The Cold-Start Problem [76, 82] is a specific problem within the context of data sparsity, where recommendations are required for a user who has recently joined the service and recommendations are needed with very little data. Generally, recommender systems depend on having a certain amount of information about a users and items to make recommendations: collaborative filtering solutions depend on expressions of preference to establish useful correlations between users, and content-based filtering algorithms often depend on inferred preferences to create profiles for users. New users and new items do not have this information, many of these methods therefore generate ineffective recommendations for these users. Because attracting and keeping new users is an important concern for many businesses, a considerable amount of work has been dedicated to solving the cold-start problem [129], with many algorithms that are able to achieve very high levels of accuracy on existing test data still performing poorly in the case of new users.

Filter Bubbles [100] occur when a user whose preferences have been established is then recommended only items that relate to those preferences and never has an opportunity to see other types of items. This problem is often self-reinforcing: if a user continues to click on their own recommendations without searching, their preferences for those items are reinforced within the system and they are even less likely to see different items in future. This is particularly a problem in news recommendation [85, 86]. In this context, a filter bubble created where a user is not exposed to information from different sources can significantly impact their views and opinions. A recommender system that does not create a filter bubble is said to have *Serendipity* [49].

Scalability [135] describes the problem where recommender systems take increasingly long to make recommendations as users and ratings are added to the system. Real datasets often have millions of users, so algorithms that can quickly make recommendations on toy examples lose their ability to do so on large datasets. Online services often require recommendations to be made in real time, which is not possible with some algorithms that require $O(n^2)$ time or even longer to make accurate recommendations.

1.1.3 Recommender System Evaluation

There are two methods commonly used to evaluate recommender systems, *Offline Evaluation* and *Online Evaluation*. This section discusses both methods, and a number of metrics that have been used to evaluate RSs.

Online evaluation involves implementing the proposed recommender system into a live service environment, where recommendations are displayed to users and their reactions to these recommendations is recorded. Exactly what data is recorded depends on the environment and what the service considers a successful interaction. A streaming service might consider a user watching a recommended movie to the end and giving it a high rating a success; an advertising

agency might consider a click to be a success. Successes and failures over a fixed period of time allow evaluation metrics described below to be calculated, which gives a measure of the system's performance.

Existing offline evaluation for recommender systems evaluates the system's ability to perform on a test dataset comprised of user preferences for items that the system has not seen before. Offline evaluation has a number of advantages. It is often much more convenient than online evaluation, as it does not require the recommender system to be implemented in a commercial environment. The same test data can also be used to evaluate multiple recommender systems, making it easy to compare results. However, it is not possible in the context of offline evaluation to present a user with recommendations and have them choose, which is the ultimate goal of RS design. It is therefore necessary to make some assumptions about what would be a useful predictor of good recommendations in the context of a static test data set.

Classification metrics tend to be more useful for most applications of recommender systems. These use a binary criteria such as whether or not a user watched a movie recommended to them to measure an individual prediction as success or failure. The most common evaluation methods use comparison metrics between the RS's predicted score for an item and the user's own rating. Early RS evaluation was done with a focus on accuracy: minimising the error between the system's predicted ratings and a user's actual ratings. The metric most commonly used in older evaluations is the *Mean Absolute Error* (MAE) [27, 130]. For N ratings of $r_i i \in 1..N$, and for p_i as the rating for r_i predicted by the recommender system, the MAE $\bar{E}$ is defined as:

$$M\bar{A}E = \frac{\sum_{i=1}^{N} p_i - r_i}{N} \tag{1.1}$$

Intuitively MAE would correlate well with real world recommender system performance. However, it is particularly unrepresentative of the most common task of recommender system: that of finding the best items to present to the user [55]. From the point of view of this type of recommender system, which might aim to select items with extremely high predicted ratings from tens of thousands of possibilities, differentiating 1/10 from 7/10 is less important than differentiating 9.4/10 and 9.5/10. However, the MAE does not take this into account, and a system with a low MAE from its ability to accurately predict poor ratings for items is not necessarily able to present satisfactory recommendations to users.

Some of these issues are solved through the use of *Mean Squared Error* (MSE).This is calculated as:

$$M\bar{S}E = \frac{\sum_{i=1}^{N} (p_i - r_i)^2}{N} \tag{1.2}$$

This ensures that all results are positive, and has the advantage of emphasising outliers in the results. However, squaring the results can make the error less intuitive, as it becomes

measured in squared units of the response variable. The *Root Mean Squared Error* is therefore more commonly used, which retains the advantage of the MSE but is measured in the same units as the MAE. This is the square root of the MSE:

$$R\bar{M}SE = \sqrt{\frac{\sum_{i=1}^{N}(p_i - r_i)^2}{N}} \tag{1.3}$$

More commonly used evaluation metrics in modern recommender systems are *Precision* and *Recall*. Precision in this context describes the proportion of recommendations that were successful, and recall describes the proportion of potentially successful recommendations retrieved [55]. If the set of recommendations made is R and the set of successful recommendations is Rs, the precision is defined as:

$$Precision = \frac{|Rs|}{|R|} \tag{1.4}$$

We define the set of items in the test set that would be considered successful recommendations by the system as Ps, then the recall is defined as:

$$Recall = \frac{|Rs|}{|Ps|} \tag{1.5}$$

There are a number of methods of combining these two metrics, most commonly the *F1 Score* [118], which is essentially the harmonic mean of precision and recall:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{1.6}$$

There also exists a more general form of the F1 Score known as the *F-Score* or F_β *Score*. This is used when the precision is known to be more or less important than the recall, and uses a modifier, β, which represents how many times more important recall is than precision. It is defined as:

$$F_\beta = \frac{(1+\beta^2) * Precision * Recall}{(1+\beta^2)\dot{P}recision + Recall} \tag{1.7}$$

However, due to the lack of current research into the correct precision/recall balance for evaluation of reciprocal recommender systems, all research in this PhD measures the standard F1 score, providing precision and recall separately.

The final metric that is commonly used in recommender system evaluation is related to the *Receiver Operating Characteristic Curve* (ROC Curve). This is a line graph that plots the true positive rate of the model against the false positive rate. An example ROC Curve (taken

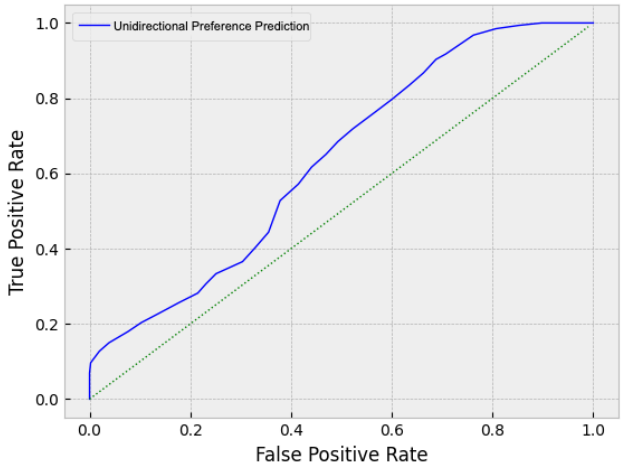


Figure 1.2: ROC Curve Example

from Chapter 5) is shown in Figure 1.2. In the case of recommender systems, an algorithm will usually output a value between 0 and 1 of predicted user preference for an item. To draw the ROC curve from test data, a threshold is varied between 0 and 1, and the item is considered a recommendation if the generated score is above the threshold. The rate of true and false positives from that threshold value then becomes a point on the graph. An algorithm's general effectiveness over all possible thresholds can be measured by the area under the ROC curve, known as the *Area Under the Curve* (AUC).

It is important to note, however, that even with the existence of the F1 Score and similar metrics, it is important to consider precision and recall independently, as the two metrics are not of equal value, and their importance depends on the recommender system and the application in question. An extremely large online shopping service might be particularly concerned with precision, retrieving all possible recommendations is less important than ensuring that the retrieved recommendations are accurate. A new streaming service with a relatively small number of movie choices might be concerned with recall to ensure that the user sees a variety of potentially successful recommendations and isn't shown the same small set of recommendations every day.

In addition to these metrics, a number of other evaluation methods have been designed to measure more specific facets of RSs [64]. For example, Ge et al.[49] describe metrics to measure *Coverage* (the number of items over which a RS has enough information to make a successful recommendation) and *Serendipity* (the extent to which a RS is capable of making successful recommendations that are not very similar to items the user has liked in the past). Diversity is sometimes used as a metric for the amount of variety present in recommendations, which can often lead to higher levels of satisfaction and engagement for users [133].

1.2 Reciprocal Recommender Systems

This section describes the basics of reciprocal recommendation, which is person-to-person rather than item-to-person recommendation, including the areas they are used in, common RRS design and evaluation, and the challenges presented by this field as compared to conventional recommender systems.

1.2.1 Reciprocal Recommender System Applications

The most common application of RRSs in the literature is online dating [72, 113]. Dating services rose to popularity in the '90s with *Match.com* and in the last decade, services with broad appeal such as *Tinder*[4] and other services such as *OurTime*[5] helped extend the appeal of the field to certain demographics. Online dating is an interesting application of reciprocal recommendation for several reasons:

- Users of dating services often provide very rich information about themselves, including categorical data such as age and job, descriptive text profiles and photographs. Different users might base their decisions on very different subsets of these criteria [45, 146].
- It is popular, especially recently, for online dating services to include binary methods of expressing positive and negative preferences for other users, which facilitates machine learning and evaluation.
- Dating services often have a huge number of registered users, sometimes in the millions, of which only a very small number might be successful matches for each other.

For these reasons, much of the research done on reciprocal recommender systems has used data from online dating services - more detail on this can be found in Chapter 2. However, very few of these datasets are public, which often impedes progress in this area.

Online dating in particular is an interesting research area because dating services have widely different objectives and presentations of information depending on their objectives. Some services such as *Twitter* are aimed at younger people, and focus very heavily on using photos as the primary decision-making process for users. Other services, such as *Match.com*, focus on slightly older markets, and present entire profiles. The correct recommender system for an online dating service may vary significantly depending on the presentation and primary market of the service.

Reciprocal recommendation is also commonly applied to job recruitment. An example of such a RRS can be seen in [132]. Recruitment is often slightly more complex in the sense that a company is generally not a single person, but conceptually two entities are aiming to make decisions about

[4] https://tinder.com/
[5] https://www.ourtime.co.uk/

each other, and it is beneficial to recommend jobs to candidates who might be able to successfully apply for them. Both candidates and companies often specify in great detail what they are looking for, making content-based filtering in this area an interesting challenge.

Social recommenders such as He et al. [53] and Tsuorougianni et al. [137] are a much larger field, thanks to the proliferation of social networks such as *Facebook*[6], and the fact that unlike dating, many social services make the user-user connections public. Much of the research into social matching is based on graphs and recommending friends of friends. While this is valuable and informative research, systems based on friends-of-friends tend not to be applicable to the one-to-one matching in dating and recruitment because in particular heterosexual dating is represented by a bipartite graph, and Alice's connection's connection is another female and therefore not a potential match. LBGTQ+ dating data was not available for experiments during this thesis, but graph-based algorithms from social networks might be applicable in this case.

Reciprocal recommenders have a number of other potential applications which have yet to be explored in depth. There are a few descriptions in the literature of systems related to education, such as matching teachers to students [160] or matching learning partners with each other [114]. The potential scope for RRSs is much wider than this, however, and they might usefully be applied in any situation where two people interact with each other, from matching customers to customer service representatives, to creating business connections and matching investors to company founders.

1.2.2 Reciprocal Recommender Design

In Section 1.1.1, methods of classifying Recommender Systems were discussed. These classifications logically extend to reciprocal recommendation: RRSs can also be described as *Content-Based*, *Collaborative Filtering* and *Hybrid*. Similarly to conventional recommendation, the objective of reciprocal recommendation is to establish a preference relation between 0 and 1. However, while a conventional recommender estimates a unidirectional preference relation of, for instance, how much Alice might like the movie *Die Hard*, a RRS estimates a bidirectional preference relation for how much Alice and Bob will like each other.

As shown in Figure 1.3, a basic RRS uses elements of conventional recommender systems to calculate this bidirectional preference relation. Depending on whether the relationship is symmetrical (as in a social service) or asymmetrical (as in recruitment or heterosexual dating), one or two models is used to establish two unidirectional preference relations that represent Alice's preference for Bob, and Bob's preference for Alice. These are then aggregated into a single unidirectional preference relation that can be used to make recommendations.

To illustrate this, a RRS *RECON* is described [113]. RECON is a RRS for online dating, and is one of the earliest reciprocal recommenders in the literature, and the paper to establish the term *Reciprocal Recommender*. It is a content-based reciprocal recommender, and makes

[6]https://www.facebook.com/

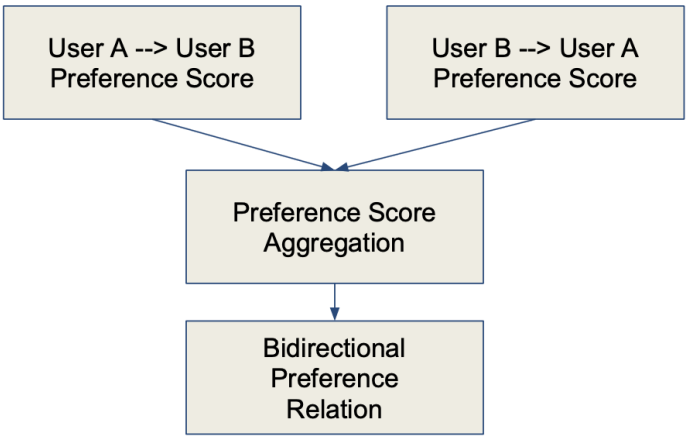


Figure 1.3: General conceptual model for Reciprocal Recommender Systems

recommendations based on categorical data (such as binned age, job and hobbies) as opposed to unstructured data such as freetext profiles and images. In order to generate unidirectional preference scores for Alice, RECON establishes a preference profile for her based on her historical expressions of preference. For example, if all of her previous messages had been to people in the age range 20 - 30, this category would be identified with a higher number in her preference profile.

In order to identify whether Bob would be a good match for Alice, RECON compares Bob's attributes to Alice's preference profile, and calculates a score that represents how closely the two match, which represents a unidirectional preference relation from Alice to Bob. RECON then performs the same operation in reverse, comparing Alice's attributes to Bob's preference profile to calculate a second unidirectional preference score from Bob to Alice. Finally, the system combines the two scores using the *harmonic mean* into a single reciprocal preference score that represents how much Alice and Bob might like each other.

While many reciprocal recommender systems follow the above pattern of generating two unidirectional preference relations that are subsequently aggregated, more recent works such as Neve et al. [92], which uses have shown that there may be advantages to predicting the bidirectional preference relation directly. This is described in more detail in Chapter 5. Some reciprocal recommender systems also base their recommendations on solutions to the *Stable Matching Problem* formulated by Gale and Shapley [47] where N men and N women must be optimally paired with each other. While solutions to this theoretical problem have limitations in terms of efficiency on a large number of real users unevenly divided into gender and sexual preference, some of the solutions have informed modern RRS designs.

1.2.3 Reciprocal Recommendation Evaluation

Conventional RSs are generally evaluated using standard machine learning methods: the dataset generally represents a sparse matrix of users and items, with users indicating either binary positive and negative preferences or scores for items. Where a model is part of the system, they are usually trained using a percentage of this dataset, and then datapoints that were excluded from the training set form the test set, from which the model's metrics such as accuracy, precision and recall can be inferred. Recommender system designers often consider precision an important metric for RSs [21], as a high precision helps to establish *trust* in the system. Users who trust their recommendations are much more likely to use them in future.

In the RRS domain, Pizzato et al. [112] suggest modified versions of the precision and recall metrics for evaluation. These take account of the fact that a recommendation in a RRS setting is only successful if both users like each other. Precision is therefore defined as the proportion of recommendations where the user being recommended and the user being recommended indicated positive preference. RL is defined as the set of users who were recommended each other and expressed mutual preference, and RN is the set of users who were recommended to each other but at least one of them expressed negative preference. Precision is then defined as:

$$Precision = \frac{|RL|}{|RL| + |RN|} \tag{1.8}$$

Recall in RRS settings is defined as the proportion of the total set of expressions of mutual preference retrieved by the model. A low recall indicates a small total number of recommendations, and therefore a high chance that a returning user will see the same recommendations repeatedly. Where P is the set of total reciprocal matches, recall is defined as:

$$Recall = \frac{|RL|}{|P|} \tag{1.9}$$

The F1 score can also be calculated exactly as described in Section 1.1.3.

Note that success of a model in the context of these metrics based on matches in a reciprocal setting implies a degree of coverage that it does not in non-reciprocal settings. Success based on matches would not be achieved by recommending the top users on the service, as these users match with a very small percentage of their recommendations, so even if it achieved one-way success, it would not generate a high precision.

1.2.4 Challenges of Reciprocal Recommendation

Reciprocal recommendation has challenges over and above the normal demands of recommendation in the sense of correctly establishing bidirectional preference. This section describes several additional challenges that are unique to reciprocal recommendation.

1.2.4.1 Data Structure

Conventional recommender systems are trained on sparse matrices of user preferences for items. In order to make effective recommendations, either numerical ratings or binary positive and negative preferences are required - we cannot make inferences about a user's preference or lack thereof for an item based on no interaction with it. On, for instance, shopping and streaming services, users are quite likely to provide both positive and negative feedback about the items.

It may, however, be unethical for social and recruitment services to provide the ability for users to publicly rate each other. Positive and negative preference must therefore be inferred implicitly from user actions. Positive preference is relatively easy to infer - users tend to interact with other users who they prefer. Negative preference is more difficult, as a simple lack of interaction might indicate a variety of factors besides negative preference, including the users not having seen each other. A likely more accurate method of inferring negative preference is to consider a lack of a response to an interaction. If Alice indicates a preference to Bob, and he sees this but chooses not to respond, Bob most likely has a negative preference towards Alice. While this method allows us to build models, it has some weaknesses. In particular, while star ratings on shopping services provide a clear scale and approximate symmetry, users might ignore preferences for a variety of reasons besides negative preference.

In addition to preference indicators, direct objectives of reciprocal recommendation are often difficult to evaluate against based on data held by the system. The ultimate objective of a recruitment service might be hiring, but the service might only hold data until the company and candidate exchange contact details. Similarly, an online dating service is unlikely to have complete data on which couples have stayed together. RRSs therefore have to use intermediate objectives such as binary indicators of preference to make their recommendations. While increasing the number of positive interactions is likely to increase the chance that one of these interactions will represent a success ultimately, it is more difficult for system designers to evaluate the impact of the system on the service as a whole than it is for conventional recommender system designers.

1.2.4.2 Fairness

In standard RS settings, extremely popular items tend to make the task of recommendation easier. On a streaming service that uses a collaborative filtering algorithm, a universally popular movie is likely to appear in Alice's recommendation lists because she is highly likely to be correlated with someone who watched and expressed a preference for it regardless of her history. However, if she also watches and enjoys it that isn't necessarily a problem.

Conversely, a similar pattern on a RRS is much more likely to result in a negative outcome. On a recruitment service, recommending a popular job opening to a large number of candidates is likely to inundate the company with applications, and candidates who fail to get the job are unlikely to be happy with the outcome. Similarly, the distribution of preference expressions on dating services tends to form a long tail, with a small number of users receiving an extremely high

volume of preference expressions. Including these users in recommendation lists, regardless of preference scores, is likely to result in a negative experience for both the user being recommended and the user viewing recommendations.

Maintaining fairness is an important challenge in RRS environments, and while a few authors have attempted to address this problem [70], it continues to be a problem for system designers.

1.2.4.3 Preference Aggregation

As discussed in Section 1.2.2, a RRS often consists of generating two unidirectional preference relations that are combined into a single bidirectional preference relation. This aggregation is unique to reciprocal recommendation and worthy of study in its own right. The formula used to combine these two scores is known as an *aggregation operator*. Even for combining only two numbers, there are a variety of options, the most simple of which is the *arithmetic mean*, defined for two numbers a and b as:

$$AM(a,b) = \frac{a+b}{2} \tag{1.10}$$

While a simple arithmetic mean might represent a balance between the two scores, there are a number of situations where this is intuitively not a representative aggregation. If Alice's preference score to Bob is 0.6 and Bob's preference score to Alice is also 0.6, an arithmetic mean results in a bidirectional preference relation of 0.6. The same result is given by the arithmetic mean if the unidirectional scores are 0.3 and 0.9 respectively, even though the much lower score from Alice to Bob makes a positive mutual preference intuitively less likely. Similarly, an extremely popular user is much less likely to respond to preference indicators irrespective of score. The most commonly used aggregation operator in the literature [111, 150] is the *harmonic mean*, defined for two numbers a and b as:

$$HM(a,b) = \frac{2 \cdot a \cdot b}{a+b} \tag{1.11}$$

The harmonic mean of two numbers punishes large differences in a and b therefore helping to resolve the main weakness of the arithmetic mean as described above. While $HM(0.6,0.6)$ still resolves to 0.6, $HM(0.3,0.9)$ resolves to 0.45, which intuitively might represent a better estimation of how likely a reciprocal recommendation consisting of these two scores is likely to succeed.

More complex aggregation operators exist [154]. Depending on how these unidirectional scores are combined the performance of the reciprocal recommender system can change substantively [94], making this a valuable area of research.

1.3 Motivation and Research Questions

Conventional RSs have received significant attention over the last decade. Partly because of commercial interests and the willingness of companies involved to finance research through competitions such as the Netflix Prize [22], there are a very large number of papers on the application of advanced machine learning techniques to user-item recommendation.

In contrast, there is relatively little in the literature regarding reciprocal recommendation, and various opportunities to advance the field, both by applying modern recommendation techniques to the reciprocal field, and through research into the aspects of RRSs that differentiate them from conventional recommenders. In this section, these gaps in the field are described in more detail based on the standard categories for recommender systems. This lack of research into RRSs forms a strong motivation for further investigation into this field.

This section briefly reviews the current state of each subcategory of RRS, and defines one or more research questions that this thesis will aim to answer. In addition, the following general research questions will be explored through every chapter:

1. Can the current state of the art for reciprocal recommender systems be improved upon?
2. What are the most effective methods for reciprocal recommendation, and how does this contrast with the most effective methods for conventional recommendation?

1.3.1 Content-Based Filtering

As discussed in Section 1.2.2, the earliest example of a RRS in the literature is RECON, which is a content-based system. RECON makes recommendations based on categorical data and binned continuous data, which is often provided by users on dating services. Other examples of content-based systems in the literature make similar inferences about user preferences from categorical data, and base their recommendations on this data.

While categorical data is an extremely useful resource in the sense that it is both straightforward to develop algorithms based around it, and often very efficient for these algorithms to run, services in RRS settings often have very rich unstructured data available. Users on dating services often provide photographs and detailed text profiles about themselves. Recruitment services often allow companies and candidates to describe in detail what they are looking for. Informal research suggests that users often focus heavily on this unstructured information when making preference decisions, and incorporating this into recommendation could significantly improve results. Modern machine learning methods, and especially *Convolutional Neural Networks*, have recently been very successful at extracting meaning, usually from unstructured text and images, and these techniques could be extracted to extract features from this data in RRS contexts and subsequently estimate user preference and make recommendations.

Content-based RRSs also have a significant advantage over conventional RSs, because they are less susceptible to outside influences. Users will often base decisions regarding what movie to

watch or what product to buy on information from outside the service, such as recommendations from friends or review sites. However, users on social and especially dating services make their decision based solely on the information available on the service itself.

Research questions for content-based filtering are as follows:

3. Can models based on unstructured data such as photos be used to improve on current content-based RRSs?
4. Can content-based RRSs be used to improve on the results of collaborative filtering RRSs in cold start situations?
5. Is historical data a useful predictor of reciprocal preference in RNNs?

1.3.2 Collaborative Filtering

The field of conventional recommendation has advanced significantly in the last decade, with relatively advanced methods making incremental improvements to recommendation results on public datasets. Modern systems often consider user-item preference as a matrix, and use dimensionality reduction methods to infer latent factors from these matrices. User preference for these latent factors are inferred from their preferences, and can be used to make recommendations.

In contrast, until 2019, the most advanced collaborative filtering algorithm used for reciprocal recommendation was based on the nearest neighbour algorithm, one of the simplest methods of collaborative filtering, and the first example algorithm introduced in Aggarwal's seminal text on recommender systems [5]. Research shows that kNN solutions to recommendation often do not exhibit the performance of more advanced methods in terms of the evaluation metrics introduced in Section 1.2.3[22]. While the basis many modern recommendation techniques are likely to be transferable to reciprocal environments, research is needed on the best ways to adapt these algorithms, and which ones perform particularly well or badly in these environments.

This thesis aims to improve on current collaborative filtering reciprocal recommender systems by adapting techniques that have been proved effective in user-item recommendation. In addition, collaborative filtering RRSs commonly determine the reciprocal score by combining results from two user-item recommender systems. This thesis aims to determine whether the use of alternative aggregation functions has an impact on the results from the RRS.

Research questions for collaborative filtering are as follows:

6. Can modern techniques such as latent factor models be effectively adapted to reciprocal recommender systems?
7. Can the efficiency of reciprocal recommender systems be improved over and above what's possible with current models?
8. Does the aggregation function applied have a significant impact on the effectiveness of the recommender system?

1.3.3 Hybrid Systems

Hybrid content-collaborative algorithms are often capable of outperforming algorithms based on only one of these two technologies. This is demonstrated through competitions such as the Netflix Prize challenge [22], which make it easy to compare the performance of two algorithms directly against each other.

However, there are very few examples of hybrid RRSs in the literature. As with the other areas, this could potentially form a set of algorithms with very high impact to the users of services in RRS environments, and further research into combining the rich content available with more advanced collaborative filtering techniques could potentially be used to develop algorithms with much stronger evaluation metrics than the ones currently available.

Where hybrid systems are concerned, this thesis explores the following research question:

9. Can hybrid systems be used to improve on the results of content-based and collaborative filtering in reciprocal recommender systems?

1.3.4 Features of Reciprocal Systems

As explained in Section 1.2.4, there are a number of unique features of reciprocal systems. Until relatively recently, RRS algorithms were usually adapted directly from conventional recommenders, with relatively little treatment given to these unique factors. Few systems attempted to account for fairness directly. When aggregation of unidirectional preference relations was required, the harmonic mean was generally used without justification.

While the RRS algorithms in the literature were successful without necessarily considering these unique aspects of the field, taking them into account might significantly improve results, and further research was needed to establish the extent to which they could effect results.

1.4 Original Contributions

This section outlines the original contributions made by the author to the field of reciprocal recommendation, divided broadly into content-based, collaborative and hybrid filtering and practical technological innovations.

1.4.1 Content-Based Filtering

Many dating services allow users to present their profiles using unstructured data. This data might include photographs and freetext profiles. Social services such as Instagram [7] increasingly use images or videos as their main form of communication. On services specific to certain hobbies, such as cooking, users might present their recipes or creations as photographs. Visual media is

[7] http://www.instagram.com

extremely important in determining interactions, and informal research demonstrates this to some extent [8].

Although categorical data can be a valuable resource, as demonstrated by algorithms such as RECON [113], unstructured data such as freetext and videos is also valuable, but before the work conducted in this thesis there were no algorithms based on interpreting unstructured data. Chapter 5 introduces a system that predicts personal preference based on images. This system outperforms previous content-based RRS algorithms, and this difference is particularly pronounced in cold-start situations.

The original contribution of this thesis to the field of content-based RRS algorithms is fourfold:

1. To the best of our knowledge, the first model to predict personal preference using image data is presented. This model uses a Siamese Network to differentiate between user preference of two photographs of each other. The fact that machine learning can predict preference to photos of each other is a significant step forward for both machine learning and potentially social psychology.

2. A novel algorithm is presented using this model to predict attraction on an online dating service. This model is demonstrated to outperform current content-based methods and to outperform collaborative filtering algorithms in cold-start situations. It is also the first RRS to use unstructured data to make recommendations.

3. A second model was developed to evaluate histories of user preferences for images and make predictions based on this history. This is the first model that interprets user history as a continuous time series for RRS.

4. This model is used as part of a RRS that predicts reciprocal preference directly instead of two bidirectional preferences. This model is extremely accurate, and predicts user preferences better than current collaborative and content-based filtering methods.

This represents a significant advancement of the field of content-based RRSs, and demonstrates that perhaps even more so than user-item recommender systems, advanced machine learning techniques can effectively be applied to the unstructured data dominant in reciprocal environments to create very accurate RSs.

1.4.2 Collaborative Filtering

Since the advent of recommender systems, collaborative filtering has been the gold standard for effective models in all settings. In user-item recommendation, this has resulted in increasingly complex models used for recommendation, based on various machine learning techniques such as Convolutional Neural Networks [163] and Restricted Boltzmann Machines [123]. Reciprocal

[8] https://www.gwern.net/docs/psychology/okcupid/weexperimentonhumanbeings.html

collaborative filtering techniques had, however, been stagnant for a number of years, with little progress past kNN based models such as RCF [150].

Chapter 3 presents a collaborative filtering reciprocal recommender based on latent factor models. Most modern user-item collaborative filtering RSs use latent factor models in some form to make recommendations; they have the advantage of both producing accurate recommendations, and being very time efficient.

An important part of collaborative filtering is preference aggregation. Collaborative filtering by design is suited to predicting unidirectional preferences, and in the case of RRSs, these unidirectional preferences must be combined into a single bidirectional preference relation. Since the development of RECON, the harmonic mean was used for this with no particular justification [113]. Testing reveals that this is not necessarily either the overall best or the best in certain situations.

The contribution of this thesis to collaborative filtering is threefold:

1. An original latent factor model is trained based on stochasic gradient descent. This model is demonstrated to effectively predict latent factors representative of user preferences for unidirectional preference estimation.

2. This latent factor model is used to develop a RRS for predicting mutual attraction. Offline evaluation demonstrates that this system has very similar evaluation metrics to the best in class RRS, but significantly improves on efficiency.

3. An evaluation of aggregation functions for preference aggregation in RRSs is performed, which demonstrates that the choice of aggregation function has a significant impact on the evaluation metrics of RRSs.

These contributions move the field of collaborative filtering in reciprocal environments closer to their user-item counterparts. The demonstration that latent factor models are effective in reciprocal environments opens the door for other researchers to experiment with more advanced methods of generating latent factors, while the results regarding preference aggregation represent the first exploration of this unique aspect of reciprocal systems.

1.4.3 Hybrid Systems

There are some ambiguities in the term *Hybrid Recommender System*. A few papers such as that by Qu et al. [115] imply a hybrid RRS in the sense of recommendations based on both unidirectional and bidirectional preferences. There are, however, no examples of hybrid RRSs in the commonly understood sense of using both content-based and collaborative filtering to generate recommendations.

Chapter 4 presents a hybrid system using techniques from both content-based and collaborative filtering to make recommendations for a social service based on recipe sharing. The

hybrid system uses a weighted system to balance the contributions of the content-based and collaborative filtering systems to the bidirectional preference relation. This system produced better results than individual techniques based on offline testing.

The contributions of this thesis to hybrid filtering are threefold:

1. A model was trained based on Word2Vec [120] that made recommendations based on freetext recipe text. This represents the first content-based model incorporating freetext used as part of a RRS.

2. An original hybrid system was developed to make recommendations on a social service for recipe sharing. This system proved more effective than the individual models for making reciprocal recommendations.

3. The vast majority of RRSs in the literature operate on two separate classes of users (for instance, male and female users as part of heterosexual dating recommendation). We demonstrate that similar technologies can be used on social services with only a single class of users.

These contributions represent a significant advancement to the field of hybrid reciprocal recommendation, and the advantages of this hybrid system over its component models indicates that hybrid filtering is also a powerful tool for reciprocal recommender system designers.

1.5 Thesis Overview

This section gives a brief overview of the ground covered by each individual chapter of this thesis following this introduction in **Chapter 1**.

Chapter 2 provides a thorough overview of the research literature related to this thesis. This is primarily research related to recommender systems and especially reciprocal recommendation. However, some peripheral topics used in the construction of certain algorithms described in this thesis are also covered. For example, certain machine learning technologies are particularly important to the design of image-based recommender systems, and these are covered extensively in the literature, so relevant papers are presented in Chapter 2.

Chapter 3 describes an algorithm, *Latent Factor Reciprocal Recommender*, (LFRR) which makes reciprocal recommendations based on user preference for latent factors extracted through training. This algorithm is evaluated against the previous best in class collaborative filtering RRS as a baseline. This chapter also discusses preference aggregation and evaluates the impact of different aggregation functions on both the baseline algorithm and on LFRR.

Chapter 4 presents an algorithm *Hybrid Reciprocal Recommender System* (HRRS) for making recommendations for recipes. The construction of a freetext-based model for predicting user preference is described in detail, and the weighted combination of the results from this model

with a latent factor model for recommendation is also described. This algorithm is evaluated against other baseline algorithms.

Chapter 5 describes contributions made to content-based reciprocal recommendation. In particular, two models are discussed in detail: a model trained to differentiate between liked and disliked images, and a modeltrained to predict user image preference based on preference history. These models are used as the key component of recommender systems that are evaluated against each other and against existing baseline content-based and collaborative filtering algorithms.

Chapter 6 provides a final brief summary of the work presented in this thesis including original contributions made, and defines the themes and conclusions that can be drawn from this work when viewed as a whole.

1.6 Published Work

Much of the material in this thesis is based on peer-reviewed and published works. In this section, a list of publications by the author is presented below, alongside the chapter in this thesis where it is described.

Arikui - A Dubious User Detection System for Online Dating in Japan [95]. James Neve, Ivan Palomares. 2018. IEEE SMC. Outside the scope of this thesis.

Latent factor models and aggregation operators for collaborative filtering in reciprocal recommender systems [96]. James Neve, Ivan Palomares. 2019. ACM Recsys. Chapter 3.

Group Decision Making with Collaborative-Filtering 'in the loop': interaction-based preference and trust elicitation [43]. Ercan Ezin, Ivan Palomares, James Neve. 2019. ACM Recsys. Outside the scope of this thesis.

Aggregation Strategies in User-to-User Reciprocal Recommender Systems [94]. James Neve, Ivan Palomares. 2019. IEEE SMC. Chapter 3.

Hybrid Reciprocal Recommender Systems: Integrating Item-to-User Principles in Reciprocal Recommendation [97]. James Neve, Ivan Palomares. 2020. WebConf. Chapter 4.

ImRec: Learning Reciprocal Preferences Using Images [92]. James Neve, Ryan McConville. 2020. ACM Recsys. Chapter 5.

Reciprocal Recommender Systems: Analysis of state-of-art literature, challenges and opportunities towards social recommendation [104]. Ivan Palomares, James Neve, Carlos Porcel, Luiz Pizzato, Ido Guy, Enrique Herrera-Viedma. 2021. Information Fusion. Chapter 2.

Photos Are All You Need for Reciprocal Recommendation in Online Dating [93]. James Neve, Ryan McConville. 2021. ArXiV. Chapter 5.

1.7 Summary

This chapter first described recommender systems and identified three classifications into which they are commonly divided: collaborative filtering, content-based filtering and hybrid filtering. These three sections form the structure for this thesis, which makes original contributions to all three areas.

Following this, reciprocal environments were described in more detail, including the general structure of reciprocal systems, how their evaluation differs from conventional user-item recommenders and some of the problems and challenges that make RRS design more difficult than standard recommender systems.

There are a number of important gaps in the literature with regard to reciprocal systems, as described in Section 1.3. RRSs and the models behind them have recieved very little attention compared to conventional systems. This thesis addresses the gaps in content-based, collaborative and hybrid systems through a number of original models briefly outlined in Section 1.4, as well as exploring various peripheral topics unique to reciprocal environments such as preference aggregation and malicious user identification.

Finally, this chapter introduced the structure of the rest of this thesis, and enumerated the peer reviewed and published work that forms the backbone of the chapters describing the original contributions made in detail.

CHAPTER 2

BACKGROUND

There has been extensive research done on Recommender Systems in general. For user-item recommender systems, there are large public datasets available provided by companies such as Netflix and Amazon. While the lack of public datasets available has made research into reciprocal environments significantly more difficult, there has nonetheless been some research conducted specifically into RRSs, usually using private datasets provided by companies such as online dating services. In this section, relevant user-item recommender systems are reviewed in addition to all research into reciprocal systems that has helped to advance the field. As described in Chapter 1, recommendation can be divided into content-based, collaborative and hybrid filtering. This section follows this division, reviewing the three types of recommender in this order.

=======

2.1 Machine Learning Background

This section provides a background on some of the general machine learning technologies used in RSs, which will be referred to throughout this and subsequent chapters. They are used for extracting features from users and items and for direct score prediction, and are referred to throughout this dissertation.

The techniques described in this section were chosen because they relate specifically to other parts of this thesis. This is either because they relate specifically to a technique described later in this thesis, such as *Random Forest Models*, which are described below and subsequently used in Chapter 5, or because they are part of an algorithm discussed in a paper that is covered in the literature review section of this chapter, such as *AdaBoost*'s use in [70], discussed in Section 2.4.1.2.

2.1.1 Supervised Learning Methods

Supervised Learning describes machine learning algorithms that learn to predict an output given a training set of input-output pairs. Recommender systems can be described in these terms. The output is simple: either a binary positive/negative preference for an item, or a score within a range. The input is less straightforward: any aspect of the user's behaviour until they indicated preference for the item in question could be considered as the input part of the input-output pair, such as previously purchased items and the user's own profile.

Most supervised learning methods are adaptable to *classification* tasks (assigning a distinct category to an item) and *regression* tasks (predicting a value within a range). While both of these are used in the context of recommender systems, regression is more common, as most recommender systems aim to rank items in order, which is easiest to do if every item has a distinct score. Regression-based methods are therefore the focus of this section, with the caveat that all of the methods described are trivially adaptable to classification tasks.

2.1.1.1 Regression Trees

Regression trees, and their sister technique for solving classification problems, *Decision Trees*, are a machine learning technique that make predictions by repeatedly splitting the data based on the input features. They are occasionally used directly in recommender systems [35, 68], and also form the basis for other techniques that are more commonly used. Regression trees consist of *branches* that split the data based on specific criteria, and *leaves* that represent predictions. For example, consider predicting a user's score for popular horror movie *The Shining* directed by Stanley Kubrick. A simple regression tree might look as follows:

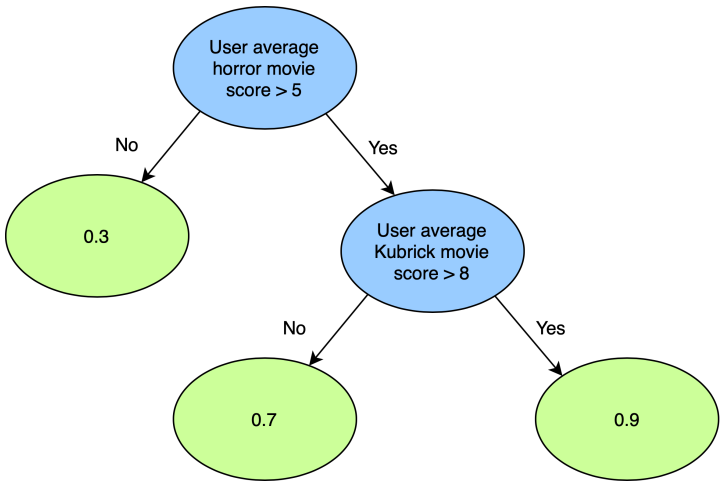


Figure 2.1: Example Regression Tree

In Figure 2.2, blue nodes represent decision points and green nodes represent outputs. The tree's branches divides the decision space into a set of J non-overlapping regions R_j. While the

above simple tree was manually constructed, for large data sets consisting of many items and parameters, the tree is learned from the data. In constructing a regression tree, an objective function known as an *Impurity Function* is minimised. An example of this is the *Residual Sum of Squares* function. This is the difference between the observed value in the training data y_i, and the mean value $\hat{y}$ of the values in its region as defined by the regression tree.

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y})^2 \tag{2.1}$$

Regression trees are generally trained by *Recursive Binary Splitting*: creating a new split that results in the greatest immediate reduction in RSS. Formally, this is a split of R_j into R_1 and R_2 where the following is minimised:

$$\sum_{i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \tag{2.2}$$

Regression trees trained naively for optimal performance on a test set have the weakness of *overfitting*: fitting too closely to the training set, reducing their efficacy on the test set and on real-world performance [122]. A detailed discussion of overcoming this is outside the scope of this thesis, but *pruning* techniques, where branches are removed to reduce the variance of the tree's performance at the cost of accuracy on the training set.

2.1.1.2 Random Forest

Random Forest [57] models aim to improve on the performance and reduce overfitting of decision trees. A simple random forest classifier for regression trains B regression trees. In order that the trees do not all end up at the same result, *bagging* is used: each tree is trained using a random subset of training data and features. For the output of a regression tree f_b, the prediction for a sample in the test set x' is typically the mean of the results of all decision trees:

$$\hat{f}(x) = \sum_{b=1}^{B} f_B(x') \tag{2.3}$$

Models such as random forest that are based on the outputs of a number of weaker models are known as *ensembles*. Random forest models are often used in recommender systems, either directly or as part of classifying or extracting features from content at an intermediate stage [9, 159].

Random forest models are widely used as part of recommender system implementations [9, 159]. They are often used as part of extracting features for collaborative filtering. In the context of this thesis, they are used in Chapter 5 for combining preferences for content-based features into a single score.

2.1.2 Boosting

Boosting is an alternative method to bagging to improve on the results of other models, most commonly used with regression and decision trees. While bagging creates an ensemble from random samples in the training set, boosting creates an initial model, and then trains each subsequent model on the incorrectly classified examples from the previous model. The ensemble consists of all of the previously trained models.

The most commonly used method of boosting in the literature at the time of writing is *AdaBoost* [46]. As with other boosting methods, AdaBoost can be used to attempt to improve on the results of any classifier, but is most commonly used with *Decision Stumps* - the weakest form of a decision tree, consisting of a single branch and two leaves, which split the data based on the criteria of the branch.

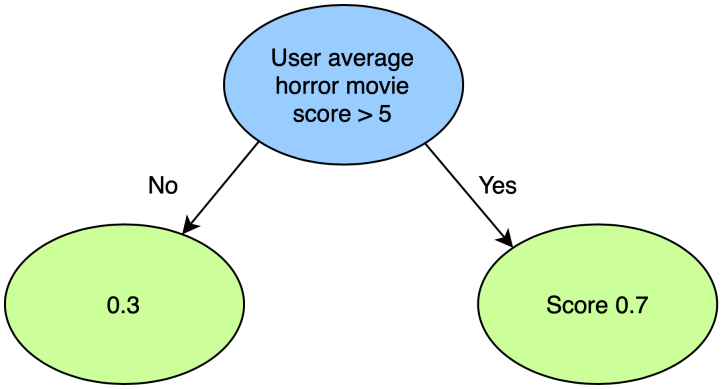


Figure 2.2: Example Stump

N stumps are initially selected, one for each variable in the data, and each initially with weight $\frac{1}{N}$. For each stump, the *Total Error* (TE) is calculated, which is the fraction of incorrectly classified examples. The performance of each stump n can then be calculated as α_n:

$$\alpha_n = \frac{1}{2} \ln \frac{1 - TE}{TE} \tag{2.4}$$

The weights of each stump w_n are then updated by:

$$w_n = w_n * e^{\pm \alpha_n} \tag{2.5}$$

Where α_n is treated as positive in cases where the predicted output and the actual output are different, and negative in cases where the predicted and actual output agree.

AdaBoost has been successful both as a core part of recommender systems [128] and in tackling peripheral problems such as fairness [70] and attack detection [155].

2.1.3 Neural Network-Based Models

2.1.3.1 Neural Networks

Neural Networks are a form of machine learning originally based on simulating the way that *neurons* (human brain cells) work. A neuron receives signals across synapses from surrounding cells. If these signals exceed a certain threshold, the neuron triggers and sends signals to other nearby cells. The logical representation of a neuron is a *perceptron*, shown in Figure 2.3.

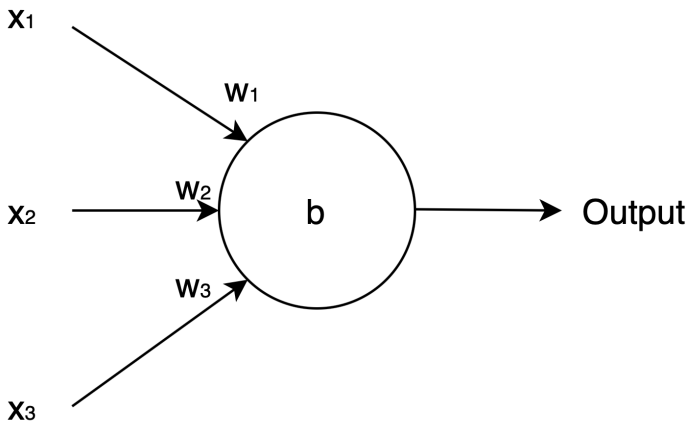


Figure 2.3: Perceptron

A perceptron takes inputs $x_1, x_2, ...$ multiplied by weights $w_1, w_2, ...$ These are generally described as vectors x and w such that their dot product $w \cdot x$ constitutes the sum of the products of the weights and inputs. The equivalent value to the neuron's threshold is the perceptron's *bias*, b. The output of the perceptron is 1 or 0 depending on whether the sum of the products of weights and inputs exceeds the bias:

$$\text{output} = \begin{cases} 0, & \text{if}(w \cdot x) + b \leq 0 \\ 1, & \text{if}(w \cdot x) + b > 0 \end{cases} \tag{2.6}$$

Single perceptrons are only capable of solving linearly separable problems. To solve more nuanced problems with non-linear solutions, they are networked together as in Figure 2.4 such that the output of one perceptron forms one of the inputs of another. The layers in between the input and output layer are known as *Hidden Layers*.

Equation 2.6 is known as the *activation function*. In this case, the activation function is a step function. In networks, *Sigmoid Neurons* are commonly used in place of perceptrons, which have the same essential function, but replace the activation step function with the *Sigmoid Function*, which is a smooth function constrained between 0 and 1:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.7}$$

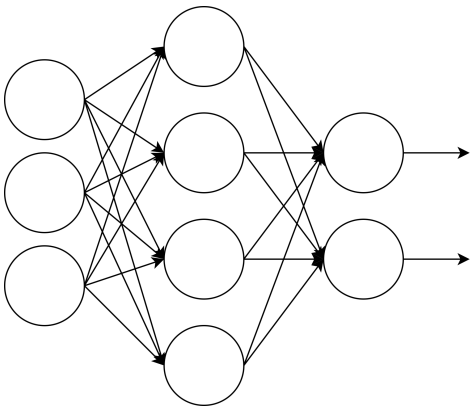


Figure 2.4: Network of Perceptrons

Neural networks are generally trained using *Stochastic Gradient Descent* (SGD). This process minimises an objective function for the network by taking small steps down the gradient of the function. A commonly used objective function for simple networks is the *Mean Squared Error* (MSE). For a training set of n pairs $(x, y(x))$ where x is the input and $y(x)$ is the desired output, and where a is the value predicted by the network, the objective function is:

$$C(w,b) = \frac{1}{2n} \sum_{x} ||y(x) - a||^2 \tag{2.8}$$

SGD repeatedly computes the error over a *mini-batch* of training examples, and then takes small steps down the error gradient based on the results of this mini-batch by modifying the weights of the final layer of neurons before the output. A technique called *backpropagation* is then used to compute the changes to previous layers in terms of the changes made to the final layer in order to graudally minimise the error. As this thesis focuses mainly on applications of neural networks and not on network design as such, a detailed discussion of the mathematics involved in this is outside the scope.

2.1.3.2 Convolutional Neural Networks

Fully-connected neural networks are often more difficult to work with compared to other methods of machine learning such as random forests due to the very large number of hyperparameters. *Convolutional Neural Networks* (CNNs) are a specific type of neural network that are particularly adept at classifying complex data such as images [79]. They have been extremely successful at tasks that other machine learning methods have struggled with, such as classifying objects on arbitrary backgrounds [74].

Conceptually, CNNs have been successful at classifying various types of data, including music and videos. However, for simplicity of language they are described here in the context of interpreting image data. CNNs slide a small $n \times n$ window across the image, known as a *local*

receptive field. At each position covered by the local receptive field, the pixels inside the field are taken as the inputs to a neuron in the network, as shown in Figure 2.5. The fact that CNNs divide images into regions in this way allows them to interpret local features of the image, as opposed to fully connected networks which treat each pixel value as an input independent of the surrounding pixels. The map from the input to the hidden layer is known as a *feature map*.

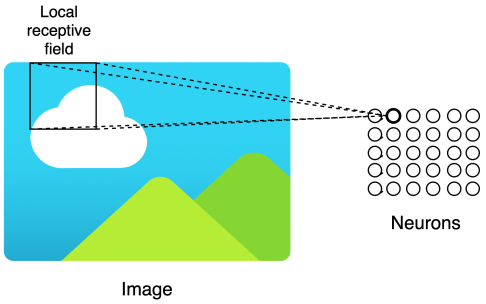


Figure 2.5: Feature Map of Local Receptive Field to Neuron

CNNs often consist of multiple feature maps across the same local receptive fields. This information is generally simplified by *Pooling Layers* which condense the information in convolution layers by mapping the outputs from a small region of neurons in a convolution layer to a single neuron in one or more pooling layers. Pooling is often a relatively simple function: for example, *Max Pooling* outputs the largest of the inputs, as a way of keeping the most significant information.

The output of a CNN is generally one or more layers of neurons that are fully connected to every output in the preceding pooling layer. Through training, this expresses the features interpreted by the hidden convolution and pooling layers as the objective of a regression or classification task. The structure of a CNN as a whole is depicted in Figure 2.6.

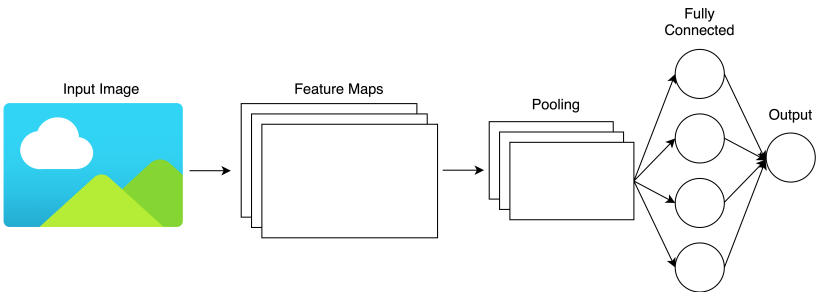


Figure 2.6: Structure of a CNN

CNNs have been successfully applied to a variety of areas, including image recognition to levels of accuracy that surpass human performance [54], as well as extracting information from various other mediums such as music [36] and video [98]. The following sections explore the application of these techniques to recommender systems.

2.1.4 Text Feature Extraction

Text comprises important content in many RS environments. In some, such as news RSs, text is the main form of content. In other cases, such as online streaming services, text descriptions of items are often used by users as part of their decision-making process. In this section, methods for extracting meaning from text commonly used in RSs are described.

2.1.4.1 Term Frequency - Inverse Document Frequency

Term Frequency - Inverse Document Frequency (TF-IDF) [51, 78]. TF-IDF aims to identify the most significant words in a document based on their frequency across the whole dataset. It is calculated as the product of two numbers: the term frequency and the inverse document frequency. The term frequency $t(t,d)$ of a term t in a document d, where $f_{t,d}$ is defined as the number of times the term t occurs in d is given by:

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{2.9}$$

The inverse document frequency is a measure of how rare a term is across all documents, and therefore how significant it is if it appears repeatedly in any individual document. If d is part of a set D of N documents, the inverse document frequency is defined as:

$$idf(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{2.10}$$

The TF-IDF is defined as their product:

$$tf - idf(t,d,D) = tf(t,d) \cdot idf(t,D) \tag{2.11}$$

Because TF-IDF is a measure of the importance of a word in a document that forms part of a set, it can be used to extract the most significant words for the purposes of matching. For example, movies about sports are quite common, but movies about very specific sports such as badminton are rare, so if a user watched several movies where the word "badminton" was mentioned several times in the description, this could be a useful predictor of their preferences and could be used to make recommendations in future.

2.1.4.2 Word2Vec

Subsequent approaches used more advanced methods for feature extraction such as Word2Vec [89] or deep learning-based methods [19], which represents words as vectors, and similarities in vectors as compared with the dot product corresponds to a similarity in meaning between words [67]. Word2Vec represents words as *embeddings*: vectors that represent the words in a

space, where similar vectors imply semantic similarity between words. For example, a Word2Vec implementation might have the word "dog" represented by $[1,4,3]$, "bulldog" represented by $[1,4,4]$ and "computer" as $[8,2,7]$. Vector similarity in this context is generally calculated as the *cosine similarity*, which for two vectors A and B is defined as:

$$sim(A,B) = \frac{A \cdot B}{|A||B|} \tag{2.12}$$

Word2Vec-based systems are trained on large corpuses of text. They are based on *next word prediction* models, which aim to predict a missing words in a corpus of text. These models are relatively easy to train because of the existence of large corpuses of text data such as Wikipedia[1], and the fact that no explicit labeling is needed. These models are generally trained using *skip-grams*: moving windows over words before and after the word being predicted. Word2Vec systems use these models to estimate similarity between words within a certain context. For example, given a skip-gram for the sentence, "John sat on a ____", and the next word prediction model estimates a high probability for the omitted word being "chair" or "sofa", we can infer that those two words have similar meaning in this context.

This can be more effective than TF-IDF alone because in recommender system contexts, text is written with the user rather than the recommender system in mind, and often different item descriptions are written by different people with different writing styles. Word2Vec-based systems are able to make recommendations even when the specific words used are different.

2.1.5 Learning from Images

Many recommender system environments contain image data, such product photos on shopping sites, preview images on movie streaming services and user photos in online dating environments. These images are part of users' decision making process, and therefore being able to extract features from them can improve recommendations.

As discussed in Section 2.1.3.2, CNNs are an extremely effective method for training models to extract information from images. This section outlines in more detail two extensions to the basic CNN model that are used later in this thesis.

2.1.5.1 Siamese Networks

Siamese Networks [71] are a particular structure of CNN that compare two items against each other, and evaluate them based on the difference between their features. Their most common use in the literature is in face recognition, where they are used to compare two images of faces against each other and decide whether or not they are photos of the same person. However, they have been adapted to a variety of domains, such as object tracking [25]. Siamese Networks also

[1] http://wikipedia.com

perform significantly better than standard CNNs at *one-shot learning*: learning a class based on a single or a very small number of examples [143]. This is a benefit in RS environments: recommenders should be able to predict preferences for a user based on a small number of expressions of preference.

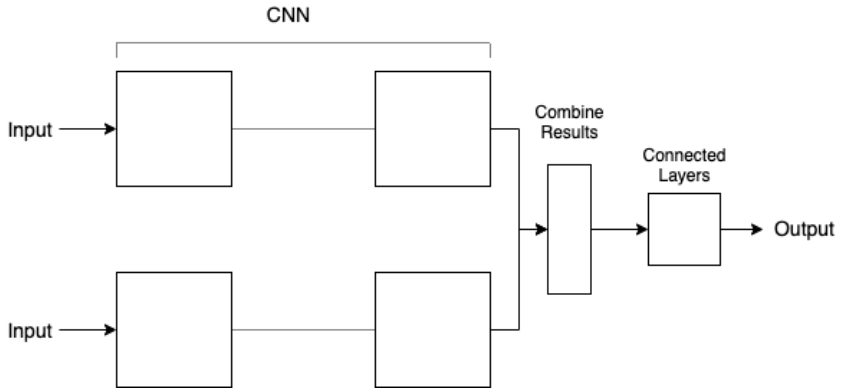


Figure 2.7: Structure of a Siamese Network

Siamese networks are structured as in Figure 2.7. For simplicity of language, the following discussion assumes that the inputs are images, although Siamese Networks can be used for any type of input. The two inputs y_1 and y_2 to be compared are used as the inputs of two symmetrical neural networks. These are commonly CNNs, although other types of networks are sometimes used. The outputs of the final fully-connected layer of the CNNs form *embeddings* h_{y_1} and h_{y_2} of the images: vectors that represent the meanings of the images in the context of the network. These two embeddings are then combined into a single vector that represents the difference between the two images.

$$D_W(y_1, y_2) = |h_{y1} - h_{y2}| \tag{2.13}$$

Siamese networks are often trained with *Contrastive Loss*. Traditional loss functions for training CNNs such as Binary Crossentropy result in a small loss when the embeddings for two images are similar to each other. This is often not desirable: in the case of face detection, classifying two similar faces as the same face is just as wrong as classifying two very different images as the same face. The Contrastive Loss function, uses a *margin* m, and depending on the size of the margin, results in a high loss when the the network's prediction is wrong about two similar images. The loss function is defined as:

$$L(y_1, y_2) = (1 - Y)\frac{1}{2}(D_W(y_1, y_2))^2 + Y\frac{1}{2}(max(0, m - D_W(y_1, y_2))^2 \tag{2.14}$$

where Y is the binary indicator representing Like and Nope, $D_W(y_1, y_2)$ *is the embedded distance between two images and m is the margin.*

Siamese Networks have been used successfully for reciprocal recommendation [92], and are described in this context with more detail in Chapter 5.

2.1.5.2 Recurrent Neural Networks

Data from RS environments often consists of sequences - in particular, when considering user behaviour. There are a number of intuitive reasons why this might be the case. When using streaming services, users might prefer certain kinds of music or movies at different times of the day or week. Purchases on an online shopping service might relate to a specific objective, such as building a new computer, and continuing to recommend them items related to this long after they have finished is unlikely to be successful.

The neural networks described in Section 2.1.3 are known as *Feedforward Networks*: the model produces an output for a single example that is independent of other examples. Recurrent Neural Networks (RNNs) are an architecture of neural network that incorporates memory such that the output from the last of a series of examples is different from that of a single example. They have been used successfully in recommender systems to incorporate time series data into recommendations [139, 147].

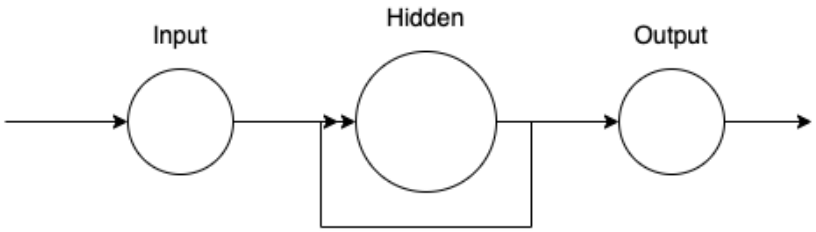


Figure 2.8: Structure of a Recurrent Neural Network

RNNs contain loops, which feed the output of a network back into current neurons. This means that they implement the concept of *memory*: they store computed results, and these results have an impact on subsequent predictions. This is shown in Figure 2.8. Each step therefore incorporates information from the previous steps into the prediction.

'Standard' RNNs are particularly good at processing short sequences, but their memory is *short-term memory*: when training them using longer sequences, the early items in the sequence have very little impact on the final prediction. This is known as the *vanishing gradient problem* [58]. (This also exists in deep neural networks, where early layers learn very slowly when trained with backpropagation).

Various structures have been proposed to overcome this limitation. One that has been particularly successful in allowing RNNs to hold and use information for longer is the *Long Short-Term Memory Network* LSTM [59]. A LSTM uses a *forget gate* comprised of a Sigmoid function that determines whether information is kept or not: a value close to 0 results in the information being forgotten by the network, whereas closer to 1 results in the information being stored. This

allows for much longer sequences to be processed, which is particularly useful in the field of recommendation, where long sequences of user behaviour are common.

2.1.6 Suitability of Methods

This section describes a large variety of machine learning methods, and the most effective method for a particular recommender system is not always immediately apparent without trial and error. However, there are some rules of thumb that might usefully be followed when choosing which to apply to a given situation.

As a general rule, problems are best explored using simple methods initially. If a problem can usefully be solved using machine learning, simple models will often demonstrate some decree of predictive power. Recently, Random Forest models are often used as part of initial modelling, particularly for tabular data, as they are not very resource-intensive, and have a small number of hyperparameters that significantly affect their predictive power, so tuning them is generally not time-intensive. A random forest model that is able to make effective predictions can often lead to the use of other methods such as boosting to gradually improve on evaluation metrics.

Two- or three-dimensional data such as photos and videos is often approached with neural networks, and specifically CNNs. Certain structures have been demonstrated experimentally to be especically good at solving certain problems, such as *Siamese Networks* for solving facial recognition problems [71].

Problems related to text data are recently approached through the use of transformer-based methods such as *BERT* [41]. These have not only been successful in general tasks such as predicting the next word in a sentence, but also produce embeddings for sentences and documents that can be conveniently visualised, clustered or used as the input to other machine learning methods.

2.2 Reciprocal Recommendation Background

This section briefly presents background for reciprocal recommendation unrelated to individual specific technologies. This includes literature reviews of RRS topics, in addition to psychological research that provides a base for some recommender systems and RRS concepts.

2.2.1 Reciprocal Recommendation Literature Reviews

Several papers have reviewed existing literature in the RRS field, and explored the extent of current technologies and opportunities for future advancement. The first paper to do this was written by Pizzato et al. [112]. The paper was published in 2013, and reviewed the state of the art at the time. The next publication to provide a review of RRS literature was a book chapter of the *Recommender Systems Handbook* titled *People-to-People Reciprocal Recommenders*, by Koprinska and Yacef [3]. Although this was intended as an introduction and is therefore briefer, but covers

some more recent literature. Most recently Palomares and Neve conducted a more thorough and up-to-date review of RRS literature, including definitions of various elements of the fields and comparison of results from the different algorithms [104].

In addition to these three general reviews, Zheng also condicted a review of RRSs specifically connected to recruitment [132]. This mainly focused on content-based recommender systems that used analysis of candidate profiles to recommend suitable jobs.

2.3 Content-Based Filtering

This section presents an overview of the literature of content-based filtering. Firstly, it describes collaborative filtering advancements in conventional recommender systems. Next, all available research on content-based filtering in reciprocal environments is presented. Finally, a detailed case study on a representative content-based filtering algorithm used as a baseline in subsequent chapters, *RECON*, is presented.

2.3.1 Content-Based Recommendation

Content-based filtering is based on the concept that users like items that are similar to each other. If 80% of Alice's purchases on an Internet shopping service are DVDs, a DVD is intuitively more likely to succeed than any other type of recommendation. This concept has been used since the early days of Recommender systems research in the 1990s [17] [107]. Meteren et al. provide an overview of this research [141].

The process of making recommendations is illustrated in Figure 2.9. The recommendation process is generally as follows [62]:

1. Relevant attributes of items are identified

2. User profiles of preferences for items are created through explicit or implicit inference

3. Items are recommended based on their similarity to user preferences

Depending on the setting, different parts of this process may be more or less challenging. In a setting such as news recommendation [78], the content is unstructured text, so feature extraction is a challenging problem. In areas such as online shopping, where items usually have predefined numerical and categorical attributes, research focuses on refining the recommendation process. These areas of research are all highly relevant to reciprocal recommendation, where both categorical and unstructured data is often available. In the following sections, approaches to different aspects of content-based recommendation are discussed, followed by a discussion of the inherent problems with content-based filtering.

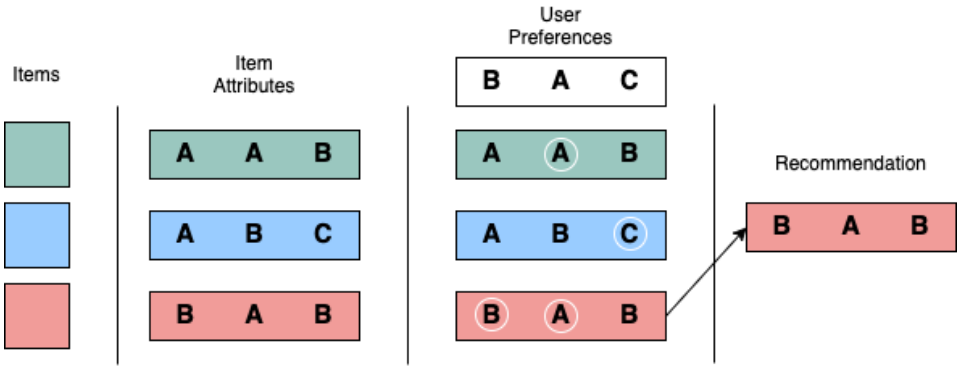


Figure 2.9: Content-Based Filtering

2.3.2 Content-Based Features

Item feature extraction is a non-trivial problem when the salient information about items is either freetext or images. Even in the case of simple numerical and categorical data, there are some nuances as far as feature extraction is concerned. For instance, Xia et al. [150] point out that when using continuous numerical data such as age and height for online dating recommendation, creating categories by binning the data might result in problems for users at the edges of bins. However, in general feature extraction for categorical data is relatively straightforward, and unstructured data such as text and images presents more challenges.

2.3.2.1 Text-Based Features

Since the inception of recommender systems, there has been a particularly strong interest in news recommendation [65]. This is partly because of the public availability of very large news datasets such as MIND dataset [148], and partly because of a number of unique challenges associated with this field. In particular, some news items lose relevance very quickly [102] and conventional collaborative filtering techniques, which often favour items with longer histories, are less likely to be useful. Users also often change their preferences rapidly [31], which makes it risky to rely on either explicit or implicit profiles [8].

Of relevance to reciprocal recommendation is the difficulty of feature extraction from news articles, which are often of a variety of different styles and levels of formality. In particular, the process of feature extraction has potential uses in reciprocal recommendation, where unstructured text profiles for users are common.

More recently, recommender systems built on transformer-based feature extraction models such as ELMO [110] and BERT [41] have become increasingly widely used [52, 152]. These are *transformers* [142], which are deep neural networks consisting of an encoder and a decoder. The encoder represents a sequence of words as a vector in high-dimensional space, and the decoder maps that representation to another meaningful sequence of words (for example, in the case of translation, this could be a sequence of words with the same meaning in another language). The

vector produced by the encoder can be used as a meaningful representation of the text for the purposes of recommendation.

2.3.3 Image-Based Features

Examples of content-based recommender systems basing their results on images is much less common. Lei et al. used user preferences to train a model based on ImageNet [40] that predicted user preference for one of two images [80]. This trains a network to map both users and images into the same space by generating embeddings for both, with images that the user preferred being close to the user in the space, and images the user did not like being further away. User preference for subsequent images can then be predicted by relative distance from the user.

A similar example is DeepStyle [84], which uses a Siamese Network (described in Section 2.1.5.1 to predict user preference for clothes based on images). DeepStyle uses pairs of positive and negative samples with user preference as the output to differentiate between the two images. This can then be used to make predictions about whether a user might like a new image by comparing it to an existing liked image. Arapakis et al. also designed an affective computing-based recommender system, where user facial expressions are used to improve recommendations.

Outside the direct field of recommendation, work has been done on feature extraction that is relevant to the work on recommendation using images presented in Chapter 5. In machine learning, feature extraction on images is often done using Convolutional Neural Networks (CNNs), which have been shown to be highly accurate at image classification tasks [40, 90]. Of particular relevance is the high performance of CNNs in classifying images of people [7]. There are fewer studies of feature extraction using human images; a few models claim to be able to identify features such as age and even height from images of faces [61, 81], but the evaluations of these models do not compare them to simple baseline methods such as predicting the average every time, so it is difficult to ascertain how accurate they are.

Of particular relevance to reciprocal recommendation in the context of online dating, which is of key interest in this thesis, is the prediction of attractiveness. Photos are an important part of attraction to profiles in online dating [140, 146], and therefore being able to predict attractiveness of a user from a photo is potentially a useful basis for the design of a recommender system for this field. There is research on predicting the absolute attractiveness of images on social services [88]. There are a number of papers that attempt to solve the problem of predicting absolute attractiveness within the bounds of the demographic represented by a dataset - to predict how attractive Alice is to the average person. Xu et al. [153] trained a neural network to predict facial beauty trained directly on image data, which accurately predicts scores. Fan et al. [44] use a deep neural network to estimate facial landmarks and predict attractiveness based on the ratios between these landmarks.

A more relevant concept is *personal preference* - whether or not Bob's photo is attractive to Alice specifically rather than whether the photo is attractive in general - is an important part of

a content-based recommender in this setting. There have been a small number of papers that attempt to predict this. Rothe et al. [121] designed a CNN based on using extracted facial features in combination with preference data from an online dating service to predict ratings for users, with 82% accuracy. Jekel et al. [63] used a similar method, predicting unidirectional preference based on features extracted by FaceNet and achieving an AUC of 0.83, which has comparable accuracy, using various models including a neural network, SVM and linear regression. Although these methods do not consider reciprocity or attempt to incorporate the models as part of a recommender system, the results are evidence for the capacity of machine learning to predict physical attractiveness from photos.

2.3.3.1 Recommendation Methods

Once relevant features for content-based recommendation have been identified, recommendation can be done using a number of different methods. Nearest neighbour (kNN) methods are especially common, where features are considered as a vector space, with items proximate to a user's previously preferred items being recommended [26]. kNN methods are also the most commonly used methods in reciprocal environments.

kNN methods represent items as vectors in a space. These vectors may be as simple as a list of numerical attributes (age, height and so on) or they may be more complex representations in high dimensional space extracted from unstructured data such as text or images. kNN methods use a *similarity function* to measure how close the two vectors are. The most common similarity function is the *Cosine Similarity*. For two vectors $\bar{X}$ and $\bar{Y}$, the Cosine Similarity is defined as:

$$\text{Cosine}(\bar{X},\bar{Y}) = \frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2}\sqrt{\sum_{i=1}^{d} y_i^2}} \tag{2.15}$$

Recommendations are made in kNN settings by determining a *Preference Profile* for the user, which consists of the average of the attributes previously preferred by the user. In the case of a ratings scale, this profile might weight the attributes towards items with higher ratings. The Cosine Similarity is then used to determine the closest items to the user's preference profile.

2.3.4 Content-Based Recommender Systems

In addition to nearest neighbour, a number of other methods have been used to make recommendations from features in content-based settings:

- *Rule Induction Methods* use rule induction to generate rules from a hierarchy of features such as categories and subcategories on a shopping service. A decision tree is constructed from these rules, and used to make recommendations for individual users [68].

- *Probabilistic Methods* use systems such as Bayesian Classifiers to predict the probability that a user will like a particular item given the items that they have previously liked [91].
- *Heuristic Methods* use custom designed algorithms to weight the importance of certain features for recommendation, and use these to determine which items will be recommended [48].

As the examples above show, the complexity in content-based algorithms tends to reside in the feature extraction process, especially where unstructured data is part of the recommendation process.

2.3.5 Limitations of Content-Based Methods

As discussed in previous sections, content-based algorithms have a number of strengths. To summarise, they are straightforward to develop, they can make recommendations based on relatively few preference expressions, and the data they use is the same data that the users have used to make their decisions. However, content-based algorithms also have a number of shortcomings:

1. Content-based filtering suffers from *overspecialisation* [2]. This occurs when a user becomes trapped in a *filter bubble* [100] of only expressing preferences for items that are similar to each other, which in turn reinforces the likelihood that those items will make up future recommendations. For example, if Alice's two favourite restaurants are Italian restaurants, she is very likely to be recommended more Italian restaurants even if there are others that she might enjoy. A few attempts have been made to design systems that compensate for this tendency and introduce more *serendipity* [49]. For instance, genetic algorithms have been proposed, where the mutations generate serendipitous recommendations that could then be reinforced by the user [131]. However, the majority of purely content-based algorithms still suffer from this problem.
2. Features used for content-based filtering are selected by 'feature engineering'. Representative features can be difficult to extract from unstructured data, and while the recommender system is often trained to weight features based on effectiveness, the original feature selection is done by the algorithm designers, depends on the domain knowledge of the designers and is sometimes arbitrary [62]. This can limit the effectiveness of even a very well designed algorithm. In addition, users can make decisions about purchases based on data that does not exist on the service - for instance, Alice might decide to watch the series *Friends* based on a review in the newspaper, or a friend's recommendation. This is important content that cannot be incorporated into the algorithm.
3. Even in offline testing, where issues such as the filter bubble do not arise, content-based filtering methods consistently underperform compared to collaborative and hybrid methods

[39]. Results for content-based filtering are generally worse on all metrics. The only area where they outperform collaborative filtering is in cold start situations e.g. Neve et al. [92], where collaborative filtering generally needs more information to make effective recommendations.

As a result of these shortcomings, it is rare for modern services to use content-based filtering as their main form of generating recommendations. Indeed, none of the top performing algorithms in the Netflix Prize challenge were purely content-based algorithms [22]. However, as a part of hybrid systems and in domains where there is rich content, they can still play an important part in the recommendation process.

2.3.6 Content-Based Reciprocal Recommendation

Reciprocal environments are an interesting application for content-based algorithms because they often negate some of the usual disadvantages of content-based systems. Certain filter bubbles are desirable in reciprocal recommendation - for example, relationships with users who live in or near the same city are more likely to be successful [134]. In general, in reciprocal environments, all of the content that users are using to make their decisions is available to the algorithm. Empirically, content-based methods have been relatively successful in reciprocal environments, with some promising results. In this section, previous content-based RRSs are presented with a discussion of their methods and results.

The earliest RRS in the literature is RECON [113], which is a content-based algorithm that uses categorical data to make recommendations. The results were promising at the time, and it provided a prototype formula for subsequent reciprocal recommenders. RECON is discussed in detail as a case study in the subsequent Section 2.3.7, so the discussion is omitted here.

Alanzi & Bain [13] used *Hidden Markov Models* (HMMs) to develop a content-based RRS for online dating. HMMs are statistical models used to interpret situations where the data consists of a series of observations $x(t)$. The causes of these events $y(t)$ must be inferred from the sequence $x(1...t)$ The system uses user profile data to determine a user's preferences: a successful interaction implies a user's preference for another user's profile data. The data consisted of interactions from a real online dating service, where users initiated interaction with each other by sending messages. A message followed by a positive reply was considered a successful interaction, whereas no reply or a negative reply was a negative interaction. The recommendation problem is conceptualised as a bipartite graph, and the HMM is used to predict the next link given a time series of previous links. The system outperformed other algorithms previously used on this dataset by 9% [12].

Tu et al. developed a reciprocal recommender using Latent Dirichlet Allocation (LDA) to cluster similar users [138]. Clustering is useful for recommendation because it identifies similarity between users within the same clusters, which can then be used to recommend users to users who have showed historical preference for certain groups. They first identify important variables

of users' profiles and eliminate redundant ones by calculating conditional entropy between these variables. LDA is used to identify latent factors from these variables, and classify users into groups that can subsequently be used to make recommendations. The recommender system presented positive results based on synthetic data, and the authors hope to present further results based on a real implementation.

O. Otakore et al. propose a system for reciprocal recommendation based on the results of a questionnaire [101]. Users fill in a questionnaire upon registration giving their own preferences and also what preferences they would accept from a potential partner. This is interesting as the only example in the literature of explicit reciprocal preference inference. Some users are likely to have strong preferences when they join an online dating service, and it would be useful to be able to make initial recommendations based on these preferences. The authors do not give the information required to evaluate in detail the effectiveness of this technique, but it nonetheless constitutes an original and interesting approach.

In content-based filtering outside of online dating, Ding et al. describe a content-based recommender system for graduate recruitment [42]. The system compares profiles of graduates to historical data from other graduates who have already found employment, and uses this to recommend jobs. The specifics of the similarity metric used are not discussed, but the system appeared to outperform other similar systems. Yu, Liu and Zhang describe a content-based RRS for recruitment based on inferring implicit and explicit preferences of companies and matching them with candidates' attributes and incorporating them into a vector space model [158]. Their solution achieved a success rate of above 50% based on offline experiments.

A similar content-based system to RECON was developed for matching learners in MOOCs (Massively Open Online Courses) [114]. These online study platforms have the level of connectivity of social networks, and they can encompass a large and diverse range of learners with different background and demographics. Another characterising feature of MOOCs are their possibilities for learning to work together in groups. Unlike previous recommender systems in MOOCs where the course itself is the item being recommended, in this algorithm the users are recommended to each other as peers to potentially study with, hence motivating the need for reciprocity. The algorithm makes recommendations based on similarities between two users' profiles. First, each attribute i is mapped to an integer distance metric; for example, the distance metric for age is the difference between binned ages. The distance score is the mean of the distances between individual attributes $d_i(x, y)$ out of N attributes:

$$\text{distance score}(x, y) = \frac{\sum_{i=1}^{N} d(x, y)}{N} \tag{2.16}$$

While this section outlines the most interesting contributions to the field of content-based reciprocal recommendation, a number of other papers have also been written on this topic. All contributions to the field in chronological order are presented in Table 2.1.

Reference	Year	Field	Recommendation Method
Pizzato et al. [111, 113]	2010	Dating	Nearest Neighbour recommendation using inferred preferences
He et al. [53]	2010	Social Networks	A Social Network-Based Recommender System
Yu et al. [158]	2011	Recruitment	Nearest neighbour graduate recruitment
Alanzi & Bain [13]	2013	Dating	Time series recommendation using Hidden Markov Models
Tsourougianni & Ampazis [137]	2013	Social Network	Recommendation on Twitter using feature extraction
Hong et al [60]	2013	Recruitment	Recommendation based on clustering profile attributes
Tu et al. [138]	2014	Dating	Recommendation based on clusters from Latent Dirichlet Allocation
Almalis et al. [14]	2014	Recruitment	Recommendation from Minkowski distance between profile and posting
Ding et al. [42]	2016	Recruitment	Nearest neighbour-based graduate recruitment
Prabhakar et al. [114]	2017	Social	Nearest neighbour recommendation
Otakore et al. [101]	2018	Dating	Questionnaire-based recommendation
Zheng et al. [162]	2018	Dating	Multi-stakeholder approach to maximise utility
Garcia et al. [119]	2019	Dating	Matching based on semantic similarity between preferences and profiles
Neve & McConville [92]	2020	Dating	Siamese Network for image-based recommendation
Neve & McConville [93]	2021	Dating	Photos Are All You Need for Reciprocal Recommendation in Online Dating

Table 2.1: Content-Based Reciprocal Recommender Systems

2.3.7 Case Study: RECON

RECON was the first RRS [113], and is a content-based RRS for online dating. The basis for RECON was a technical report by Pizzato et al., who proposed a method for learning implicit user preferences, considering user messages as expressions of preference [111]. In subsequent chapters, RECON is used as a baseline for comparison and it is therefore presented in detail

here.

RECON infers user preferences for specific attributes based on their previous expressions of preference towards users with those attributes. Assume a user profile U_x of a user x represented by:

$$U_x = \{v_{x,a} \forall\ a \in A\} \tag{2.17}$$

where $v_{x,a}$ represents the value of an attribute $a \in A$ associated with U_x.

Defining m as the number of times a user expresses preference for a user y with attribute value $v_{y,a}$ on attribute x, the authors define the preference of a user x for that attribute value as a distribution:

$$p_{x,a} = \{(v, m) : \forall \text{unique discrete values v of a}\} \tag{2.18}$$

RECON then calculates preference scores (how much a user x likes a user y) by comparing the inferred preferences to profiles of potentially recommended users. In order to make recommendations, the harmonic mean aggregation operator is applied between pairs of unidirectional preference scores to generate predicted reciprocal preference scores, which indicate how much two users might like each other.

RECON performed favourably in offline testing compared to normal user search, and also compared to standard non-reciprocal recommenders. It also has the advantage of largely avoiding the cold start problem [76, 82, 129], as it is able to start making recommendations for a user after having provided their first expression of preference. However, it does have several weaknesses in addition to those inherent in content-based systems:

1. It accounts for continuous user attributes such as age by dividing them into buckets. This reduces the likelihood that users whose attributes are on the edges of those buckets will be recommended to similarly aged users in an adjacent bucket, even if they would otherwise be suitable matches, thereby often categorising similarly-aged users as unsuitable for matching. It was subsequently demonstrated that calculating distance scores rather than bucketing continuous attributes improves results from RECON [150].

2. It does not account for the bias often caused by user popularity in its recommendations. Users with universally popular traits are likely to be recommended disproportionately, whereas less popular users might seldom be recommended.

2.4 Collaborative Filtering

Collaborative filtering has been the most successful method for making recommendations [39]. This section first describes collaborative filtering techniques for user-item recommender systems.

After that, reciprocal recommender systems based on collaborative filtering are described. Finally, a case study on a collaborative filtering algorithm that is used as a baseline, RCF, is described.

2.4.1 Collaborative Filtering for Recommendation

Collaborative filtering uses similarities between users to make recommendations. If Alice has watched and enjoyed *The Matrix* and *Apocalypse Now*, and Bob has watched and enjoyed both of those movies as well as *Die Hard*, we might infer that Alice's preferences are similar to Bob's, and make recommendations based on this.

Even at this basic level, it is clear that collaborative filtering has some advantages over content-based filtering. The data required to perform collaborative filtering is significantly easier to extract and interpret - only users, items and preferences are required. Recommendations are also not dependent on assumptions about which pieces of content are relevant: Alice might pick movies by genre or by favourite actor, but in either case she will be similar to users who have made choices based on the same motivations and therefore likely to receive accurate recommendations.

Collaborative filtering is commonly divided into memory-based methods and model-based methods. Because the steps involved in making recommendations for these two methods are quite different, they are discussed separately in Sections 2.4.1.1 and 2.4.1.2 respectively.

2.4.1.1 Memory-Based Collaborative Filtering

Memory-based collaborative filtering describes collaborative filtering strategies that calculate recommendations as required rather than doing significant precomputation, and using these to make recommendations. The most common of these is *Neighbourhood-Based Collaborative Filtering* [5]. This was one of the earliest models developed for collaborative filtering. This is described below as a representative example of memory-based collaborative filtering, and as a basis for the design of a number of reciprocal collaborative filtering algorithms.

Memory-based methods make recommendations by estimating user ratings for items they have not interacted with, and recommending those estimated to have high ratings. There are two steps to neighbourhood-based collaborative filtering: identifying similar users to the target user, and making recommendations from these users. Users' preference histories are represented as vectors. Similar users are then identified by calculating a *Correlation Coefficient* - a measure of how alike two vectors are - between them and other users. Predicted ratings are then calculated from ratings given by similar users. This is usually done by taking a weighted average of other scores from N users depending on their degree of similarity.

This process is illustrated by the example in Table 2.2 using fictional users of an online streaming service and their ratings for TV series. Series that the user has not previously interacted with are denoted by "?", and the objective of the example is to estimate a rating for Alice for *The Sopranos* so as to determine whether or not to recommend it.

Series → / User ↓	The West Wing	Friends	The Sopranos	24
Alice	7	5	?	9
Bob	9	5	9	7
Charlie	1	?	4	4
Dick	2	7	3	?

Table 2.2: Example Ratings for Series by Users

First, some notation is introduced to be used throughout this section. Collaborative filtering methods consider user-item ratings as a $m \times n$ *Ratings Matrix*, notated by $R = [r_{u,j}]$ for m users and n items, where $r_{u,j}$ is the rating by user u for item j. (The reasons for considering it a matrix rather than a series of vectors becomes clearer in discussions of model-based filtering in Section 2.4.1.2.) The set of items rated by a user u is defined as I_u. For example, I_{Alice} = {The West Wing, Friends, 24}. The intersection operation for two users $I_u \cap I_v$ is also commonly used to determine the ratings they have in common. For example, $I_{Alice} \cap I_{Dick}$ = {The West Wing, Friends}.

In order to know whether or not to recommend *The Sopranos* to Alice, we must estimate its rating. In this example, we use the *Pearson Correlation Coefficient* [24], which is a commonly used similarity measure in collaborative filtering [5]. Before this can be effectively used, the user ratings are first *mean normalized*. In our example above, Bob rates everything very highly, and Charlie gives everything a very low rating; mean normalization allows us to calculate similarities based on their relative preferences without a bias towards low or high numbers affecting this. The mean calculation μ_u for each user u is defined as:

$$\mu_u = \frac{\sum_{k \in I_u} r_{u,k}}{|I_u|} \tag{2.19}$$

The mean μ_u for the example in Table 2.2 is 7.0. We can then calculate the mean-centered rating for each user $s_{u,j}$ as:

$$s_{u,j} = r_{u,j} - \mu_u \tag{2.20}$$

The mean-centered ratings for each user are presented in Table 2.3.

The correlation coefficient between two users, $Sim(u,v)$ can then be calculated using Pearson's method:

$$Sim(u,v) = \frac{\sum_{I_u \cap I_v} s_{u,k} \cdot s_{v,k}}{\sqrt{\sum_{I_u \cap I_v} {s_{u,k}}^2} \sqrt{\sum_{I_u \cap I_v} {s_{v,k}}^2}} \tag{2.21}$$

The similarities calculated using Pearson's method are displayed in Table 2.4. The calculation is commutative, so the resulting table is symmetrical along the diagonal, and a user's similarity

Series → / **User ↓**	**The West Wing**	**Friends**	**The Sopranos**	**24**
Alice	0	-2	?	2
Bob	1.5	-2.5	1.5	-0.5
Charlie	-2	?	1	1
Dick	-2	3	-1	?

Table 2.3: Mean-Centered Ratings

Series → / **User ↓**	**Alice**	**Bob**	**Charlie**	**Dick**
Alice	1.0			
Bob	0.48	1.0		
Charlie	0.44	-0.37	1.0	
Dick	-0.83	-0.95	0.6	1.0

Table 2.4: Similarity ratings between pairs of users (2SF)

with themselves is always exactly 1. In our example, Alice's similarity with Bob based on Equation 2.21 is 0.48, her similarity with Charlie is 0.44 and her similarity with Dick is -0.83 (to 2sf). Neighbourhood methods estimate scores using the top k correlated users, where k is generally chosen based on empirical evidence from the application of the algorithm. Let $P_u(j)$ be the set of k closest users to u from the correlation coefficient who have rated item j. The prediction function for $\hat{r}_{u,j}$, the estimated score for user u and item j is as follows:

$$\hat{r}_{u,j} = \mu_u + \frac{\sum_{v \in P_u j} Sim(u,v) \cdot s_{v,j}}{\sum_{v \in P_u j} |Sim(u,v)|} \tag{2.22}$$

If we set $k = 2$ in our example, Alice's two closest users are Bob and Charlie so her estimated score for *The Sopranos* is calculated as follows:

$$\hat{r}_{Alice,TheSopranos} = 7.0 + \frac{0.48 \times 1.5 + 0.44 \times 1.0}{0.48 + 0.44} = 8.26 \tag{2.23}$$

This is a relatively high estimated score for Alice, so depending on estimated scores of other series, this might well be featured on the service.

There are some alternative methods of calculating similarities in memory-based collaborative filtering such as the *Cosine Similarity*. While there are some differences in the applications of these functions in data science in general, within the context of Recommender Systems the terms *similarity* and *correlation* are essentially interchangeable, with this thesis preferring the term *similarity*.

Memory-based methods have the advantage of being straightforward to design and implement. The main problem with memory-based methods is that they do not scale well [5]. For a service

with m users and a naive implementation, similarities between m^2 pairs of users must be calculated to make recommendations. A popular service might have millions of users and items, so the computational power required to calculate correlation coefficients becomes intractable.

Some solutions to this problem have been proposed, often involving heuristic rule-based clustering [103, 127] so that similarities only need to be calculated between subsets of users. These solutions do, however, exclude based on heuristics a certain amount of information that could potentially be used in recommendation. As a result, model-based implementations of collaborative filtering, which are able to make recommendations using all of the available information through precomputation and machine learning methods, are generally more accurate [27]. These are discussed in the next section.

2.4.1.2 Model-Based Collaborative Filtering

Memory-based collaborative filtering, described in Section 2.4.1.1, uses *lazy* methods to generate recommendations, doing the computations on demand to determine similar users. As discussed, this is often inefficient for large datasets without heuristic-driven preprocessing. Model-based collaborative filtering methods learn models that reduce the data down to important factors that describe the facets of the data important for recommendation, and that can then be used to make recommendations efficiently. This gives them several important advantages over memory-based methods:

1. Model-based methods are empirically more successful predictors of preference than memory-based methods. Competitions such as the Netflix Prize Challenge allow for the comparison of different types of algorithm on the same data; model-based methods are significantly more effective in general [22].

2. While memory-based methods often require computations, model-based methods generally only need a trained model to make predictions, which represents a summary of the relevant parts of the data. This allows model-based methods to be faster and more responsive.

3. Model-based methods are less influenced by random fluctuations in the data. An extremely high or low rating by a user in a neighbourhood method can significantly influence another user's recommendations. Model-based methods use global information to make recommendations, so this is unlikely to be a problem.

This section describes an example of a model-based method of recommendation, and explains how this method is representative and generalisable to other techniques while retaining the same basic principles. The method explained is the *Latent Factor Model*, which aims to simplify recommendation by extracting latent factors of items and users' preferences for these latent factors. Latent factors can be thought of as inherent attributes that all items possess, and are often used by people in everyday conversation. For example, if Alice wants to recommend a movie

to Bob, asking him for all the movies he's ever watched and his rating for them would be very arduous. Instead, she might ask him what genre he likes, or his favourite director. In this case, genre and director are latent factors of movies: inherent properties that all movies have which reduces a very large amount of information down to a much more manageable amount.

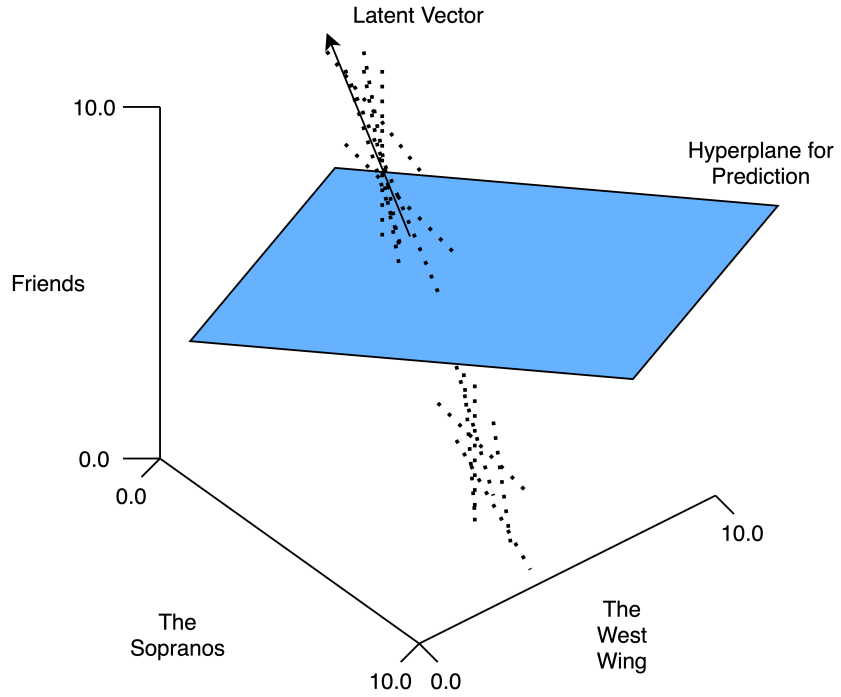


Figure 2.10: Using Similarities to Make Predictions. (Inspired by diagram on page 92 of [5].)

Recall that the user-item preference table can be considered a matrix. Model-based methods have their origins in linear algebra, where techniques for performing computations to reduce the complexity of matrices existed long before there was any need for recommendations. The m by n user-item matrix for a popular service is often very large, where m might be in the millions. However, because of similarities, the approximate dimensionality of the data is likely to be much smaller. For example, imagine a streaming service has three series: *Friends*, *The Sopranos* and *The West Wing*. Ratings between 0.0 and 10.0 are distributed as shown in Figure 2.10. After noise reduction, the ratings are distributed along a one-dimensional *Latent Vector*. Given the rating for one series, the ratings for the other two series can be accurately predicted by the intersection of the plane that the rating describes with the latent vector.

A much larger service with many more series is unlikely to have such a simple distribution, but we assume that similarities and inverse similarities between series means that the matrix describes a hyperplane with k dimensions, where k is the rank of the matrix. For a complete matrix with no noise, this hyperplane is described by the eigenvectors of the matrix, which can be found by *factorising* the matrix. There are a number of different methods for factorising complete matrices, such as the *Singular Value Decomposition*, which was used as early as the year 2000

for matrix factorisation in recommendation [124]. In early implementations such as this one, unknown cells in the matrix were filled with a fixed value because the SVD only operates on complete matrices. However, this often creates bias in the results [5], because most matrices found in recommendation environments are very sparse, so the filled values come to dominate the estimations.

Figure 2.10 shows this process. The ratings by a number of users for the three series in the diagram (*Friends*, *The Sopranos* and *The West Wing*) are represented by points on the diagram. The latent vector represents the trend of those ratings. A rating for *Friends* by a new user describes a plane, represented in the diagram in blue. The point in 3D space at the intersection between this plane and the latent vector predicts this user's rating for *The Sopranos* and *The West Wing*.

The alternative is to approximate matrix factorisation from the existing values using optimisation techniques found across machine learning, such as *Stochastic Gradient Descent* (SGD). This method was first described independently of recommendation in this context as part of the literature on *missing value analysis* - methods of estimating values from an incomplete matrix [6].

A ratings matrix R with dimensions $m \times n$ and rank k can be described exactly as the product of factors U (an $m \times k$ matrix) and V (a $k \times n$ matrix):

$$R = UV^T \tag{2.24}$$

In this formulation, U contains the k basis vectors for the column space of R. For the ratings matrix, the i the row of U, $\bar{u_i}$, is the *user factor*. Similarly, V contains the k basis vectors for the row space of R and the jth column of V, $\bar{v_j}$, is the *item factor*.

While factorisation methods aim to calculate U and V exactly, this is not possible for incomplete matrices. However, even for a sparse matrix, U and V can be estimated such that:

$$R \approx UV^T \tag{2.25}$$

For any k. Furthermore, it may be advantageous to choose k as lower than the actual rank of the matrix because noise in the similarities increase the rank of the matrix, and accounting for this increases the time and space complexity of calculating U and V without necessarily improving recommendations. Note that the result of this factorisation is that any single rating $r_{i,j}$ can be estimated from the product of its factors in U and V:

$$r_{i,j} \approx \bar{u_i} \cdot \bar{v_j} \tag{2.26}$$

Note the difference between this method and the memory-based filtering described in Section 2.4.1.1: although estimating U and V can be a complex operation, it is done as part of the

preprocessing step. From then on, estimating any rating is guaranteed to be $O(1)$, and estimating the ratings for all items for any user scales linearly with complexity $O(m)$ for m items.

In order to approximate the matrices of factors U and V using machine learning, we define a process where the error J is minimised in the objective function:

$$J = \frac{1}{2}||R - UV^T||^2 \tag{2.27}$$

Where $||\cdot||^2$ symbolises the sum of the squares of the matrix entries. The objective is then to minimise the error J and thus compute representative matrices of factors U and V Because there are unknown values in R, J cannot be calculated directly through this formula. It must therefore be redefined in terms of the entries of R that are known. Defining S as:

$$S = \{(i,j) : r_{i,j} \text{ is known}\} \tag{2.28}$$

Then values $r_{i,j}$ of R can be approximated by:

$$\hat{r}_{i,j} = \sum_{k=1}^{s} u_{i,s} \cdot v_{j,s} \tag{2.29}$$

Then the error $e_{i,j}$ for the estimation of an entry in R by the factorisation is given by:

$$e_{i,j} = \hat{r}_{i,j} - r_{i,j} \tag{2.30}$$

Of course, the error is only calculable for known values of R. The objective function in equation 2.27 can therefore be modified to:

$$J = \frac{1}{2} \sum_{(i,j) \in S} e_{i,j}^2 \tag{2.31}$$

The process for minimising J is known as *gradient descent*. U and V are initialised to random values, the error is calculated, and then small steps are taken down the gradient of the objective function to drive the error towards its minimum value. In order to calculate the adjustments to the values of U and V that reduce the error, we take partial derivatives of J, which determines the gradient of the objective function with respect to $u_{i,q}$ and $v_{j,q}$.

$$\begin{aligned} \frac{dJ}{du_{i,q}} &= \sum_{j:(i,j) \in S} (r_{i,j} - \sum_{s=1}^{k} u_{i,s} \cdot v_{j,s})(-v_{j,q}) \\ &= \sum_{j:(i,j) \in S} (e_{i,j})(-v_{j,q}) \end{aligned} \tag{2.32}$$

$$\begin{aligned}\frac{dJ}{dv_{i,q}} &= \sum_{j:(i,j)\in S}(r_{i,j} - \sum_{s=1}^{k} u_{i,s}\cdot v_{j,s})(-u_{j,q}) \\ &= \sum_{j:(i,j)\in S}(e_{i,j})(-u_{j,q})\end{aligned} \tag{2.33}$$

The process *Stochastic Gradient Descent* (SGD) uses the derivative of the error function directly to make updates to individual components of U and V to move them closer to an accurate estimation of R. It uses a step size α to moderate the size of the change, generally a small value relative to the size of the scores in R because larger steps are more likely to overshoot the minimum. The update formulae, derived from equations 2.32 and 2.33 respectively, are:

$$u_{i,q} \leftarrow u_{i,q} + \alpha \cdot e_{i,j} \cdot v_{j,q} \forall q \in \{1...k\} \tag{2.34}$$

$$v_{i,q} \leftarrow v_{i,q} + \alpha \cdot e_{i,j} \cdot u_{j,q} \forall q \in \{1...k\} \tag{2.35}$$

U and V are initialised to random values, and then the known values of R are repeatedly cycled through and the above updates applied in order to drive the estimations $\hat{r}_{i,j}$ of values of R closer to their actual values, and therefore determine matrices of factors that can be used to effectively estimate the unknown values.

2.4.1.3 Collaborative Filtering for User-Item Recommender Systems

Collaborative filtering has been used as part of recommender systems since the term was coined in 1992 [50]. Over this time, there have been a number of significant advances in the field. These have covered both memory-based and model-based collaborative filtering. This subsection briefly describes the state-of-the-art in collaborative filtering. The depth of research in user-item collaborative filtering is also useful as a point of comparison with reciprocal recommendation.

A number of improvements have been made to memory-based methods of collaborative filtering, which are still popular due to the relative simplicity of implementation. For example, Herlocker et al. use a neighbourhood model that uses concepts such as trust to weight similarity coefficients [56]. There are a number of proposed solutions for clustering in neighbourhood models that help to solve the inherent complexity in scaling the [103, 127].

Latent factor models for conventional recommender systems have been frequently and thoroughly investigated in the literature, and have been extensively studied. Koren, Bell and Volinsky provide an overview of the various modern methods of computing latent factor models [23]. Matrix factorization techniques such as Singular Value Decomposition (SVD) can be used to extract vectors that describe the user-item matrix from similarities between user interactions, and similar items being interacted with. Mathematically, these techniques can only be used on complete

matrices. However, it is unrealistic to assume there is complete information about every user's opinion on every item. Previous works used various methods to estimate the missing values in the user-item matrix (e.g. [125]), but modern methods initialize latent factor vectors with random values from a distribution, and then they learn the correct values via optimisation techniques such as gradient descent [4, 73, 106]), as described in Section 2.4.1.2. Latent factor models have been relatively successful, and competitions such as the Netflix Prize have demonstrated that on conventional recommender systems, they are superior to nearest neighbour implementations [22].

A number of different machine learning technologies have been used to create models for recommendation. Models based on random forests are popular for their simple implementation, and often achieve strong results [9, 159]. Approaches based on neural networks are also pervasive [161], and used across various industries such as video [38] and app recommendation [34]. In particular, *Restricted Boltzmann Machines* have been popular as a method for creating latent factor models using neural networks [123].

2.4.2 Collaborative Filtering for Reciprocal Recommendation

Reference	Year	Field	Recommendation Method
Krzywicki et al. [75]	2010	Dating	Memory-based nearest neighbour
Xia et al. [150, 151]	2015	Dating	Memory-based nearest neighbour
Al-Zeyadi et al. [12]	2017	Dating	Graph-based collaborative filtering
Kleinerman et al. [70]	2018	Dating	Memory-based with AdaBoost classifier weighting
Vitale et al. [144]	2018	Dating	Collaborative filtering-based clustering with performance guarantees
Neve et al. [94]	2019	Dating	Memory-based nearest neighbour testing aggregation operators
Neve et al. [96]	2019	Dating	Model-based latent factor
Ramanathan et al. [116]	2020	Dating	Model-based latent factor

Table 2.5: Collaborative Filtering Reciprocal Recommender Systems

Thus far, collaborative filtering trends in reciprocal recommendation have not been dissimilar to those in conventional user-item recommendation, which is to say, they have generally outperformed their content-based counterparts. This is not surprising: collaborative filtering is able to bypass the often very complex data on social services to make recommendations based on similarities.

The earliest collaborative filtering algorithm was RCF [150]. This is a neighbourhood-based method tested on data from an online dating service. RCF is commonly used as a baseline and is discussed in detail as a case study in Section 2.4.2.1 so the discussion is omitted here.

Kleinerman, Rosenfeld, Ricci & Kraus designed *Reciprocal Weighted Score* (RWS): an RRS based primarily on RCF, but using a model-based approach to take user popularity into proper consideration [70]. The solution proposed in RWS consists of finding for each user x a weight α_x which represents the influence of that user on successful interactions. If the user's own preference score being high implies a successful interaction, that user has a higher importance, whereas if their partner's preference determines most interactions, they are assigned lower importance. Although RWS did not outperform RCF in the appeal of the recommendations to the users, RWS was demonstrated to outperform baseline RCF in terms of reciprocal precision and recall, indicating that the recommendations, while not as appealing as the original ones, were an improvement in terms of their satisfaction to both parties.

Compared to the detail of the literature on model-based user-item recommender systems, there have been few model-based RRSs. Neve et al. [96] is the first example, which is described in detail in Chapter 3. Ramanathan et al. [116] designed a recommender based on this for a different dating service, and were able to replicate the results.

2.4.2.1 Collaborative Filtering Case Study: RCF

Reciprocal Collaborative Filtering (RCF), a collaborative filtering based algorithm, was able to significantly improve on RECON's results [150]. RCF also calculates scores for multiple users, but unlike RECON, it uses a nearest-neighbor collaborative filtering approach, where user a's likelihood to like user b is estimated from their similarity to other users who liked user b. RCF's evaluation proved the algorithm to significantly outperform RECON.

RCF is commonly used as a baseline for testing collaborative filtering-based RRSs on small datasets (e.g. [70], [69]). It is a nearest neighbour-based collaborative filtering algorithm that retrieves nearest neighbours based on other users who have liked the user in question. For example, to calculate the likelihood that a user a will like a user b, RCF calculates the similarity between a and other users n who have previously liked b. A high degree of similarity implies that a will probably like b, and vice-versa. The similarity between users a and n, based on the well-known Jaccard Coefficient, is defined as:

$$Similarity_{a,n} = \frac{IFrom_a \cap IFrom_n}{IFrom_a \cup IFrom_n} \tag{2.36}$$

Where

$$IFrom_a = \{b : b \text{ has received a Like from } a\} \tag{2.37}$$

$$ITo_b = \{a : a \text{ has sent a Like to } b\} \tag{2.38}$$

In the following algorithm, let a be the current user. Then:

$$Candidates = \{b : b \text{ is a possible recommendation for } a\} \tag{2.39}$$

In the online dating domain, this may be the entire set of opposite-sex users, or there may be restrictions (for instance, opposite-sex users within a certain geographical distance from the current user). $Recs$ is the output of the algorithm, a set of pairs such that:

$$Recs = \{(b, reciprocalScore_{a,b})\} \tag{2.40}$$

Where $reciprocalScore_{a,b} \in [0,1]$ is the aggregation of two unidirectional preference scores between current user a and recommendation candidate b.

The full algorithm is described in Algorithm 1. Note that the normalising factor in lines 11 and 12 is necessary because the number of neighbours of a will depend on the number of users who have liked b, and will therefore vary from user to user.

Algorithm 1 RCF Algorithm for target user a

1: $Recs \leftarrow \emptyset$
2: **for all** $b \in Candidates$ **do**
3: $score_{a,b} \leftarrow 0$
4: $score_{b,a} \leftarrow 0$
5: **for all** $n \in ITo_b$ **do**
6: $score_{a,b} \leftarrow score_{a,b} + Similarity_{a,n}$
7: **end for**
8: **for all** $n \in ITo_a$ **do**
9: $score_{b,a} \leftarrow score_{b,a} + Similarity_{b,n}$
10: **end for**
11: $score_{a,b} \leftarrow \frac{score_{a,b}}{|ITo_b|}$
12: $score_{b,a} \leftarrow \frac{score_{b,a}}{|ITo_a|}$
13: $reciprocalScore_{a,b} \leftarrow$ **Agg**$(score_{a,b}, score_{b,a})$
14: $Recs \leftarrow Recs + (b, reciprocalScore_{a,b})$
15: **end for**
16: **return** $Recs$

The complexity of RCF depends on the computation of the similarity of user a with the n users who have liked user b. If all users have an average of k interactions with other users, the process will require approximately nk computations. However, if by chance the n users have significantly more than an average number of interactions, the computation could take much longer than this, which is deeply undesirable in an online service.

2.5 Hybrid Filtering

Hybrid methods of recommendation combine one or more recommendation methods in order to make a recommendation. In the literature, the most common use of hybrid filtering is to describe the combination of content-based and collaborative filtering to make recommendations [20, 29]. However, it can also refer to any combination of methods, including combining model- and memory-based collaborative filtering [117] and systems combining types of recommendation other than content-based and collaborative filtering, such as knowledge-based filtering [28].

Designers of hybrid systems are particularly keen to overcome the weaknesses of the two families of recommender systems. In the case of collaborative filtering, the cold-start problem [76, 129] can be overcome through content-based elements. In the case of content-based filtering, filter bubbles [100] can be broken by introducing collaborative filtering elements.

2.5.1 Hybrid Recommendation

Burke [29] describes a number of different methods of combining different recommender system techniques into a single hybrid system. This thesis uses his terminology to describe the different types of hybrid system.

Weighted Hybrid Recommender Systems combine two or more recommender systems together by assigning a weight to each of the two systems, and treating the predicted score as a weighted average of the two results. This has the advantage of being simple to implement. The weights could be fixed, or dynamic based on the accuracy of the algorithm under different weights [108].

Switching Hybrid Reccommender Systems makes recommendations from one recommender system depending on the user and the item, chosen based on some criterion. The most common way to do this is based on some measure of confidence: if a primary system cannot make recommendations with sufficiently high scores, the secondary (or even tertiary) system is used and may be able to generate recommendations with higher confidence. For example, [136] switches based on the extent to which each technique can accurately estimate a user's current ratings, which is potentially a good predictor for that technique's capacity for making new accurate recommendations. Switching systems can potentially overcome the weaknesses of each individual technique used in situations, but introduce more complexity than weighted systems because the criteria for switching must be decided, and must be demonstrably more accurate than using either individual system, which is often difficult to demonstrate in real settings.

Mixed Hybrid Systems present recommendations from more than one recommender system together in a list for the user to decide from. Instead of choosing the effective technique from the data, this places the burden on the user to pick the better recommendations for themselves [37]. However, mixed hybrid systems do not inherently solve the problem of how to rank recommendations from different sources, and depending on the context, some users may only view the first

few items in their recommendation list. A heuristic from switched or weighted hybrid systems must be used to decide which recommendations come first in this case.

Cascading Hybrid Systems use one system to generate an initial list of candidate recommendations, and then a second system to narrow down or rank this list. As with other hybrid systems, the intention is to address the weaknesses in two different techniques to create an optimal list of recommendations [77].

2.5.2 Hybrid Reciprocal Recommendation

Reference	Year	Field	Hybridisation
Akehurst et al. [11] [10]	2011	Dating	CB/CF Cascading
Zhang et al. [160]	2016	Education	CB/CF Weighted
Qu et al. [115]	2018	Dating	CB/CF Weighted
Neve et al. [97]	2019	Social	CB/CF Weighted

Table 2.6: Content-Based Reciprocal Recommender Systems

Compared with the other techniques, there have been relatively few contributions to the field of hybrid reciprocal recommendation. This is mainly because RRS is still a relatively small field compared to user-item recommendation, and a deeper knowledge of both collaborative filtering and content-based filtering is required to determine which is more generally effective and how best to combine them.

The earliest hybrid approach was by Akehurst et al. [10, 11], with the design of CCR (content-collaborative recommender). CCR used an original form of nearest neighbour collaborative filtering as part of its reciprocal recommendation. It defines *interaction groups* for a user x based on the users whom they have liked and the users who have liked them. It also defines a distance metric between users based on their similarity in terms of content-based attributes such as age and location, under the assumption that similar users have similar preferences. The hybrid reciprocal recommender uses a cascading approach to calculate the final recommendation, with collaborative filtering refining initial lists generated by the content-based approach.

The most recent contribution to the literature by Neve et al. [97] uses a weighted combination of a semantic similarity-based content-based filtering method with latent factor collaborative filtering to make recommendations. This is discussed in more detail in Chapter 4.

2.6 Peripheral Topics

This section reviews other techniques for reciprocal recommmendation that do not fit into the umbrellas of content-based, collaborative or hybrid filtering. It also covers areas that are relevant to the discussion of reciprocal recommendation, including related fields such as *Multistakeholder Recommendation* and *Social Network Recommendation*.

2.6.1 Other Methods of Reciprocal Recommendation

Although most reciprocal recommender systems adapt techniques from collaborative filtering or content-based filtering, some adapt other methods of matching people together from areas of computer science besides user-item recommendation. This section briefly explores research that uses these techniques.

The most common of these methods are solutions to the *Stable Matching* problem. In the Stable Matching problem, n men and n women have ranked each other in order of preference. The problem is solved by n pairs of men and women, such that there exists no pair that would rather have each other than any of the existing pairs.

In 2013, Yin et al. implemented the *Gale-Shapley* algorithm [47] as part of an RRS algorithm [156] to match users with social event organisers. The Gale-Shapley algorithm works as follows:

1. Each of the n men *proposes* to their highest rated woman. Each woman accepts the proposal from their highest rated man, and these pairs are considered provisional.

2. Each of the remaining men who are not part of the set of provisional pairs then repeats the process, proposing to any of the women including those in provisional pairs.

3. The algorithm terminates when every man and woman is part of a pair.

While this algorithm does find optimal pairs and is guaranteed $O(n^2)$, the challenge in using it for reciprocal recommendation is that no service includes ratings from every member to every potential match; indeed, the ratings matrix is generally sparse. Yin et al.'s method uses a content-based utility function based on a cosine similarity between a user's preferences and an event's attributes, and incorporates unidirectional social factors such as preference of the user for events that include their friends to estimate unidirectional preferences for each user and event.

Xia et al. developed a reciprocal recommender, *WE-Rec* based on *Walrasian Equilibrium* [149]. The Walrasian Equilibrium is a concept from economic systems where the objective is to meet preferences of buyers and sellers equally in such a way that equilibrium is maintained. Xia et al. accomplish this by defining a utility function based on the similarity between user profiles and using this to generate recommendation lists between users based on the output from the utility function.

2.6.2 Multistakeholder Recommendation

A *Multistakeholder Recommender System* is any recommender system that takes into account the preferences of multiple parties when generating recommendations. Abdollahpouri et al. [1] define a *Stakeholder* in this setting as "any group or individual that can affect, or is affected by, the delivery of recommendations to users". Reciprocal recommendation is therefore a subtype of multistakeholder recommendation, with two parties who are equally affected. This section

provides a brief background of the concepts in multistakeholder recommendation, as some of these concepts are relevant to reciprocal recommender system design.

Multistakeholder recommender systems were originally designed for e-commerce services. The field defines *Consumers*, who are viewing the recommendations provided by the system, and also *Producers*, who are providing the recommended objects. While traditional recommender systems focus entirely on optimising the experience for the consumer, producers often have their own objectives.

Multistakeholder recommendation generally becomes reciprocal recommendation when it consists of two parties whose objectives are aligned. For example, in an online dating setting, the objective is to match two people who are looking for each other. However, there are several other types of multistakeholder recommendation problems, each of which have different common approaches used to solve them.

The most common of these is *Value-Aware Recommendation*. While standard recommender systems aim to maximise value for consumers, value-aware recommenders also take into account the value of a recommendation for the producer. For example, Nguyen et al. [99] design a multistakeholder recommender system for hotel recommendation from third parties. The service receives a different commission depending on the hotel, so the system aims to optimise recommendation based on both users' preferences and commission. In the general case, this is often done by training a system to maximise the values of two optimization functions [145], one which represents the best case recommendation for the consumer and another which maximises profits for the producer.

A second example of multistakeholder recommendation is *Fairness-Aware Recommendation*. Because most recommender systems are trained on existing datasets, they are likely to adopt biases inherent in these datasets [30]. These biases might be reflected on the consumer side as, for example, a difference in recommendations based on age or gender. They might also cause problems for producers, for instance by recommending certain musicians significantly more than others. Although fairness is in the interests of everyone using the system, it represents a different optimisation function to that of standard recommender systems, and is therefore considered a separate stakeholder.

2.7 Summary

This chapter began by introducing machine learning fundamentals, including various methods of training that are commonly used in recommender systems research, such as neural networks and random forest models, with examples of cases where they have been used in this field. This section in particular focused on how these methods are used for extracting features from complex data, which is commonly required for content-based filtering.

Subsequent sections in this chapter covered the three subdivisions of recommendation:

content-based filtering, collaborative filtering and hybrid filtering. For each type, its respective section described basic theory, a brief review of the technique in user-item recommendation and a more thorough review of the most important developments in reciprocal recommendation.

The final section described peripheral techniques for reciprocal recommendation, including the stable matching problem and multistakeholder recommendation.

The following three chapters will cover the original contributions made to the three fields of content-based, collaborative and hybrid filtering respectively over the course of this PhD.

CHAPTER 3

COLLABORATIVE FILTERING

This chapter describes contributions made to Collaborative Filtering. As discussed in Chapter 2, this is recommendation based on correlations between user preference histories without analysing the context. It first describes aggregation strategies for combining two unidirectional preference scores into a single bidirectional preference relation. This is important for CF, as all current techniques use two symmetrical models to calculate preference of two users for each other before combining them. It then describes *LFRR*, a novel CF technique for RRS based on latent factor models.

=======

3.1 Introduction

As described in Chapter 2, collaborative filtering algorithms make predictions about user preferences based on correlations in the data. If a user a liked users x, y and z and a user b liked x and y, a collaborative filtering algorithm might recommend z to b based on b's correlation with a.

Collaborative filtering algorithms are broadly divided into two categories: memory-based and model-based systems. Memory-based systems are generally based on nearest-neighbour algorithms, where in order to make recommendations for Alice, her most similar users are calculated in real time based on a similarity metric such as the *Pearson Correlation Coefficient*. Model-based systems base their similarity computation on a pre-trained model, often a *latent factor model*, which abstracts the complexity of a large number of comparison into latent attributes of items, and users' preferences for each of these attributes.

Prior to the work described in this thesis, collaborative filtering work on RRSs used memory-based methods for making recommendations. While this produces strong results on smaller

datasets, computations take unreasonable times on larger datasets, and would not therefore be practical to implement on mainstream dating and social services. This chapter introduces a reciprocal recommender system based on latent factor models and evaluates it against the current state-of-the-art memory-based collaborative filtering method.

Collaborative filtering reciprocal recommender systems generate two unidirectional preference scores and then aggregate them into a single reciprocal score. In the literature, this is done with the harmonic mean, but the use of the harmonic mean is not justified in the papers in which it is used. This chapter presents several alternative aggregation functions, and the evaluation demonstrates the importance of the choice of function in this context.

3.2 Background

This section briefly reviews the literature relevant collaborative filtering reciprocal recommendation and the systems described in this chapter. For a more in-depth literature review of all topics and papers surrounding this area, see Chapter 2.

3.2.1 Collaborative Filtering

Collaborative filtering has been very successful in recommender system design. Neighbourhood methods have been able to make recommendations with a high degree of accuracy, and have seen significant development, both in accuracy and in efficiency [127, 128]. However, model-based methods have in general been significantly more successful [22, 39], and this chapter focuses on these.

There are a number of different ways of developing models for collaborative filtering [5]. These incorporate various machine learning technologies such as random forests [9] and neural networks [34]. In particular, latent factor models have received significant attention, and developed in a number of papers as solutions to a variety of recommendation problems [4, 126].

3.2.2 Collaborative Filtering for Reciprocal Recommendation

As a strategy for reciprocal recommendation, collaborative filtering proceeded content-based filtering. The earliest collaborative filtering RRS in the literature is RCF [150], which is a nearest-neighbour algorithm using similarity comparisons to proximate users to make predictions about preferences.

Following the publication of Xia et al.'s [150] paper on RCF, a number of other algorithms were developed in a similar vein, often using RCF as a basis. For example, Kleinerman et al. [69] developed an extension to RCF that used an AdaBoost classifier to ensure fairness, and demonstrated that enforced fairness improved reciprocity on an online dating service.

More recently, following the advances described in this chapter, latent factor model-based algorithms have been more popular in reciprocal recommendation. As described in Chapter

2, latent factor models use matrix factorisation to extract latent attributes of items and user preferences for those attributes. For example, Ramanathan et al. [116] developed a latent factor model-based algorithm for reciprocal recommendation in online dating.

3.3 Aggregation Strategies for Collaborative Filtering

RRSs in the literature apply an aggregation process to two unidirectional preference scores that indicates how likely two users are to *like* or *match* with each other. The most common strategy is to calculate two unidirectional preference scores representing how much two users are to like each other based on conventional recommender system technology, and then they combine both scores into one (e.g. [113], [150]). Since RECON [113], the first RRS in the literature, all subsequent systems have used the harmonic mean to combine these two scores. The harmonic mean is described by the formula:

$$H = \frac{2x_1x_2}{x_1 + x_2} \tag{3.1}$$

The choice to use the harmonic mean is assumed to have been arbitrary, as there is no explanation justifying this choice in the existing RRS literature.

RCF [150], the first collaborative filtering RRS, which is often used as a benchmark for new algorithms, is used as a baseline to test the aggregation operators. RCF, like other RRSs, calculates two preference scores and combines them using the harmonic mean. The novel contribution in this study consists of exploring a number of alternative aggregation functions to the harmonic mean for combining two preference scores into a single one in RRSs. Specifically, three Pythagorean Means: the harmonic, arithmetic and geometric means. An aggregation function with mixed behaviour is also considered, namely the uninorm [154]. This exploration is important because combining the two preference scores plays a crucial role in the reciprocal recommendation process, thereby differentiating it from a conventional recommender system. Even in a situation where there is no information about the two users beyond their preference scores, the way in which preference scores are combined could have a profound impact on the recommendations produced. For example, on an online dating service, it is not ideal for very strong preferences in one direction to compensate for weak preferences in the other direction, as the result would be a recommendation that could not lead to a match. In order to examine the results in different settings, these aggregation functions are tested on a dataset from a popular online dating service.

There was a significant difference in the effectiveness of different aggregation functions. Based on this, RRS algorithms designed in future that rely on combining two unidirectional preference scores should rely on evidence-based decision about the aggregation function they use to combine preference scores.

3.3.1 Aggregation Functions

The arithmetic mean, geometric mean, and harmonic mean make up the classical means. All three of these have been widely used in various fields, including statistics and machine learning. Of these three, only the harmonic mean has been used for aggregating unidirectional preference scores in reciprocal recommender systems. Without loss of generality, this investigation is limited to aggregation operators in the unit interval, as all preference scores occur in this interval.

The arithmetic mean for two values x_1 and $x_2 \in [0,1]$ is defined as

$$M(x_1, x_2) = \frac{x_1 + x_2}{2} \tag{3.2}$$

The *arithmetic* mean is the most commonly used averaging function. The general arithmetic mean is not a robust statistic and greatly influenced by outliers when averaging a large number of results. However, the arithmetic mean of two values will coincide exactly with the middle of the two values. This means that it may be more appropriate in certain applications of RRSs, as it accurately expresses the two users' preferences for each other without being *biased* towards the higher or lower score.

The geometric mean for two values is defined as

$$G(x_1, x_2) = \sqrt{x_1 x_2} \tag{3.3}$$

The *geometric mean* is commonly used in business and finance, to calculate averages in situations where percentages accumulate over time, such as portfolio growth rates. In the case of two values, the smaller value exerts a greater influence over the recommendation. This is a desirable trait for RRSs: if one of the preference scores is too low, that user is likely to reject the recommendation, making it useless, even if the other party's score is extremely high.

The harmonic mean for two values is defined as

$$H(x_1, x_2) = \frac{2x_1 x_2}{x_1 + x_2} \tag{3.4}$$

The *harmonic mean* is the smallest of the three means and, similar to the geometric mean, more significantly influenced by the smaller of the two values than by the larger. The harmonic mean is the only aggregation function that has been used in RRSs in the past.

The uninorm is also a potentially useful aggregation operator for RRSs. Uninorms are a generalisation of the t-norm and t-conorm classes of aggregation operator. A t-norm is a mapping $T : [0,1] \times [0,1] \to [0,1]$. A t-norm is associative, symmetric and has a neutral element of 1. Conversely, a t-conorm is a mapping $S : [0,1] \times [0,1] \to [0,1]$ exhibiting similar properties as t-norms but having a neutral element of 0. Uninorms [154] are similar to this definition, with

the difference that the neutral element can lie anywhere in the interval $]0,1[$. The cross ratio uninorm is used as a representative example of this family of functions [16], defined as

$$U(x_1,x_2) = \frac{x_1x_2}{x_1x_2 + (1-x_1)(1-x_2)} \tag{3.5}$$

Uninorms exhibit the *self-reinforcement property*. This means that 1) two low values are aggregated to produce a lower result 2) two high values are aggregated to produce a higher result. Even more so than the classical means, the result of this is that two relatively high bidirectional scores are more likely to be aggregated to a high reciprocal score than a very high score is to compensate for a low score. Intuitively, this is desirable behaviour for a reciprocal recommender system: a positive match is more likely to occur where both users have a moderate preference for each other than in the case of only one extremely strong unidirectional preference.

3.3.2 Methodology

3.3.2.1 RRS Implementation

RCF [150] is implemented as the base algorithm for testing aggregation functions in the computation of reciprocal preference scores. As described in Chapter 2, RCF calculates preference scores between user a and user b. The preference of user a for user b is calculated as the similarity between the behaviour of a and other users who have interacted positively with user b. The similarity between users a and n is defined as:

$$Similarity_{a,n} = \frac{IFrom_a \cap IFrom_n}{IFrom_a \cup IFrom_n} \tag{3.6}$$

Where

$$IFrom_a = \{b : b \text{ has received a Like from } a\} \tag{3.7}$$

This similarity measure, normalised across users, can be used to determine if a user a is similar to other users who have shown interest in user b, and therefore whether user a is likely to have a preference for b or not. The full algorithm is outlined below, and this is used to test the aggregation functions. Note that below, besides the above formulae, a normalising factor ITo is also introduced, defined as:

$$ITo_a = \{b : b \text{ has Liked } a\} \tag{3.8}$$

The baseline RCF algorithm is outlined in pseudocode below.
In the above algorithm, the **Agg** function represents the aggregation function used to combine the preference relations from a to b and vice versa into a single reciprocal score.

Algorithm 2 RCF Recommender System

1: $Recs \leftarrow \emptyset$
2: **for all** $a, b \in Candidates$ **do**
3: $score_{a,b} \leftarrow 0$
4: $score_{b,a} \leftarrow 0$
5: **for all** $n \in ITo_b$ **do**
6: $score_{a,b} \leftarrow score_{a,b} + Similarity_{a,n}$
7: **end for**
8: **for all** $n \in ITo_a$ **do**
9: $score_{b,a} \leftarrow score_{b,a} + Similarity_{b,n}$
10: **end for**
11: $score_{a,b} \leftarrow \frac{score_{a,b}}{|ITo_b|}$
12: $score_{b,a} \leftarrow \frac{score_{b,a}}{|ITo_a|}$
13: $reciprocalScore_{a,b} \leftarrow \mathbf{Agg}(score_{a,b}, score_{b,a})$
14: $Recs \leftarrow Recs + (b, reciprocalScore_{a,b})$
15: **end for**
16: **return** $Recs$

3.3.2.2 Aggregation Function Implementation

Four aggregation functions have been implemented as part of **Algorithm 2, line 13**: 1) the arithmetic mean, 2) the geometric mean, 3) the harmonic mean and 4) the uninorm described in equation 3.5.

3.3.3 Evaluation

In this section, the method and metrics used to evaluate the recommender system with various aggregation functions and the results of the evaluation are outlined.

3.3.3.1 Experimental Setup

In order to test the algorithm, data was sampled from one month of activity. There are two reasons for this choice. Firstly, many users are not active on the site for long periods of time, and there are fewer connections between users who are active several months ago and those who are currently active. Secondly, users often change their preferences over time, and using only recent data is therefore likely to make the most satisfactory recommendations.

In addition to narrowing down the data to one month, there were three limitations placed on the data used in the evaluation:

1. Users who have sent at least 10 *Likes*, to prevent the cold-start problem [76].

2. Users living in Tokyo or its suburbs, which are an active hub of users, and they can easily meet each other in person without geographical limitations.

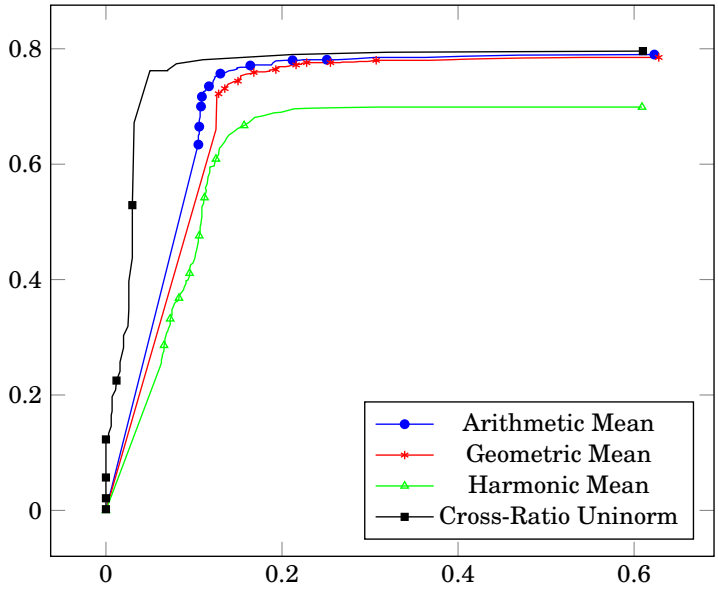


Figure 3.1: ROC curve obtained for each aggregation function considered in the RRS model.

3. Users between 18 and 30 years of age, for the same reason: these users make up the vast majority of active users.

The first of these is because RCF, like many collaborative filtering algorithms, suffers from the cold-start problem. There are no published effective solutions for the cold-start problem with RCF.

3.3.3.2 Results

The arithmetic mean in general led to high recall and relatively low precision compared to the other metrics. This is not surprising: the arithmetic mean's value lies in the middle of the two preference scores, and is therefore as likely to filter out positive results as it is to recommend negative results. The geometric mean performed in many respects very similarly to the arithmetic mean.

The best F1 score as found by varying the threshold for recommendation and its associated precision and recall are shown in Table 3.1. The harmonic mean and uninorm both generally maintained a higher precision than the other two classical means. However, the harmonic mean's F1 score was diminished by a relatively low recall, while the uninorm maintained a high precision with a similar recall to the geometric and arithmetic means.

In RRSs, precision is particularly important: it is much more important that all the recommendations are good than the fact of finding all the recommendations [21]. It is unlikely that a user will have time to look at all possible recommendations, but likely that a user will quickly lose trust in a system that is producing recommendations she is not interested in. Although the best precision of the arithmetic and geometric means is marginally higher than the harmonic mean's,

Algorithm	**Best Precision**	**Best Recall**	**Best F1 Score**
Arithmetic Mean	0.86	**0.79**	0.802
Geometric Mean	0.851	0.785	0.787
Harmonic Mean	0.835	0.699	0.736
Uninorm	**0.955**	0.762	**0.847**

Table 3.1: Best results obtained by varying the thresholds for different aggregation functions

in general, the harmonic mean's precision was more consistently high across all thresholds, whilst the uninorm's was much higher.

Of the three classical means, the arithmetic mean has the highest recall, and was therefore able to retrieve the largest number of total correct recommendations. However, the uninorm significantly outperformed the three classical means in F1 Score and Precision, and therefore both retrieves correct rather than incorrect recommendations, and provides the best balance between precision and recall.

3.4 Latent Factor-Based Collaborative Filtering

Many modern recommender systems take advantage of Latent Factor (LF) models to make recommendations with successful results (e.g. [4], [73]). LF algorithms use matrix factorization to generate vectors of factors that describe correlations in the original preference matrix, thereby condensing correlations between preferences into a much smaller number of factors. This often results in better predictions on very large, sparse matrices, which is often the case in online dating, where most users only indicate a preference for an unduly small number of other users. To the extent of our knowledge, LF-based solutions have never been investigated and applied to RRSs, in spite of the clear potential these models have as evidenced by their prevalence in the landscape of traditional recommender systems. Therefore, how LF models can influence the process of making effective reciprocal recommendations was an unaddressed research question in the field.

To overcome the aforementioned limitations inherent in previous RRS solutions, a novel reciprocal recommender system based on latent factor models was developed. This approach requires two latent factor models to be generated - one that determines the preference of individual male users for female users, and one that determines the preference of individual female users for male users. Many previous approaches used datasets of fewer than ten thousand instances [13, 101], which is not representative of the size of many real-life applications for RRSs. This method is effective on datasets containing hundreds of thousands of users and millions of instances. The aggregation functions evaluated in the previous section are used to combine the two unidirectional preference scores into a single reciprocal score and investigate their effect on the resulting recommendations.

3.4.1 Methodology

In this section, LFRR, a novel reciprocal recommender system approach based on latent factor models, is described. In order to account for the bidirectional nature of reciprocal recommendation, two latent factor models are trained; one to indicate male users' preferences for female users, and one to indicate female users' preferences for male users. These two preference metrics are subsequently combined using aggregation operators to indicate which are likely to be a positive match, and therefore which recommendations to display.

LFRR is designed for use with online dating, and in particular, it has been evaluated for its use on a popular online dating platform. Users must send each other a *Like* before they are able to communicate with each other by messages. Much of the message data is ambiguous, as users often quickly exchange contact details and move their communications off the service. However, users who succeed in achieving a large number of *Matches* are more likely to subsequently find a relationship. The objective is therefore to maximise the number of *Matches* as a proxy for helping the largest possible number of users to succeed.

3.4.1.1 Latent Factor Model for Reciprocal Recommendation

Machine learning-driven latent factor models initialize latent factor vectors to random values from a distribution, and then aim to minimise the error on the known ratings. Specifically, the likelihood of user a liking user b is calculated by the dot product between user a's feature vector p_{u_a} and user b's feature vector q_{u_b}. Using the training set of known ratings $R = (r_{u_a,u_b})_{rows \times columns}$, the error can be calculated with regularization parameter λ.

$$\min_{q,p} \sum_{(u_a,u_b) \in R} (r_{u_a u_b} - q_{u_b}^T p_{u_a})^2 + \lambda(||q_{u_b}||^2 + ||p_{u_a}||^2) \tag{3.9}$$

It can then be minimised by gradient descent. Because the dataset contains sparse, explicit data, stochastic gradient descent was used for the minimisation. This tends to perform better on explicit data than the other common method, Alternating Least Squares, which tends to give better results with dense, implicit data [5]. Stochastic gradient descent calculates the error for each individual datapoint.

$$e_{u_a u_b} = r_{u_a u_b} - q_{u_b}^T p_{u_a} \tag{3.10}$$

It then modifies the relevant feature vectors in the negative direction of the gradient proportional to the learning rate, γ

$$q_{u_b} \leftarrow q_{u_b} + \gamma(e_{u_a u_b} p_{u_a} - \lambda q_{u_b}) \tag{3.11}$$

$$p_{u_a} \leftarrow p_{u_a} + \gamma(e_{u_a u_b} q_{u_b} - \lambda p_{u_a}) \tag{3.12}$$

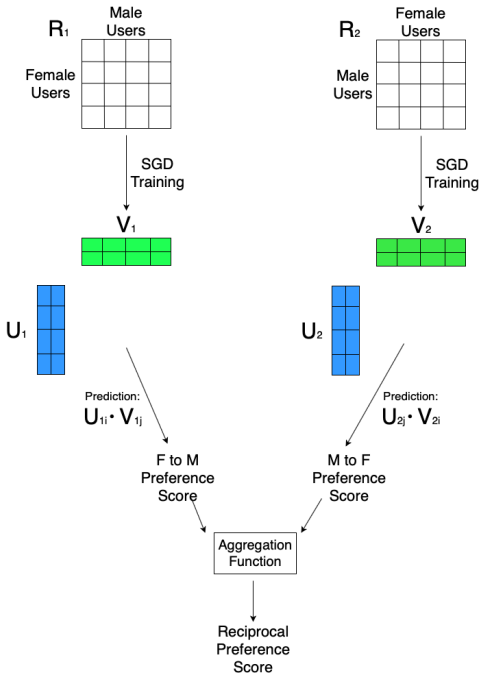


Figure 3.2: LFRR Visualisation

The datapoint order is randomised, and this process is repeated for a number of epochs until the feature vectors are stable. They can then be used to make predictions about how likely user a is to like user b by the dot product of their feature vectors.

Given matrices for a trained female-male preference latent factor model U_1 and V_1, and the same matrices vice-versa for male-female preferences U_2 and V_2, let the vector representing the row for a user a in matrix U_1 be denoted by $U_{1,a}$. Then the prediction algorithm for LFRR is described in Algorithm 3. (The other variables follow the same conventions as in Algorithm 1.)

As is evident from the algorithm description, computing a reciprocal score requires only a single dot product, and is therefore guaranteed to be linear time, regardless of the number of interactions from that particular user.

3.4.2 Evaluation

In this section, the dataset and the experiments performed and the metrics used to evaluate them are described. The results of these experiments are displayed alongside a comparison of the evaluations of RCF and LFRR with four different aggregation functions.

3.4.2.1 Experimental Setup

The data for the evaluation was provided by a popular online dating service, surpassing 10M users at the time of writing. As described in Section **1**, users express preference for each other by sending a *Like*. Users send a total of approximately 9 million *Likes* per week.

Algorithm 3 LFRR Predictions for two users a and b

INPUT: Trained feature matrices U_1, U_2, V_1, V_2

$Recs \leftarrow \emptyset$
for all $a, b \in Candidates$ **do**
  if a is female **then**
    $score_{a,b} \leftarrow U_{1,a} \cdot V_{2,b}^T$
    $score_{b,a} \leftarrow U_{2,b} \cdot V_{1,a}^T$
  else
    $score_{a,b} \leftarrow U_{2,a} \cdot V_{1,b}^T$
    $score_{b,a} \leftarrow U_{1,b} \cdot V_{2,a}^T$
  end if
  $reciprocalScore_{a,b} \leftarrow \mathbf{Agg}(score_{a,b}, score_{b,a})$
  $Recs \leftarrow Recs + (b, reciprocalScore_{a,b})$
end for
return $Recs$

Most datasets that have been used to test collaborative filtering-based reciprocal recommender systems in the past have been relatively small in comparison, comprising only a few thousand interactions at most. To the best of our knowledge, there is no example in the literature of a reciprocal recommender system being tested on a dataset of the size of the this dataset.

Data was sampled from varying time periods, from one day to 3 months of interactions. Previous marketing analysis of user data indicates that the vast majority of users either find a relationship or cease using the site during this time, and that long-term users often change their preferences. 3 months was therefore the longest period of time useful for generating recommendations for data in the online dating field.

There are also several other limitations on the data selected for the experiments, which are outlined below:

- Users who live in Tokyo and the surrounding areas. These users represent a significant majority of users.
- Users between 18 and 30 years of age, for the same reason as above. Users outside this age range are outliers in the user base.
- Users who have sent at least 10 *Likes*.

Limitation **3** is because both LFRR and RCF suffer from the *Cold Start Problem* [76]. There are currently no effective solutions to the Cold Start Problem for RCF. The data is therefore limited to users with enough data to make effective recommendations using both algorithms.

Further information about the dataset used and exclusions can be found in Appendix A.

3.4.2.2 Efficiency Evaluation

The standard evaluation metrics for effectiveness are used to evaluate LFRR. In this case, an efficiency evaluation was also conducted, and the metrics used to evaluate the efficiency of the algorithm are described here.

To the best of our knowledge, all RRSs in the literature work by calculating a reciprocal preference score between pairs of users (e.g. [113]). They generate recommendations by calculating reciprocal preference scores between a user a and every candidate user x, ordering users in descending order of score, and then taking the top N users as recommendations. There are therefore two important metrics to consider where efficiency is concerned:

1. The time taken to calculate a single reciprocal preference score between two users

2. The time taken to generate a list of N recommendations by calculating the reciprocal preference scores between a given user A and every other user in the dataset

These two metrics are considered separately, because it is easy to imagine situations where an RRS which could perform (1) efficiently might still be useful (for example, by displaying to the active user the likelihood that another user might be a good match), even in the absence of the ability to perform (2) in reasonable time.

A significant difference between RCF and LFRR is that LFRR requires training of the latent factor matrices U and V, whereas RCF does not rely on any pre-training. The training times for LFRR are also listed. These results are presented with the caveat that training times generally do not significantly affect the usefulness of the model unless training takes more than a few hours.

Further information about validation procedures and details of hyperparameters can be found in Appendix B.

3.4.2.3 Effectiveness Results

The ROC curves are based on one week of data. Figure 3.3 displays results from various aggregation functions used to aggregate results from the RCF algorithm. Figure 3.4 shows results from the same aggregation functions used on the LFRR algorithm.

In the case of the RCF algorithm, as is evident from the area under the ROC curve, the HM significantly outperforms the AM and GM. More interestingly, the cross-ratio uninorm outperforms the state-of-the-art RCF based on harmonic mean. This is consistent with the hypothesis that, because the HM and uninorm penalise situations where the two aggregation inputs differ strongly from each other, they are likely to perform better in the online dating domain, where two relatively high scores are more likely to result in a match than a very high and very low score.

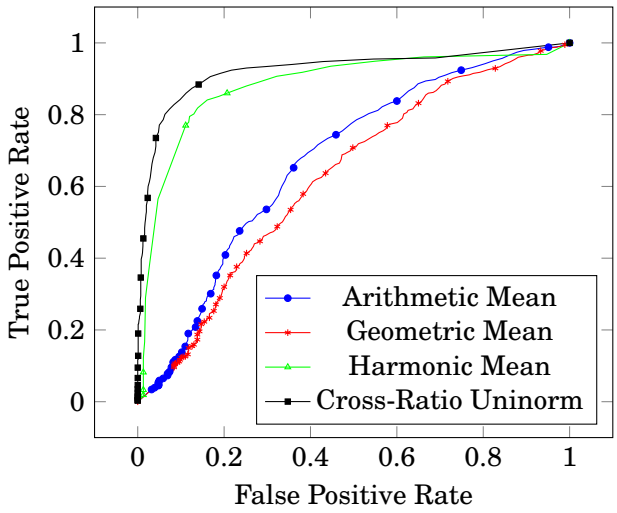


Figure 3.3: ROC curve obtained for each aggregation function considered in the RCF model.

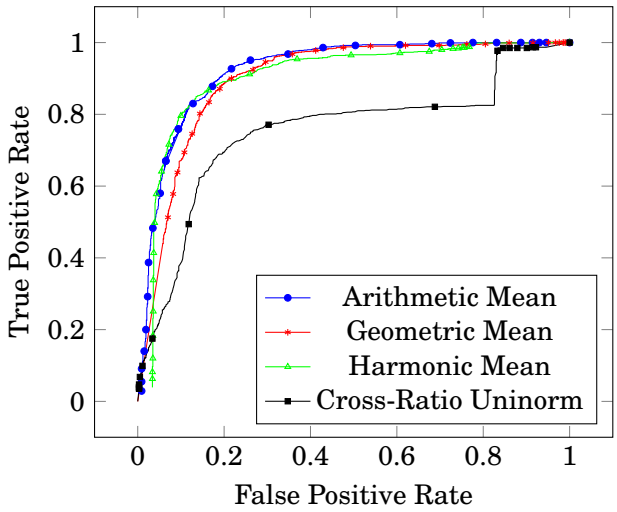


Figure 3.4: ROC curve obtained for each aggregation function considered in the LFRR model.

The difference between the performance of the aggregation functions is less evident on the LFRR model, where the AM, GM and HM are much closer together. The uninorm performance is significantly different. The shape of the cross-ratio uninorm on the LFRR algorithm can be explained by the mixed behaviour of uninorm functions. The threshold passing the neutral element of the aggregation function causes it to move from averaging behaviour to disjunctive behaviour, and can cause a sharp increase in the number datapoints classified as positive.

The best F1 scores with their corresponding precision and recall from each aggregation function are displayed in table 3.2 for both RCF and LFRR. Based on the best F1 score, the harmonic mean consistently gives high precision, which is ideal for recommender systems in online dating: users are more likely to trust a system that gives them a high proportion of very good matches. However, in the case of RCF, the cross-ratio uninorm gives both a higher precision

Algorithm	Precision	Recall	Best F1 Score
RCF (Arithmetic Mean)	0.58	0.86	0.69
RCF (Geometric Mean)	0.55	0.90	0.68
RCF (Harmonic Mean)	0.83	0.84	0.84
RCF (Uninorm)	0.84	0.90	0.87
LFRR (Arithmetic Mean)	0.81	0.92	0.87
LFRR (Geometric Mean)	0.83	0.86	0.85
LFRR (Harmonic Mean)	0.86	0.85	0.86
LFRR (Uninorm)	0.54	0.98	0.70

Table 3.2: Results based on best F1 score from each aggregation function tested applied to RCF and LFRR

and higher F1 score. The offline evaluation shows that compared to RCF, LFRR is more consistent across different aggregation functions. However, using the optimal aggregation function for each of algorithm (the cross-ratio uninorm in the case of RCF and the harmonic mean in the case of LFRR), there is no significant difference between the performance of the two algorithms based on offline evaluation. However, the evaluation does demonstrate a significant difference between the different aggregation functions for each algorithm. In the domain of online dating, depending on the algorithm used, the harmonic mean consistently gives the best performance. However, in situations where optimising recall is desirable (such as a social network where recommending a large number of possible friends might not diminish trust in the system), the arithmetic mean might be desirable.

3.4.2.4 Efficiency Results

A collaborative filtering-based reciprocal recommender system's running time has never been measured on dataset with a large number of users. Two aspects for both the RCF and LFRR algorithms were measured: the time taken to calculate a reciprocal preference score for a single pair of users, averaged over 100 pairs of users, and the time taken to generate a recommendation list by calculating the scores for all candidate users and ordering them in descending order. Because LFRR requires a training step, the training time on the same datasets is also displayed, with the caveat that a long training time doesn't have a significant impact on the algorithm's usefulness as compared to the time taken to generate recommendations. The size of the dataset is measured based on the number of interactions, which is the factor that has the highest impact on the time taken to generate recommendations and to train LFRR. To generate the datasets, an equal number of male and female users were sampled from the same dataset used to measure the effectiveness of the algorithms, with the restrictions detailed in section **4.1**.

For these tests, a *Google Cloud AI Platform* [1] server was provisioned. The identification code for the machine used is *n1-highcpu-16*. The server is currently listed as a legacy type, but at the

[1] https://cloud.google.com/ai-platform/docs/technical-overview

time of testing, this server was provisioned with 120GB of memory, 16 virtual CPUs and four NVIDIA Tesla K80 GPUs.

Size	RCF Score	RCF List	LRFF Score	LFRR List
10^3	0.003	1.75	1×10^-5	0.0001
10^4	0.005	13.7	1×10^-5	0.001
10^5	0.008	163	1×10^-5	0.025
10^6	0.09	> 1800	1×10^-5	0.63
10^7	0.5	> 1800	1×10^-5	2.0

Table 3.3: Time (seconds) to calculate a user-user score, and to generate recommendations, from a dataset of N interactions

Table 3.3 shows the time taken to generate recommendations on various dataset sizes. Where recommendations are generally expected by users in real time, times to generate recommendations of over 30 minutes (1800 seconds) are not practical, and hence recorded as >1800.

RCF struggles to generate recommendations in real time for even datasets containing very small numbers of interactions. On a dataset containing a million interactions (a realistic number for a very small dating service), the algorithm took over thirty minutes to produce a list of recommendations for a single user. However, the scaling of RCF is such that even a parallel implementation would not make the algorithm feasible for a large dating service with millions of interactions per week.

LRFF remains able to generate recommendations in real time with a serial implementation for datasets of up to ten million interactions. However, as the time taken to generate a single reciprocal preference score is constant, calculating a large number of reciprocal scores would be easy to parallelise, and generating recommendations in real time for datasets with hundreds of millions of interactions would therefore be feasible.

Size	LFRR Training Times
10^3	0.74
10^4	2.8
10^5	23.2
10^6	229
10^7	2332

Table 3.4: Training time in seconds for LRFF over 10 iterations of gradient descent, from a dataset of N interactions

The training times of the feature matrices for LFRR are displayed in Table 3.4. A dataset with a hundred million interactions (the monthly volume of an extremely popular service) takes a few hours to train.

3.5 Summary

In this chapter, a novel algorithm LFRR was designed, which applies latent factor models to reciprocal recommendation. It was evaluated against current baseline for reciprocal collaborative filtering, RCF. The effectiveness of LFRR is similar to RCF based on offline evaluations. Both algorithms were also tested on a larger dataset than has previously been used for reciprocal recommendation. LFRR has considerably better efficiency, and is therefore a more realistic algorithm for generating reciprocal recommendations in real time on a service with a large number of interactions.

Previous reciprocal recommender systems have used only the harmonic mean. This chapter presented an analysis of four functions for aggregating preference scores. Although the harmonic mean was the most consistent aggregation function across both RCF and LFRR, the cross-ratio uninorm function gave marginally better performance in the case of RCF. The choice of aggregation function therefore has a significant impact on the effectiveness of the model, and future research in this field should consider the testing of various aggregation strategies to find which one fits best with the algorithm and data in question.

LFRR was successful at applying latent factor models to reciprocal recommendation. However, more advanced modelling techniques are available, and models based on deep learning have been particularly successful in the field of user-item recommender systems [34]. These results might be improved upon in future by applying these models either to the current structure of two unidirectional models or to predicting matches directly.

CHAPTER 4

HYBRID FILTERING

Hybrid filtering describes any algorithm that makes recommendations by combining methods from different areas. These algorithms often aim either to improve the results of existing collaborative filtering algorithms by adding in content-based elements, or to address the weaknesses of collaborative filtering such as the cold-start problem by introducing a content-based or knowledge-based element to early recommendations. There are very few hybrid reciprocal recommender systems. In this chapter, a novel hybrid system is presented, and its efficacy is demonstrated on data from a popular social network.

=======

4.1 Introduction

Hybrid recommender systems describes the class of recommender systems that incorporates multiple algorithms from different fields of recommendation in order to improve on the results of any single algorithm. While the term can in theory describe any combination of two methods, it is most commonly used to refer to systems that have a content-based and a collaborative filtering component.

Prior to the work conducted for this thesis, there were no RRSs in the literature that combined collaborative and content-based results. This chapter describes a hybrid RRS designed for a social service for recipe sharing. This represents a novel contribution to the field of reciprocal recommendation.

The collaborative filtering part of this system is a latent factor model-based approach. The content-based system is a novel similarity metric based on Word2Vec that uses sentence embeddings of recipe titles to make recommendations based on similarity to previously preferred

recipes. The system is evaluated and it is demonstrated that the hybrid system outperforms its individual components.

4.2 Background

This section briefly reviews the literature relevant to hybrid RRSs and the systems described in this chapter. For a more in-depth literature review of all topics and papers surrounding this area, see Chapter 2.

4.2.1 Hybrid Filtering

Hybrid filtering in recommender systems describes algorithms that combine multiple different sub-types of recommender system in order to improve on the results of any individual system [29]. Often this means a combination of collaborative and content-based methods, although other less used subtypes such as knowledge based systems are sometimes included.

Burke defines terminology for hybrid systems based on the method of combining them. Different methods are generally used to overcome the weaknesses of individual systems [29] depending on the system. For example, a switching hybrid system might be used to change the result from a content-based to a collaborative filtering-based prediction depending on the estimated accuracy of each for a particular user [136].

There are a few examples of hybrid reciprocal recommender systems in the literature. The earliest of these was designed by Akehurst et al. [11], which used nearest neighbour-style collaborative filtering in combination with content-based attributes to make recommendations.

4.3 Hybrid Filtering for Social Networks

In *Reciprocal Recommender Systems* (RRSs), users are recommended to each other, therefore unlike classical RS where preference relations are unidirectional (user-to-item), in RRS preference relations among pairs of users need to be considered. RRSs are most often used in online dating websites [112] [150], where explicit indicators of positive and negative preference are gathered from users. However, they are noticeably having emergent applications in areas such as recruiting [132] and social networks [53].

Despite ongoing RRS primarily focus their application on online dating, different algorithms may be effective on different types of online services for connecting users. For instance, a fundamental characteristic in online dating websites is that they often have two distinct classes of user - male and female - whereas social websites such as *Twitter* and *Facebook* have only a single class of user where any one user can be recommended to any other. There is a clear shortfall of research on single-class RRSs as of now. Moreover, the rise of skill sharing and social platforms such as *Meetup.com*, in which contents published and shared among users play an

important role, raises the need for new or extended RRS models that accommodate user-user recommendations in these scenarios. Extant RRS research in general lags a long way behind traditional RS research, with broad areas such as hybrid RRSs still completely unexplored in a reciprocal context.

To address the aforesaid challenges, a novel Hybrid RRS (HRRS) is described, for recommending users to connect with each other socially in content/skill sharing platforms where: (i) unlike most online dating services, there exists a single class of users and (ii) both user-to-item and user-to-user preference information coexist. A model was developed that employs a CF-based RRS algorithm based on latent factor models recently proposed in [4], and combines it with classical RS principles relying on users' preferences towards content. For this, a novel method that exploits free text content information using word embeddings is also introduced, for calculating similarities between users predicated on their implicit preferences towards content. The results of using the proposed HRRS model are evaluated on *Cookpad*, a popular recipe sharing website in countries such as Japan, Taiwan and Indonesia.

The contributions of this model are fourfold:

1. The first hybrid RRS framework and model in the literature, combining reciprocal CF with principles from classical CF and CB. Importantly, many item-to-user RS services use hybrid techniques to produce better and more robust recommendations, but these techniques have not been yet explored in the field of reciprocal recommendation.

2. This system is also the first RRS model in the literature that operates on a single class of users, i.e. recommending users to each other within the same class, unlike e.g. opposite-sex online dating.

3. A novel similarity metric based on word embeddings modeled after free text information associated to content shared and/or liked by users.

4. A preliminary offline evaluation that includes an experimental study on real data, with a real time implementation of the algorithm.

It has been demonstrated that standard RS metrics such as precision and recall based on cross-validation are not always representative of the real effectiveness of RRS.

4.3.1 Hybrid Single-Class Reciprocal Recommendation

This section firstly introduces a novel HRRS framework. It then describes the proposed HRRS algorithm, characterised by:

1. Incorporating principles from classical item recommendation in the process of recommending users to each other. Despite the numerous hybrid CF-CB models devised in the literature

for *item-to-user* recommendation, this is the first hybrid model of its kind in a *user-to-user* setting.

2. Calculating reciprocal preferences among pairs of users who belong to the same class. This is a notable difference from most existing RRS algorithms that rely on predicting matching scores on pairs of users belonging to two classes, e.g. male and female in an online dating domain.

A novel framework for Hybrid RRS or HRRS, where both inter-user preferences and user-to-content preferences coexist, is formulated as follows:

- There exist a set of users U in the system, with $a, b \in U$ denoting any two users. A framework was designed where, unlike opposite sex online dating, all users belong to the same class and therefore any two users in U can be potentially recommended to each other.
- There is a set of content items X. In skills sharing platforms, items $r_a \in X$ are associated to a user a who published them, hence preferences towards such content can be taken as an indicator of potential preference from one user to another.
- There exist indicators of user-user preference $Pref(a \rightarrow b)$, e.g. likes or follows towards users.
- There exist indicators of user-item preference $Pref(a \rightarrow r_b)$, e.g. liked on content posted by other users.
- The HRRS recommendation problem consists in recommending users b to a target user a taking both types of preference indicators, $Pref(a \rightarrow b)$ and $Pref(a \rightarrow r_b)$, into consideration.

Figure 4.1 shows the general pipeline followed by the model to predict the level of matching between two users a and b, consisting of three main stages: (i) item-to-user based matching or *non-reciprocal* matching, (ii) reciprocal CF matching, and (iii) aggregation of item-to-user and reciprocal predicted matching scores. This model also introduces in (i) a novel extension of the Jaccard index formula to calculate pairwise similarities between users' preferences on content shared among users, predicated on word embeddings associated with content descriptions.

4.3.2 Item-to-User (Non-reciprocal) Matching

This component of the model considers users' preferences towards content published by other users and then pairwise user similarities are calculated based on content liked by both users. Without loss of generality, in the application domain of the Web skill-sharing platform considered in this study, the basic unit of *content* posted by a user $a \in U$ is a recipe $r_a \in R$, Therefore, the aim is to assign a higher matching to pairs of users whose preferences on content items, i.e. recipes,

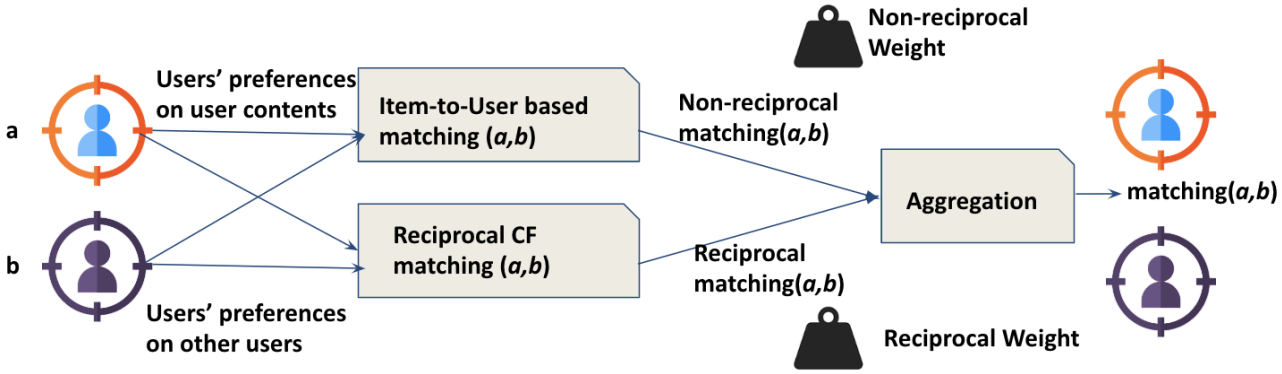


Figure 4.1: General scheme of the HRRS Model

are similar. Preference-based similarities among users are known to be challenging to calculate in domains where only implicit rating information (e.g. liked or seen items) are available. A classical solution for this is to identify recipes commonly liked by two users a and b, and use the Jaccard Index to calculate their similarity:

$$\frac{R_a \cap R_b}{R_a \cup R_b}$$

with $R_a, R_b \subset R$ the subsets of recipe items liked by a and b, respectively. This presents however an important limitation in skillsharing platforms where many instances of content published by different users can be highly similar to each other, because the Jaccard index only detects co-occurrences of *same* items in any two users' preferences. Consider for instance that user a liked four types of risotto recipes, and user b liked another four risotto recipes different from those liked by a. If R_a and R_b contain only these risotto recipes for each user, their Jaccard similarity would be zero (no risotto recipes in common). Users who have liked similar recipes but none of the same recipes would be identified as being dissimilar to each other, which is undesirable in a content-based system. A modified form of Jaccard similarity is used to account for similarity between non-identical content items that users may have liked. A non-identical content similarity measure is integrated into a user-user similarity metric in terms of their preferences towards content.

In order to quantify recipes and thus calculate their similarity, Word2Vec is used [89]. Word2Vec is a series of models trained on shallow neural networks that produce word embeddings such that words with similar meanings are represented by vectors with a short Euclidean distance between them. The training is based on proximity between words in large document corpuses such as Wikipedia [120]. Many pre-trained word embeddings are available, and for this project the *Google News Vectors*[1] were used, which contain 300-dimensional vectors representing a dictionary of 3 billion words, and have been tried and tested in a number of previous projects

[1] https://code.google.com/archive/p/word2vec/

[83]. The Jaccard Index is then modified as follows. An adjustment term is introduced, to smooth the (typically pessimistic) similarity degree obtained by the classic Jaccard Index.

$$\frac{|R_a \cap R_b| + \lambda}{|R_a \cup R_b| + \mu} \tag{4.1}$$

where,

$$\lambda = \sum_{r_a \in R_a - R_b} \sum_{r_b \in R_b - R_a} \delta(r_a, r_b) \tag{4.2}$$

$$\mu = |R_a - R_b| \cdot |R_b - R_a| \tag{4.3}$$

and λ is a sum over soft (non-strict) similarities $\delta(r_a, r_b)$ between non-identical pairs of recipes r_a, r_b found in R_a or R_b, respectively, but not in both. μ is the total number of such recipe pairs. Let $|r_a|$ be the number of vector representations of words associated to recipe r_a (obtained for instance by applying Word2Vec). Then, by looking at pairs of word vectors in r_a and r_b, a similarity degree is calculated between these two recipes as follows:

$$\delta(r_a, r_b) = \sum_{l=1}^{|r_a|} \sum_{k=l+1}^{|r_b|} sim(w_l, w_k) \tag{4.4}$$

assuming we chose r_a and r_b such that $|r_a| \leq |r_b|$. Here, $w_l \in r_a$ and $w_k \in r_b$ are vector representations of words present in both recipes, e.g. ingredients in common, and sim is a vector similarity metric between both vectors.

4.3.2.1 Reciprocal Matching

RRS approaches normally rely on indicators of preference between users. In the case of the Cookpad skillsharing platform, preference indicator data consist of *Follows*. Users can follow other users in order to be notified of their new recipe content whenever they post it. The data also contains a number of indirect preference indicators, such as *Bookmarks* (users can bookmark others' recipes to find them easily in future) and *Cooksnaps* (when a user makes another user's recipe, they can post their results along with a short review). *Follows* $F(a,b)$ and *Bookmarks* $B(a,b)$ were used to construct a unidirectional preference score that represents a user a's preference for a user b, $P(a,b)$, as follows:

$$P(a,b) = F(a,b) + \sum B(a,b) \tag{4.5}$$

With this, we construct a two-dimensional square preference matrix representing the preference of every user for every other user, and use this as the core of the reciprocal matching part.

The reciprocal matching process relies on two indicators of preference associated with each user a:

- Followed users by a.
- Users $b \neq a$ whose associated content has been liked by a.

Let $U_{m\times N}$ and $V_{m\times N}$ be two matrices. Each row in U, denoted $u_a = (u_{a1}\ u_{a2}\ \ldots\ u_{aN})$ contains N preference latent factors associated with user a, which represent what a likes. Each row in V, denoted $v_a = (v_{a1}\ v_{a2}\ \ldots\ v_{aN})$ contains N attribute latent factors associated with user a, which represents the properties of a. Both matrices have been obtained by applying a Stochastic Gradient Descent (SGD) algorithm to reduce the dimensionality of the original $|U| \times |U|$ matrices built upon the aforesaid preference indicators. Calculating the preference or affinity level from user a towards b, boils down to computing the similarity between a's preferences and b's attributes. Concretely, a unidirectional preference score from a to b is determined by applying the vector product, $p(a \rightarrow b) = u_a \cdot v_b^T$. Conversely, the preference score from b to a is similarly determined as $p(b \rightarrow a) = u_b \cdot v_a^T$. Both unidiretional preference scores are combined into a reciprocal preference score or matching score $m_{CF}(a,b) \in [0,1]$ using the harmonic mean operator [113, 150]:

$$m(a,b) = \frac{2\ p(a\rightarrow b)\ p(b\rightarrow a)}{p(a\rightarrow b) + p(b\rightarrow a)} \tag{4.6}$$

The harmonic mean operator has been typically used in previous user-to-user recommendation approaches to calculate reciprocal preference scores (as described in Chapter 3), due to its tendency to generate a lower aggregation output when none of the inputs are high enough, compared to other classical mean operators. This is convenient in reciprocal recommendation domains where a match between two users should be identified only when both users have potential preference towards each other to some extent.

4.3.2.2 Aggregation of User Matchings

The outputs of the non-reciprocal and reciprocal matching processes are finally aggregated into an overall matching between a pair of users a and b. Without loss of generality, in this work a weighted average is used for this aggregation (within the scope of the experiments equal weights are considered). Notwithstanding, a recent study on the effect of using other averaging and mixed behavior aggregation operators in RRS can be accessed in [94].

4.3.3 Results and Discussion

The two components of the HRRS model were evaluated individually, and then evaluated the hybrid model. For detailed discussions of the data used and the validation procedures, see Appendices A and B respectively. The ROC curve for each of the three models is shown in Figure 4.2. The neutral 0-1 line is also given on the figure as a dashed black line for reference. As there are no other examples of RRSs used in the context of a recipe sharing service, or on a single class

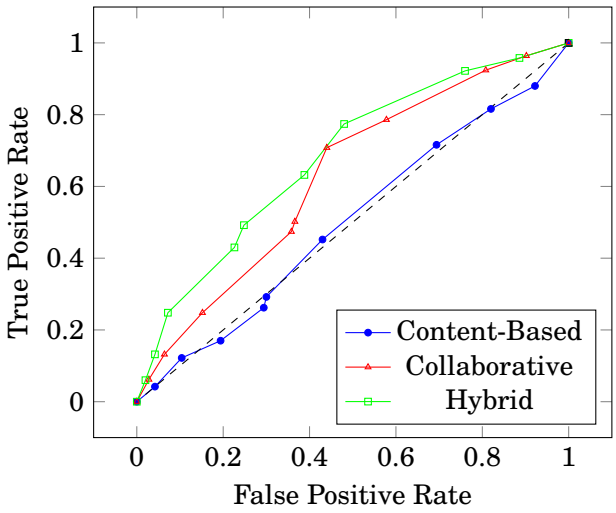


Figure 4.2: ROC curve obtained for the content-based, collaborative and hybrid models.

of users, it was not appropriate to evaluate the system against other current RRS, which were designed for online dating and not suitable for the aforementioned reasons.

The non-reciprocal algorithm where similarities among users are calculated upon user-content preference information, performed relatively poorly by itself, only slightly better than random filtering for some threshold setting. This was largely a result of a large number of false positives which in turn leads to a lower precision and F1 score. As intuitively, users with similar terms in their liked and created recipes would be more likely to like each other, this is attributed to the offline testing procedure. As discussed, the data provided us with no negative preference indicators among users, and so users who had not shown preference indicators towards each other were used as as the negative test.

In contrast, the reciprocal algorithm where both directions of preference between users are analysed, produced good results, with a positive ROC curve. The improvement seen in the hybrid model that incorporates the (poorly performing by itself) non-reciprocal algorithm, in spite of its neutral ROC curve, is not surprising. The reciprocal matching algorithm's positive results are able to be further refined by the non-reciprocal counterpart, with significantly dissimilar users in terms of their created content being filtered out by the reciprocal process.

For reference, the F1 scores for each of the model versions is displayed Table 4.1. The highest F1 score occurred at a threshold of 0.2. However, as is evident from the ROC curve, the precision of the system is significantly higher at thresholds closer to 0.8. As RS approaches normally value precision more highly than recall (users are more likely to trust a system with a few positive recommendations), higher thresholds may be more beneficial to a live system.

Based on the experimental study conducted, the precision and recall of the proposed hybrid approach is high enough to convince us that it will produce good recommendations for end users, and that the hybrid system is more useful than either the non-reciprocal or the purely reciprocal

Threshold	Rec. F1 Score	NonRec. F1 Score	HRRS F1 Score
0	**0.666**	**0.666**	**0.666**
0.1	0.672	0.628	**0.673**
0.2	0.676	0.619	**0.687**
0.3	0.664	0.594	**0.686**
0.4	**0.659**	0.480	0.625
0.5	0.537	0.366	**0.565**
0.6	0.517	0.336	**0.519**
0.7	0.354	0.249	**0.375**
0.8	0.220	0.199	**0.224**
0.9	**0.113**	0.077	0.111
1.0	0	0	0

Table 4.1: Results obtained by varying the threshold for bidirectional preference score-based recommendation

algorithm by itself. However, due to the lack of negative data for validation, and the complexity of the recommendation domain in general, this should be considered as a promising yet preliminary result only, and that further real time testing is required to determine the system's effectiveness.

4.4 Summary

This chapter described a hybrid reciprocal recommender system for a social networking service focused on skill sharing, and demonstrate its effectiveness on a representative dataset. The algorithm used latent factor model-based collaborative filtering, combined with a novel similarity metric for recipe titles based on word embeddings generated using Word2Vec.

The algorithm was evaluated through cross-validation, and it was shown that the hybrid system was more effective than either the item-to-user based or the purely reciprocal collaborative filtering systems by themselves. This work is novel in a number of ways, most significantly that it is the first hybrid reciprocal recommender system, and the first RRS to operate on a single class of users.

Although the system is able to produce recommendations for users, the effectiveness of HRRS is not as high as the systems developed for online dating services in Chapters 3 and 5. Part of this is due to the quantity of data available, and due to the lack of clear positive and negative preference indicators as exist in online dating. However, there are a number of methods by which the algorithms themselves might be improved in future works. More effective methods of collaborative filtering might be applied to the collaborative filtering portion. Similarly, research on transformers has shown the effectiveness of systems such as BERT [41] at producing meaningful embeddings for text which might improve the content-based recommender.

CHAPTER 5

CONTENT-BASED FILTERING

Content-based filtering, as outlined in earlier sections, establishes explicit or implicit preferences for users, and makes recommendations based on the properties of items. This chapter describes original contributions to collaborative filtering in reciprocal recommendation. In particular, the focus is on using image data for reciprocal recommendation. Machine learning techniques can be used to establish implicit preferences for images based on user preference history, and use this to make recommendations. In addition to being a powerful predictor for online dating, this technology has other potential applications such as for image-based social networks.

=======

5.1 Introduction

As described in Chapter 2, content-based recommender systems make use of user preferences for content to make recommendations. In user-item recommender systems, preference for content generally means content provided by the service itself: for instance, on a movie recommendation service, content might mean genre, specific actors, directors and so on. In reciprocal recommendation, where the targets are other users themselves, *content* implies information provided by the users themselves. In the context of online dating specifically, this content can commonly be divided into three types: categorical data (such as age, height or pets), free text data (generally a few paragraphs that the user writes about themselves) and photographs, usually of the user themselves.

This chapter describes the development of two algorithms that use photographs to predict mutual preference. The first, *ImRec*, is based on a Siamese network which determines whether, given

one previously preferred photograph as an anchor point, a user will like a different photograph. The second, *TIRR* incorporates an LSTM to interpret results from the same Siamese network, making predictions about the next preference based on a sequence of previous preferences.

ImRec, developed first and using a simpler structure, represents an advance of the field of content-based reciprocal recommendation. It also performs better than the state-of-the-art collaborative filtering solutions in cold start situations. *TIRR* performed better than any other RRS algorithms on a large dataset from a popular online dating service, and represents an advance of the field overall.

5.2 Background

This section briefly reviews the literature relevant to content-based reciprocal recommendation and to the systems subsequently described in this chapter. For a more in-depth literature review of all topics and papers surrounding this area, see Chapter 2.

5.2.1 Content-Based Reciprocal Recommender Systems

Content-based solutions are the most common form of RRSs in the literature. These originated with *RECON*, developed by Pizzato et al. [113]. RECON uses categorical data to make recommendations based on implicit user expressions of preferences. RECON showed promising results during offline testing on a dataset from an online dating service.

Since RECON's publication, a number of other content-based reciprocal recommender systems have been developed that improve on the results from RECON. For example, Alanzi and Bain [13] present a time-sensitive content-based RRS. This has also extended outside of online dating into other fields such as recruitment. Almalis et al. [14] developed a content-based RRS based on Minkowski distance for recruitment.

5.2.2 Machine Learning for Attractiveness

Prediction of attractiveness is a relatively new field in machine learning, with relatively little research. In general, modern machine learning and especially CNNs have been shown to be very adept at classifying images and extracting specific attributes [40]. A number of papers attempt to predict specific attributes such as age and height from photographs [61]. More recently, a small number of papers describe models that predict attractiveness in the absolute sense [153].

A more relevant concept for reciprocal recommendation is *personal attractiveness*, which is the prediction of how attractive one user is to another user. Models that predict personal attractiveness are much sparser in the literature. Two models based on using trained CNNs to predict attractiveness on online dating services demonstrate promising results [63, 121], although neither was adapted into a RRS.

5.3 Siamese Network-based Model for Image Preference

Content-based RRSs in the literature have only been designed to work with categorical data. RECON was trained on a service that did not include user image data. This type of service is in the minority - most online dating services use images, and some very popular ones such as *Tinder*[1] encourage users to make initial decisions based entirely on images. There is informal evidence that users on dating services overwhelmingly make decisions based on image data, even when detailed text profiles are available[2]. In addition, recent social networks such as *Instagram*[3] often focus on images rather than written or categorical content. As such, attractiveness of users, generalised to attractiveness of images to individuals, could be used to improve social RRSs. As attractiveness is subjective, this measure should account for the tastes of individual users, and make recommendations based on specifically who is attractive to whom. The phrase "personal attractiveness" is used to refer to the attractiveness of one person's image to another person. The whole image is potentially a trigger for attraction, and not only the person in the image's physical appearance.

To overcome the limitations of content-based RRSs that use only categorical data, an original recommender system, *ImRec* was developed, which predicts preference of users x and y for each other given their images and their history of positive and negative preferences. ImRec is based on a Siamese Neural Network that predicts, given an image of a user already liked by x and an image of y, whether x will like y. To the best of our knowledge, this is the first example in the literature of a machine learning model successfully predicting personal attractiveness. The results from this model are aggregated across the history of x and y's preferences, to give two unidirectional preference scores. These two scores are then aggregated into a single bidirectional preference relation. Offline testing demonstrates that the model is capable of successfully differentiating between positive and negative indicators of preference in this context, where the baseline algorithm RECON was not able to.

This research was produced in collaboration with a popular Japanese online dating service. Images on this service are mostly photographs, and are all manually checked to ensure that the user and no other people are present in the photograph. The dataset is therefore of relatively high quality.

5.3.1 Methodology

In this Chapter, a Siamese CNN is described [71] as a method of estimating a user's x's preference for a user y's image, based on x's history of positive and negative preferences for images.

[1] https://tinder.com/
[2] https://www.gwern.net/docs/psychology/okcupid/weexperimentonhumanbeings.html
[3] https://www.instagram.com/

5.3.1.1 Problem Formulation

The online dating service currently only supports heterosexual relationships. It can therefore be assumed that for a set of users of one gender $X = \{x_1, x_2, \ldots, x_{|X|}\}$ there is a set of candidate users for recommendation $Y = \{y_1, y_2, \ldots, y_{|Y|}\}$. A user may have an ordered history of preference expressions for users of length n, for example, $Sx = \{Sx_{t_0}^{y_i}, Sx_{t_1}^{y_j}, Sx_{t_m}^{y_k}, \ldots, Sx_{t_n}^{y_l}\}$ where $Sx_{t_m}^{y_i} \in Y$ represents the expression of positive or negative preference of user x for the user y_i at time t_m.

In our reciprocal system, our objective is to estimate $R^{x,y}$, the reciprocal preference score that represents the projected degree of preference of two users for each other. $R^{x,y}$ is a function of the historical preferences of x and y as well as the two users themselves, and train a model to predict it using all of this information:

$$R^{x,y} = f(S^x, S^y, x, y; \theta) \tag{5.1}$$

Where θ represents the parameters of the model. Note that contrary to most previous approaches to RRSs, our approach trains a single model to predict reciprocal preference using all of the information, as opposed to combining the results of two models predicting unidirectional preference. Also note that the reciprocal preference is symmetrical i.e. $R^{x,y} = R^{y,x}$.

5.3.1.2 Network Structure

Siamese networks have been successful in various areas such as object recognition [143] and tracking [25]. They are particularly apt at solving classification problems where the system is required to adapt to new examples quickly, known as *one-shot learning*. In the case of a classification problem, the network is trained with triplets broken into alternating pairs. The first image in the triplet, y_a is the anchor. The positive y_p is from the same classification group as the anchor, while the negative y_n is from a different group. The labels are either 1 or 0 depending on whether the inputs are (y_a, y_p) or (y_a, y_n) respectively.

The network structure is visualised in Figure 5.1. The symmetrical CNNs reduce the images to a 128-dimensional vector. Note that the CNN trained to create the embedding is not visualised for space concerns, but is represented in Table 5.1 instead. The final part of the network is then trained on the difference between the embeddings. The network is trained using a loss function that attempts to minimise the difference between the two images if they are y_a and y_p, and maximise the difference if they are y_a and y_n. This generalises the network to differentiate between images of different classes - in this case, to differentiate between an image which a user x would *Like* and an image which a user x would *Nope*. This subsequently allows us to make predictions about preference.

The specific structure of the network is described in Table 5.1. Because the face is likely to be an important part of a user's positive or negative reaction to other users, a number of layers

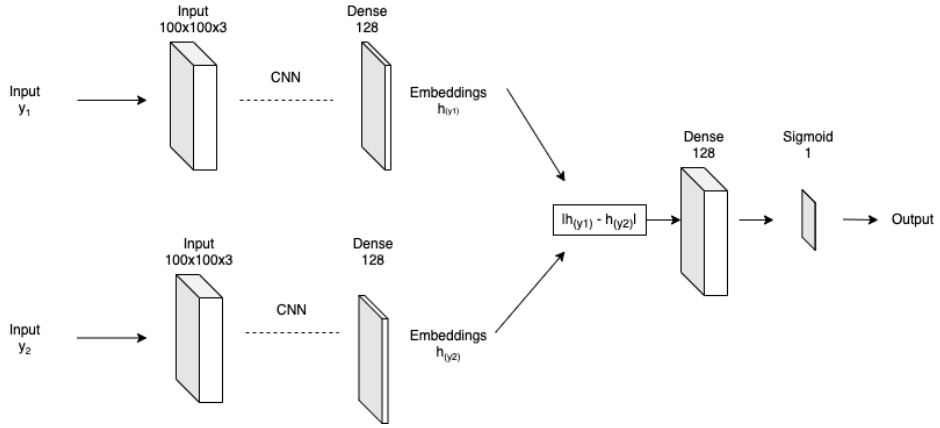


Figure 5.1: Siamese network visualisation. Refer to Table 5.1 for the CNN architecture details.

Layer	**Size-in**	**Size-out**	**Kernel**	**Param**
input		100x100x3		0
conv1	100x100x3	100x100x3	7x7x3	444
maxpooling1	100x100x3	34x34x3	3x3	
normalization1	34x34x3	34x34x3		12
conv2	34x34x3	34x34x64	3x3x64	1792
maxpooling2	12x12x64	12x12x64	3x3	
normalization2	12x12x64	12x12x64		256
conv3	34x34x3	12x12x192	2x2x192	49344
maxpooling3	12x12x64	4x4x192	3x3	
conv4	4x4x192	4x4x384	2x2x384	295296
maxpooling4	4x4x384	2x2x384	3x3	
conv5	2x2x384	2x2x256	1x1x256	98560
conv6	2x2x256	2x2x256	3x3x256	590080
maxpooling5	2x2x256	1x1x256	3x3	
flatten	1x1x256	256		
dense1	256	256		65792
dense2	256	128		32896

Table 5.1: The structure of the CNN used as the symmetrical part of the network to create embeddings

with small convolution kernels were used, which has been demonstrated to be effective in face recognition and evaluation settings.

The network learns based on the difference between the outputs of the two symmetrical parts of the network via a shared weight parameter W. We use W to map y_1 and y_2 to h_{y1} and h_{y2}, which are two points in a 128 dimensional space. We can then calculate the distance between these two lower dimensional points as follows:

$$D_W(y_1, y_2) = |h_{y1} - h_{y2}| \tag{5.2}$$

Siamese networks are often trained with *Contrastive Loss*. The Contrastive Loss function, uses a *margin m*, and depending on the size of the margin, results in a high loss when the the network's prediction is wrong about two similar images.The Contrastive Loss function is defined as:

$$L(y_1, y_2) = (1 - Y)\frac{1}{2}(D_W(y_1, y_2))^2 + Y\frac{1}{2}(max(0, m - D_W(y_1, y_2))^2 \tag{5.3}$$

where Y is the binary indicator representing Like and Nope, $D_W(y_1, y_2)$ is the embedded distance between two images and m is the margin.

In many situations where siamese networks are used, the objective is to distinguish between distinct classes of items, and in this case a high error for similar objects in different classes is appropriate. In the case of preferences, this is not necessarily appropriate, as preferences are not necessarily categorical. Binary Cross-Entropy, defined in Equation 5.4, which does not punish incorrect classifications of similar images, was a more effective loss function.

$$L(y_1, y_2) = -(Y \log(g(D_W(y_1, y_2))) + (1 - Y)\log(1 - g(D_W(y_1, y_2)))) \tag{5.4}$$

where Y is the binary indicator representing Like and Nope, g is a Multi Layer Perceptron and $g(D_W(y_1, y_2))$ is the predicted probability of $D_W(y_1, y_2)$ resulting in a Like.

5.3.2 Recommendation Algorithm

In this section, the operation of the ImRec algorithm is described, incorporating the model described in Section 3.2. The algorithm is visualised in Figure 5.2. Given two users x and y, the algorithm has four steps to calculate a bidirectional preference relation that represents the likelihood the two users will like each other.

In Step 1, the users previously *Liked* by users x and y are identified, and those users' main images are extracted. The number of images used for each user was capped to the 30 most recent. This decision was made to maintain relevance.

In Step 2, the images from Step 1, in addition to the inputs of the candidate user, are used as inputs to the appropriate siamese network. For instance, the case that x is a male user, y's image is used as the anchor (y_a), and the images x has *Liked* (y_p) are used as the positive samples while the images that x has *Noped* (y_n) are used as negative samples for the model trained on male preferences for female user images. The output is a list of scores, one for each comparison between x's image and each (y_a, y_p) pair.

The output from Step 2 is a list of scores, and Step 3 aggregates these scores into a single score. In order to do this, the scores are separated into 5 bins of equal size between 0.0 and 1.0, and convert the bins to a distribution. This distribution is the input to a random forest. The regressor was trained on a training set of 10000 samples independent of the training data for the

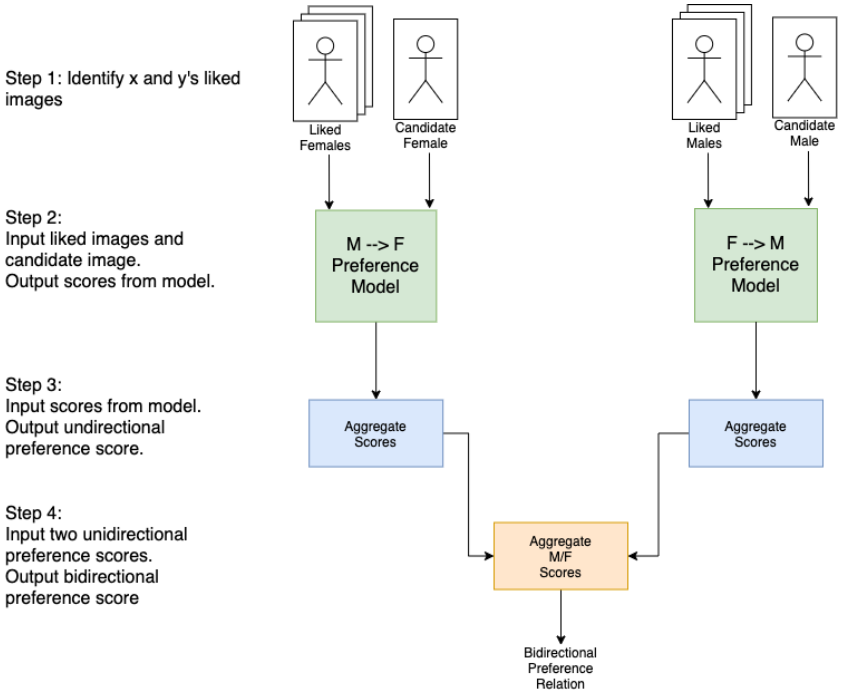


Figure 5.2: ImRec visualisation

Siamese network. This slightly outperformed simpler methods such as the Pythagorean means, and there was no difference between the random forest and a neural network.

In Step 4, the two unidirectional preference scores (representing x's preference for y and y's preference for x) are aggregated into a single bidirectional preference score using the harmonic mean. Our decision here is motivated by research indicating that the harmonic mean performs well in RRS contexts [94], and also because of our desire to keep our research as consistent as possible with our baseline, RECON, which also uses the harmonic mean.

The methods in this section are tailored to matching female and male users because of the data available to us and because the algorithm is easier to visualise and explain with two distinct classes of users. However, the algorithm could easily be adapted to users of any orientation by creating a personalised preference model for each user with their *Liked* and *Candidate* users, including only those for whom reciprocal interest is possible based on their own orientation.

5.3.3 Evaluation

This section describes the evaluation and results of ImRec. For details on the dataset used, see Appendix A. For details on hyperparameters and validation procedures, see Appendix B.

5.3.3.1 Image Preference Model Results

In this section, the results of the image preference prediction model are described. This is the siamese network described in Section 3.2 that represents Step 2 in Figure 5.2. To the best of our knowledge, this model is the first of its kind: there are no other models that attempt to predict personal attractiveness based on images. Because of this, the results for this model are presented without a point of comparison.

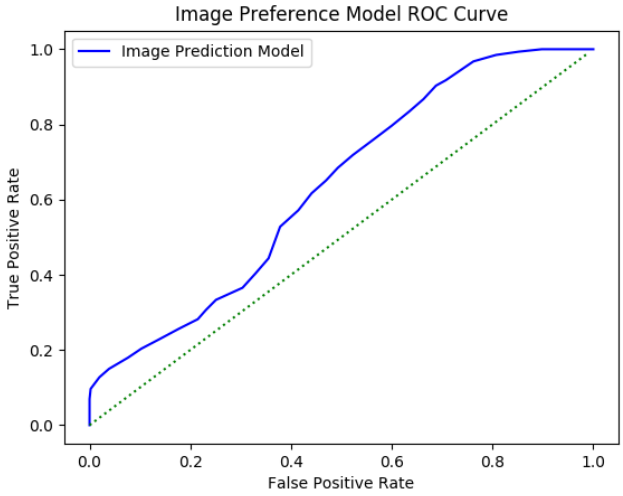


Figure 5.3: ROC Curve for siamese network to predict image preferences.

The ROC curve, based on a test set of 20000 interactions from users not in the original dataset, shows that the model is capable of successfully predicting user preference based on a single image. Although the model is not always accurate in this prediction, the fact that users often *Like* a relatively large number of other users means that this model can be used as a base for predictions based on results from the model over a large number of interactions.

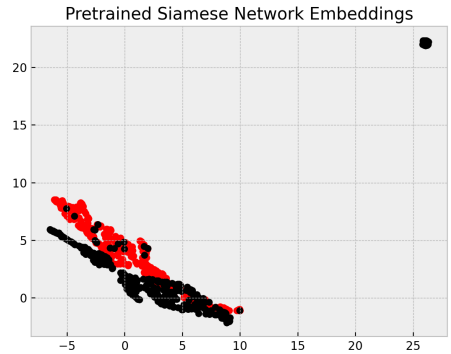


Figure 5.4: Pretrained Siamese Network Embeddings

The output of the Siamese network is a 128-dimensional vector, which forms the input of the RNN. It is therefore useful to visualise these embeddings. In order to do this, we use *Uniform Manifold Approximation and Projection for Dimensionality Reduction* (UMAP) [87] to reduce the 128-dimensional vectors to two-dimensional vectors for visualisation. This visualisation is displayed in Figure 5.4.

In this visualisation, the black datapoints represent *Noped* images and the red datapoints represent *Liked* images. It is clear from the visualisation that the embeddings are separable to some extent. The anomalous black cluster in the top right of the image represents heavily distorted or very poor quality images, or images misclassified by the face detection algorithm (i.e. images that do not contain a face). These tend to be almost universally *Noped*.

5.3.3.2 Results for Content-Based Algorithms

In this subsection, we present the results for ImRec compared to the current state-of-the-art content-based RRS, RECON.

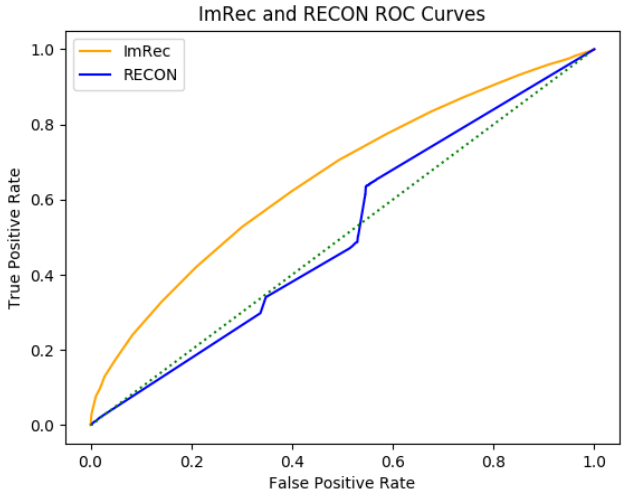


Figure 5.5: ImRec and RECON ROC curves.

Figure 5.5 shows the ROC curve for ImRec versus our baseline of RECON. The reference line is displayed as a dotted line. The graph was drawn using 1000 different thresholds between 0.0 and 1.0. ImRec generally has a positive and predictable curve, indicating that it is correctly predicting indicators of preference based on the users' images. On this dataset, our baseline RECON performed poorly, often worse than the reference.

The main reason for RECON's poor performance on this dataset in spite of a good performance on its original test dataset is likely to be the lack of images in the dataset it was tested on. Pizzato et al. state that RECON was designed for a dataset where, "The profile of a user is made of two components: free text information and a pre-defined list of attributes, ..." [113]. In contrast, many

modern online dating services and social networks use images very prominently, and users are often given an opportunity to make a positive or negative decision about another user based on an image and no other information. In this situation, ImRec provides a clear advantage.

Algorithm	**Precision**	**Recall**	**Best F1 Score**	**AUC**
ImRec	0.59	0.91	0.71	0.65
RECON	0.59	0.64	0.61	0.51

Table 5.2: Results based on best F1 score for all relevant algorithms.

Table 5.2 shows the best F1 scores for the relevant algorithms, which was found by varying the threshold. In this case, ImRec performs about 0.1 better than RECON. However, as is evident from their respective ROC curves, it is much easier to improve precision in the case of ImRec by increasing the threshold, whereas RECON performs much worse under these conditions on our dataset. Precision is vital for trust in recommender systems, as users who are shown a large proportion of recommendations that are not relevant to their interests are less likely to continue using the system.

5.3.3.3 Cold-Start Results

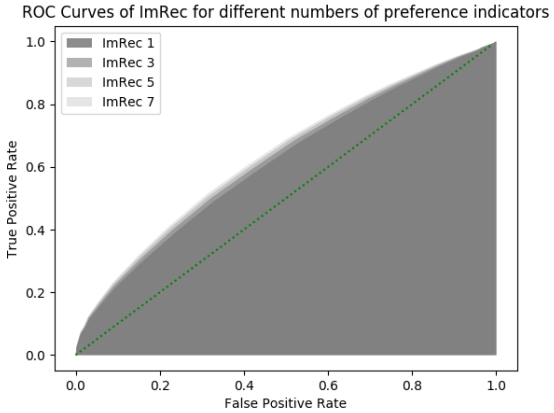
In this subsection, we present the results for ImRec in cold-start situations against the current state-of-the-art RRS, LFRR.

We hypothesised that ImRec would perform better than collaborative filtering algorithms in cold-start situations. We tested ImRec against the current best in class collaborative filtering algorithm, LFRR [96]. Using all available data, LFRR outperforms ImRec. However, with very little data, correlations between user preferences provide less useful information about the user's preferences than the information in the content-based model.

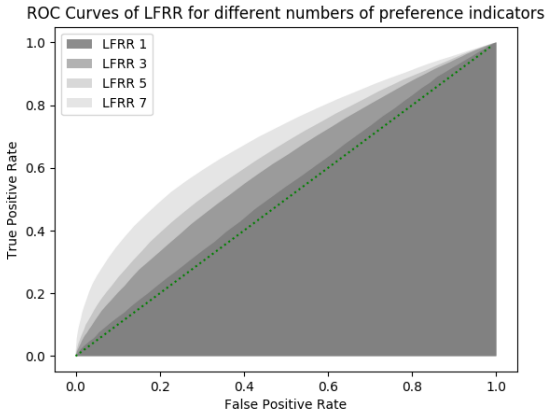
We tested ImRec and LFRR on a set of 20000 users interactions (10000 *Matches* and 10000 *Nopes*). From these users, we restricted interaction data available to the algorithm in training to a fixed number of interactions in order to simulate a new user. We make the assumption that new users *Like* and *Nope* in equal quantities, which in general is true. We name the algorithms trained on restricted data *Algorithm K* where K is the number of *Likes* and *Nopes* from the users in the test set available in the training set. For example *LFRR 1* is the LFRR algorithm tested on a set of users whose training data consisted of one *Like* and one *Nope*; *ImRec 3* is the ImRec algorithm tested on a set of users whose training data consisted of three *Likes* and three *Nopes*.

Figure 5.6 shows the ROC curves for the ImRec and LFRR algorithms trained with restricted data. The LFRR curve is very close to random choice when trained with only one expression of preference, and improves quickly with more data. On the other hand, ImRec produces significantly better results with very little data, and improves more slowly as more examples become available.

Table 5.3 shows the AUC for each of the algorithms trained with restricted data. It is clear from this that with fewer than 5 positive and negative indicators of preference available, ImRec outperforms LFRR, and the converse is true at 5 or more. The first day of a user's interactions



(a) ImRec curves.



(b) LFRR curves.

Figure 5.6: Curves for ImRec and LFRR for cold-start situations for various numbers of preference indicators.

Algorithm	**1 Indicator**	**3 Indicators**	**5 Indicators**	**7 Indicators**
ImRec	0.613	0.625	0.633	0.639
LFRR	0.530	0.604	0.639	0.696

Table 5.3: AUC for ImRec and LFRR for different preference indicators.

on a dating service is often essential, with the user deciding whether to commit to the service long term or give up based on their personal experience. As such, being able to make effective recommendations at an early stage is extremely useful for an RRS.

Based on these results, there are a number of ways that ImRec could be used to improve on the current best in class as part of a hybrid system. However, even the most simple method: a switching hybrid system that uses ImRec for recommendations up to 5 positive and negative interactions, is a clear and significant improvement.

5.4 Recurrent Neural Network-based Model for Image Preference

In this section, a novel recommender system is described, *Temporal Image-Based Reciprocal Recommender* (TIRR), that uses a Recurrent Neural Network (RNN) to interpret a user's history of preferences for images, and make predictions about their future preferences in order to make recommendations. This is a significant improvement on the only image-based RRS, *ImRec*[92] described in the previous section, in the sense that it outperforms both ImRec (previously the state of the art in content-based reciprocal recommendation) and also the current state of the art collaborative filtering solutions.

In addition to the advantages in terms of its improvement in the ROC curve on cross-validation, TIRR is also an advance of the field in the sense that it provides a unified system that predicts matches directly, as opposed to two separate predictions of unidirectional preferences followed by an aggregation. There is some doubt as how to combine two unidirectional scores into a single bidirectional score in a way that is fully representative of two users' bidirectional preference for each other; TIRR solves this by predicting the bidirectional relation end to end.

The system was tested using a popular online dating service. We used 200000 users and approximately 800000 expressions of preference combined split across train and test sets.

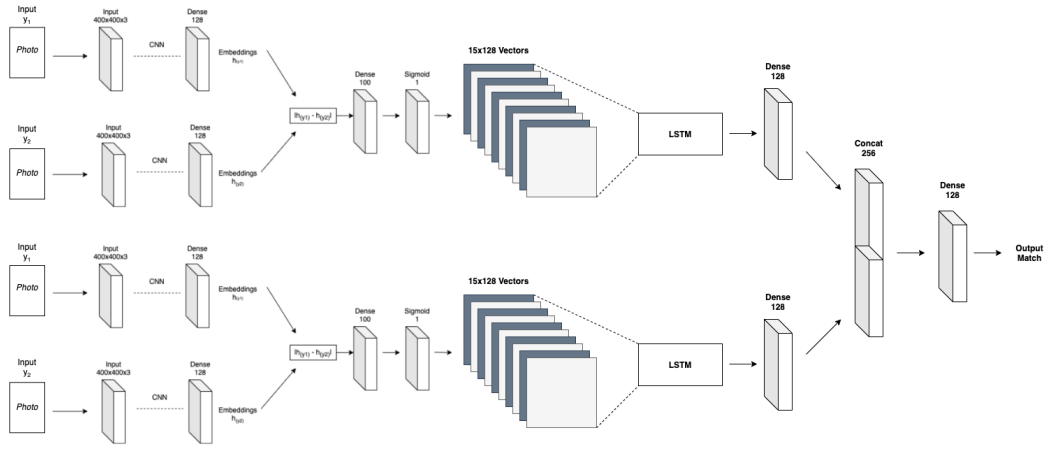


Figure 5.7: TIRR: the architecture to predict matches using an LSTM to interpret historical preference data on user photographs.

The Siamese network described above, when trained on unidirectional preference, is an effective model. In this section, we describe the RNN we use to interpret the user history based on the results of the Siamese network.

The output of the Siamese network is a point in 128-dimensional space that represents the preference of a user x for an image y_k based on comparison with the anchor image y_a. Based on initial experimental work, we chose an LSTM-based RNN architecture to interpret the time

series of images. The *forget gate* of the LSTM is particularly intuitive in this case. For a state s_t at time t, a forget gate described by f_t, a write gate i_t and a candidate write $\tilde{s}_t$ derived from the input and the previous state, the next state is described by the equation:

$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t \tag{5.5}$$

We might intuitively expect that preferences expressed by users would change over time, and the forget behaviour of the LSTM allows us to model this, with the input for the state s_t of the LSTM modelling the preferences of user x being the user S_t^x, and the final input at $s_{|S^x|+1}$ being the user y whom we wish to estimate x's preference for.

The LSTM is visualised in Figure 5.7. Because users have variable length preference histories, we fill the histories of users with shorter histories with dummy images and use a masking layer to filter them. The LSTM and subsequent dense neural network form a representation in 256-dimensional space of the user's preference as a time series.

Layer	**Size-in**	**Size-out**	**Kernel**	**Param**
input		128x15		0
LSTM	128x15	128	128	128
dense1	128	256	1	128
concat	128x2	256	1	256
dense2	256	128	1	256
output	128	1	1	128

Table 5.4: The layers of TIRR following the mapping of images into 128-dimensional space by the pre-trained Siamese network

Specifically, the network consists of an input layer, which accepts a maximum of 15 outputs from Siamese networks in 256-dimensional space concatenated together. Experiments determined that more than this did not significantly alter the performance of the network. The layers are described in Table 5.4. If a user has fewer preferences expressed than this, the earlier images are filled with zeroes, and the network learns to interpret this as dummy data. Following the LSTM, the network consists of a single dense layer of 128 neurons, and then a dropout layer with a dropout rate of 0.4. The network was trained with an Adam optimiser with a learning rate of 0.0001.

5.4.1 Training and Match Prediction

This section describes training the network to predict matches between two users. As described in Section 5.3.1.1, our objective is to differentiate between interactions consisting of bidirectional expressions of preference, *Matches*, and unidirectional expressions of negative preference, *Dislikes*.

The full training process is visualised in Figure 5.8. Our experiments determined that the network trained extremely slowly when trained in its full form from an initial randomised state,

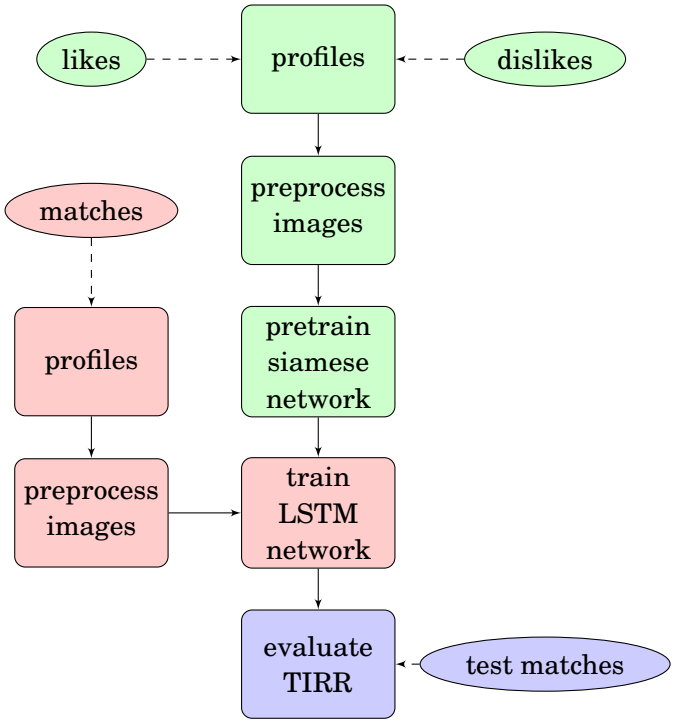


Figure 5.8: The process by which TIRR is trained. Three independent datasets used represented by different colours.

and we therefore pre-trained the Siamese network segment of the network using one dataset, shown in green. The subsequent training of the full system on matches was done using a separate dataset, shown in red. The final evaluation was done using a third dataset, shown in blue. In addition, Neve et al. demonstrated that the Siamese Network training was more effective when two networks were trained separately on male and female data [92]. As the service providing our data currently only supports heterosexual dating, this split does not decrease the usefulness of the application in this case.

Training for the Siamese networks were based on 500000 triplets (y_a, y_p, y_n) sampled from 200000 users split evenly over male and female images. Images were cropped and centered on the faces of users before training. Other methods of preprocessing such as affine transformations, which have been shown to improve the predictive power of other networks [81] did not have any impact on performance. The Siamese networks were trained to predict unidirectional preferences i.e. y_p was an image x had *Liked* (but not necessarily with reciprocity) and y_n was an image x had *Disliked*.

Following convergence of the Siamese network, the LSTM network was trained based on the preference histories of 100000 users to predict *Matches* and *Like-Dislike Tuples*. This dataset was separate from the dataset used to train the Siamese network. Histories were capped at one year, because of concerns that changes to the service's design and search algorithm over time might have an effect on user preferences. They were also capped to a maximum of 15 preferences,

because initial experiments showed that longer sequences did not improve accuracy, and because some outlier users express thousands of preferences, which results in an unreasonable increase in training and prediction times.

Finally, the LSTM was tested on a separate dataset of 20000 *Matches* and *Like-Dislike Tuples*. There was no overlap in preference expression between the three datasets. There was overlap between the users contained in these datasets, but as in a real-world situation the system would be trained based on users on the service and subsequently used to make predictions for those users in addition to new users, testing in this way is valid and representative.

The following subsections describe the results for TIRR. For details on the dataset used, see Appendix A. For details on hyperparameters and validation procedures, see Appendix B.

5.4.2 TIRR vs Content-Based Algorithms

As described in Section 1.2, recommender systems are divided into content-based algorithms and collaborative filtering algorithms.

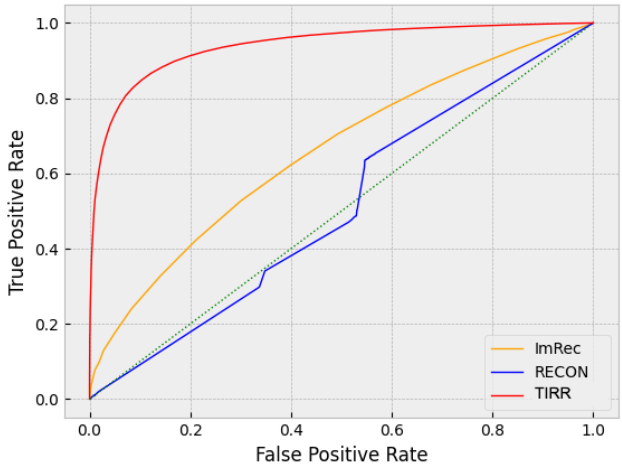


Figure 5.9: Content Based Algorithm ROC Curves demonstrating the significant improvement in AUC with TIRR.

Figure 5.9 displays a comparison of TIRR with other content-based algorithms. As described in Section 2.3.6, *RECON* [113] is a an algorithm that identifies a user's implicit preferences for categorical data, and *ImRec* [92] is an algorithm that uses images to make predictions without the RNN-based component of TIRR, instead using a Random Forest and aggregation function.

RECON struggled to generate effective recommendations on our dataset. As RECON was also evaluated on a private dataset, without comparing the datasets directly, it is difficult to establish why this is, but one possibility is that modern dating services place a higher emphasis on visual content than services did ten years ago, at the time *RECON* was developed. *ImRec* performs better than *RECON*, but performs significantly worse than our proposed method *TIRR*. The key difference between *TIRR* and *ImRec* is the RNN-based process that allows *TIRR* to

interpret historical and time-series data in order to make predictions, whereas *ImRec* treats user preferences in a global way, with no ability to capture individual users preferences.

Algorithm	F1 Score	Precision	Recall	AUC
RECON	0.61	0.56	0.68	0.51
ImRec	0.71	0.60	0.88	0.65
TIRR	0.87	0.86	0.88	0.91

Table 5.5: Results based on best F1 score for content-based algorithms. Here we can see that the proposed method TIRR significantly outperforms the other approaches.

The AUC and maximum F1 score for the three algorithms is described in Table 5.5. The scores are based on the threshold that gave the best F1 score in the training set, used in the test set. We consider that this significant improvement of our proposed method *TIRR* derives from the ability of our algorithm to interpret a user's history of preferences for images over time, and take account of a user's potentially shifting preferences, whereas Imrec provides a global model across all users without distinguishing more than one preference per user at a time, and *RECON* doesn't make use of images at all.

The table also lists the precision and recall at the points where the best F1 score was recorded. While F1 is an excellent measure of overall performance of an algorithm, the individual precision and recall numbers and their balance are particularly important in RS research because precision tends to influence the trust users have in the RS, which in turn affects their use of it [55]. It is noteworthy that while *ImRec* was relatively successful at predicting which image a user would like, its precision was relatively low in comparison with other algorithms, whereas *TIRR* has very high precision, and is therefore more likely to be trusted and used.

5.4.3 TIRR vs Collaborative Filtering

In addition to comparing *TIRR* to other content-based RRSs, tests were also run comparing it to the current best-in-class collaborative filtering algorithms, *RCF* and *LFRR*.

LFRR is a collaborative filtering algorithm based on latent factor models trained by stochastic gradient descent, and *RCF* is a neighbourhood-based collaborative filtering algorithm. *TIRR* outperformed both of these algorithms on our test dataset, although by a slimmer margin than its lead on current content-based filtering algorithms. Nonetheless, this represents a significant advancement in the field of reciprocal recommendation, as in services where images prominently used, our algorithm is likely to be more effective than current collaborative filtering methods.

Algorithm	F1 Score	Precision	Recall	AUC
LFRR	0.86	0.86	0.85	0.90
TIRR	0.87	0.86	0.88	0.91

Table 5.6: Results based on best F1 score for the TIRR and LFRR algorithms. Here we can see that the content-based TIRR improves upon the collaborative filtering-based LFRR.

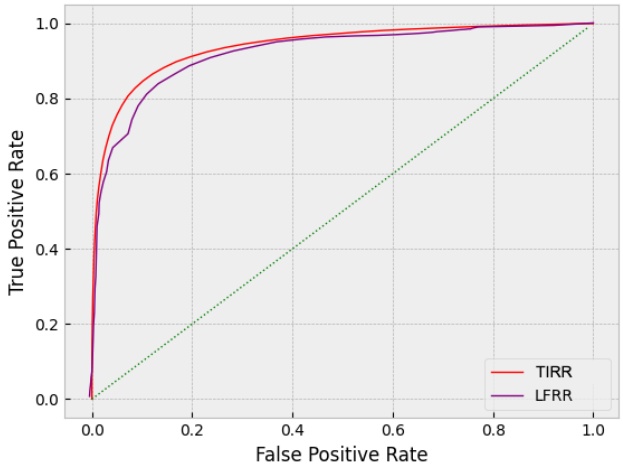


Figure 5.10: ROC Curves showing the performance of the content-based TIRR against the current state of the art collaborative filtering algorithm LFRR.

Table 5.6 lists the peak performance metrics for the two algorithms. In addition to the higher F1 score, *TIRR* also has a comparable balance of precision and recall to *LFRR*.

5.5 Summary

In this section, a novel model that predicts user preference for image-based attractiveness was developed. There are a small number of models in the literature that predict general attractiveness [61, 153], but none that predict personal preference. Based on the large scale evaluation on real world data, that this model successfully differentiates between positive and negative preferences.

Using this model, a novel recommender system, ImRec, that uses scores from our model to predict unidirectional, and subsequently bidirectional preferences. ImRec outperforms the previous best in class content-based recommender system, RECON, which made predictions based on categorical data. The success of ImRec over RECON establishes the importance of images in online dating as compared to text-based information - a subject that would subsequently benefit from an in-depth analysis. It also outperforms the state of the art collaborative filtering RRSs in the case where very little data is available, and therefore helps to solve the cold start problem.

A second algorithm, TIRR, was then developed to interpret user preference history using *only* photographs using an LSTM and make predictions about future preferences for reciprocal recommendation. This can effectively be used as a predictor for the probability of mutual preference between two users, and therefore forms the basis for an effective recommender system. This algorithm outperforms state of the art reciprocal recommender systems in offline tests using a large dataset from a dating service with real users.

This research demonstrates the value of including historical preference in reciprocal recom-

mendation. Previous research has demonstrated the value of using RNNs to interpret sequences of preferences in user-item recommendation, but this is the first time it has been used in reciprocal recommendation. The improvement over a similar algorithm that does not use sequences of data shows the value of this approach.

Finally, the model itself represents a significant advance in the field of content-based reciprocal recommendation. The model's success allows us to draw interesting conclusions about the significance of photographs in online dating, given their strong predictive power in this dataset. It also provides interesting insight into the potential power of content-based algorithms in online dating: while in many fields, they are outperformed by collaborative filtering, the algorithm presented in this paper performs better on evaluation metrics than the current state-of-the-art collaborative filtering algorithm.

Although the results for TIRR in particular are promising within the field of RRSs, it has been demonstrated that recommender systems often do not maintain their offline performance in online settings [21], and so further research is needed to ensure that this performance translates to effective recommendations in online settings. In addition, it is a little counter-intuitive that content-based algorithms outperform collaborative filtering algorithms in this field given that collaborative filtering algorithms significantly outperform content-based filtering algorithms in user-item recommendation [163]. It is possible that TIRR would be outperformed by modern collaborative filtering techniques as applied to reciprocal recommendation.

CHAPTER 6

CONCLUSIONS

This thesis has described a variety of contributions to the field of Reciprocal Recommender Systems, including hybrid, content-based and collaborative filtering systems. Each of these individual contributions represents a significant advancement of the field. This final chapter begins with a brief summary of the methods and the results of the algorithms described in each chapter. The original contributions that each of these chapters represents are then summarised. Finally, the themes that tie these individual contributions together are outlined.

6.1 Summary of Results

In this section, the results from algorithms developed for the three main types of filtering: content-based, collaborative and hybrid filtering are summarised. Figures and tables are reproduced from their respective chapters to illustrate the effectiveness of the methods used in each case.

6.1.1 Collaborative Filtering Results

Prior to the work conducted in this thesis, collaborative filtering algorithms in reciprocal recommendation were memory-based algorithms that calculated recommendations in real time, such as *RCF* [150]. These algorithms struggle to make predictions in reasonable time on larger datasets because of this. In Chapter 3, a novel collaborative filtering algorithm *LFRR* [96] was proposed based on latent factor models. Calculating a latent factor model based on correlations between positive and negative preference expressions allows predictions to be made much more efficiently in real time.

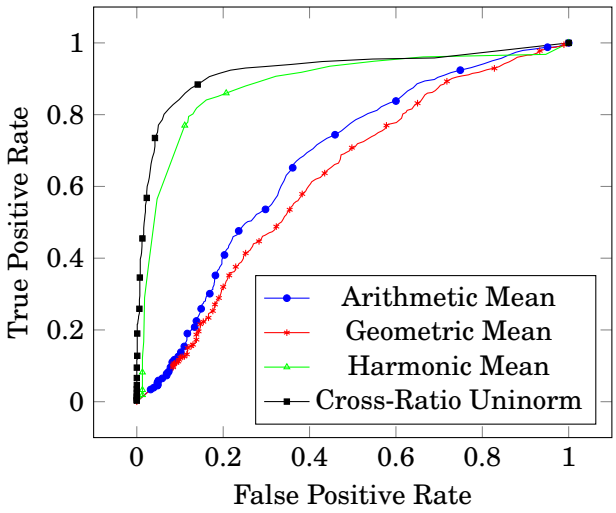


Figure 6.1: ROC curve obtained for each aggregation function considered in the RCF model.

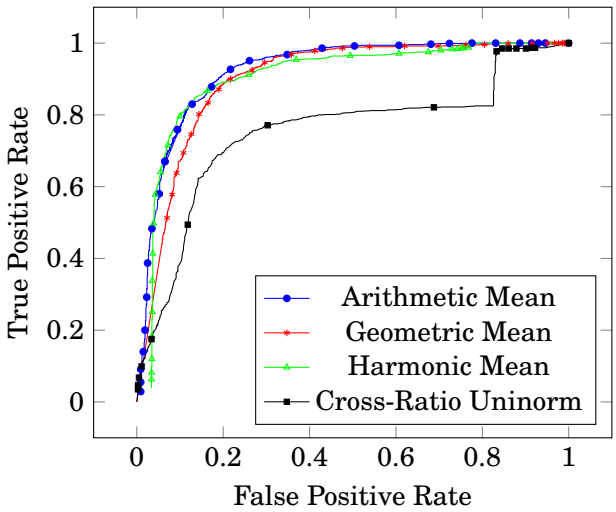


Figure 6.2: ROC curve obtained for each aggregation function considered in the LFRR model.

LFRR was evaluated against *RCF*, which was the previous best in class for collaborative filtering RRSs. In terms of the ROC curve and the best F1 Score, LFRR and RCF had similar performance. However, LFRR performed significantly better where time efficiency was concerned.

As shown in Table 6.1, *RCF* failed to generate predictions in any reasonable time for datasets over 10^5 users, whereas the model-based method *LFRR* was able to generate predictions efficiently up to 10^7, which was the largest amount of test data available.

Collaborative filtering methods for reciprocal recommendation involve generating two separate scores and then combining them into one. Traditionally, this was done with the harmonic mean. Chapter 3 also described experiments based on varying the aggregation function, and the effects this had on the results. Changing the aggregation function did significantly alter the predictive power of the algorithm, and the most effective aggregation function depended on the

Size	RCF Score	RCF List	LRFF Score	LFRR List
10^3	0.003	1.75	1×10^-5	0.0001
10^4	0.005	13.7	1×10^-5	0.001
10^5	0.008	163	1×10^-5	0.025
10^6	0.09	> 1800	1×10^-5	0.63
10^7	0.5	> 1800	1×10^-5	2.0

Table 6.1: Time (seconds) to calculate a user-user score, and to generate recommendations, from a dataset of N interactions

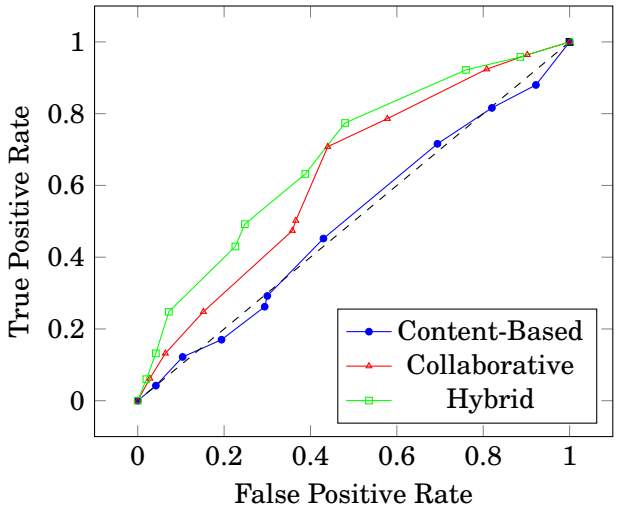


Figure 6.3: ROC curve obtained for the content-based, collaborative and hybrid models.

algorithm used.

6.1.2 Hybrid Filtering Results

Chapter 4 described a novel hybrid algorithm for use on social networks [97]. The algorithm was based on using a combination of latent factor-based collaborative filtering and text embeddings from Word2Vec to recommend recipe creators to users to follow and interact with, based on previous successful interactions.

The hybrid variant of the algorithm was based on aggregating predicted preference scores from each of the two individual algorithms. It was more successful than either the individual algorithms, one of which was *LFRR*, which was the previous state of the art.

6.1.3 Content-Based Results

In Chapter 5, a Siamese network was trained to differentiate between two images: a *Liked* image by one user, and a *Disliked* image by the same user, using a third image as the anchor. This network was able to differentiate between these two classes. This can be visualised: a representative sample of images was used as an input to the network, and the final dense layer

of neurons before the output treated as an embedding in 128-dimensional space. Using *UMAP* [87], these results were mapped into two dimensions, which is visualised below.

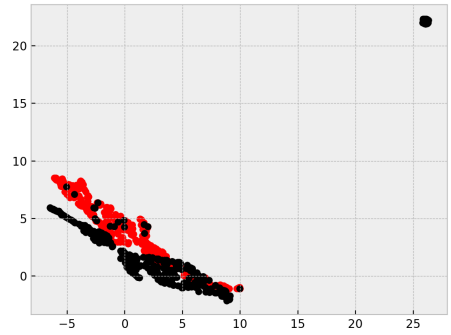


Figure 6.4: Siamese Network Embeddings

From this Siamese Network, a novel algorithm *ImRec* [92] was proposed, and initially constructed using a *Random Forest* to predict whether a user A would like another user B based on outputs from the Siamese network comparing A's previously *Liked* photos with B's photo. This algorithm outperformed previous content-based algorithms such as *RECON*[113].

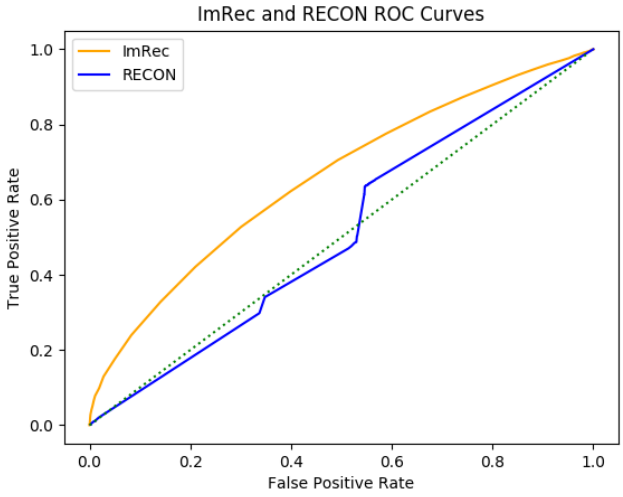


Figure 6.5: ImRec and RECON ROC curves.

However, *ImRec* only outperformed the state-of-the-art collaborative filtering algorithms in cases where there were only a few points of data for an individual user. While this is useful in cold-start situations, most users on a dating service do have more than five expressions of preference.

Building upon the same Siamese Network, a second algorithm was developed, *TIRR* [93]. *TIRR* considered preference history over time using an *LSTM* to interpret sequences of preference. *TIRR* did not predict two unidirectional preferences that were then aggregated as is common in

RRS algorithms, but instead predicted matches directly based on two users' past sequences of preferences for photos.

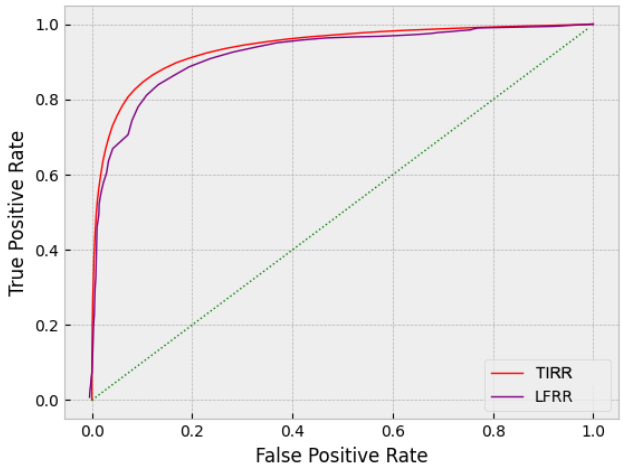


Figure 6.6: ROC Curves showing the performance of the content-based TIRR against the current state of the art collaborative filtering algorithm LFRR.

TIRR improved on not only the previous results from *ImRec* as shown in Figure 6.6 (and therefore the state of the art in content-based filtering) but also the state-of-the-art in collaborative filtering algorithms, *LFRR*. These algorithms demonstrate the importance of images in RRSs for online dating, and also the importance of interpreting historical data as sequences using RNNs in making accurate predictions in these environments.

6.2 Summary of Original Contributions

This chapter outlines the specific original contributions made to the field of reciprocal recommender systems by this thesis. This is organised by content-based, collaborative and hybrid reciprocal recommender systems.

6.2.1 Collaborative Filtering Contributions

The previous state of the art in collaborative filtering reciprocal recommendation was nearest neighbour-based approaches. These were memory-based methods that calculated recommendations in real time. These methods do not perform optimally on larger datasets, as the amount of time required to generate a recommendation grows polynomially with the size of the dataset. In addition, the harmonic mean was used almost exclusively for aggregation of unidirecional preferences, without justification.

In Chapter 3, an algorithm *LFRR* was introduced. This algorithm represented an advance of the field in several ways:

1. *LFRR* represents the first application of latent factor models to reciprocal recommendation. Latent factor models have proved to be extremely effective in user-item recommendation, and *LFRR* demonstrates that they show a similar level of effectiveness in reciprocal recommendation.

2. *LFRR* is shown to be of similar effectiveness to the state of the art nearest neighbour methods, but much more efficient. *LFRR* is a model-based approach, and can generate recommendations at $O(1)$ complexity, meaning that it can be used to predict scores in datasets of arbitrary size.

3. While other methods in the literature have been tested on online dating service datasets of relatively smaller sizes, *LFRR* was tested on a dataset containing hundreds of thousands of users.

In addition, Chapter 3 examined the aggregation functions used to combine two unidirectional preferences into a single bidirectional preference. The precedent of using the harmonic mean was established by *RECON* without justification, and was subsequently used in many other RRSs. The original contributions made by this section are as follows:

1. Four aggregation functions were tested: the arithmetic, harmonic and geometric means and the cross-ratio uninorm. This is the first time that aggregation functions besides the harmonic mean were applied to preference aggregation in reciprocal recommender systems.

2. It was demonstrated through evaluation on datasets using multiple algorithms that not only does the choice of aggregation function significantly impact on the results of the RRS algorithm, but that the choice of aggregation function might depend on the algorithm in question.

6.2.2 Hybrid Contributions

In the recommender system literature, a hybrid system is commonly used to describe one that combines the results of content-based and collaborative filtering algorithms to improve on the results that either one of those could produce. In this sense, prior to this thesis, there were no examples of hybrid reciprocal recommender systems in the literature.

Chapter 4 described an algorithm *HRRS* which combines content-based and collaborative filtering to make recommendations on a social network. The original contributions made by this algorithm are as follows:

1. *HRRS* represents the first hybrid RRS model in the literature, in the sense that it combines content-based and collaborative filtering to produce an algorithm that outperforms either method individually. In particular, the system outperforms standard collaborative filterings very significantly in cold start situations.

2. This one of very few RRS algorithm developed to operate on a single class of users. Other RRS algorithms in the literature work on online dating services, where users can be clearly divided into two classes, often male and female. HRRS operates on a social network which has only one class of users.

3. The content-based part of *HRRS* introduces a novel similarity metric based on word embeddings generated with Word2Vec. This metric is able to accurately predict user preference for recipes and improves the algorithm when used in combination with collaborative filtering.

6.2.3 Content-Based Contributions

Prior to the work conducted in this thesis, content-based filtering in reciprocal recommendation was based on using categorical data such as age, location and hobbies to make recommendations. The results from these algorithms indicated that they are effective on some datasets.

In Chapter 5, two algorithms were described that improved on these results. *ImRec*, described in Section 5.3, used a Siamese Network to differentiate between *Liked* and *Disliked* images of people. The original contributions made to the field by this algorithm are as follows:

1. It provides a model based on a Siamese Network that predicts personal attractiveness using image data. This is the first model to predict attractiveness from photos, and potentially has a variety of applications in other social networks based on images.

2. The RRS based on this Siamese Network, *ImRec*, was the first content-based reciprocal recommender system to use unstructured data such as photos (as opposed to categorical data) to make predictions. Previous recommender systems had relied on categorical data to make predictions, largely ignoring freetext and photo data, which intuitively seems like it would have greater predictive power.

3. ImRec was demonstrated through tests on data from an online dating service to outperform the state of the art content-based RRSs, and demonstrated to outperform the state of the art collaborative filtering algorithms in cold start situations. This represents an advance of the field of content-based RRSs, which were previously inferior in every respect to collaborative filtering RRSs.

TIRR, described in Section 5.4, is an algorithm based on a recurrent neural network that uses the results from the Siamese Network designed as part of *ImRec*, interpreted as a time sequence through an LSTM. The original contributions made by this work are as follows:

1. *TIRR* is the first RRS to make recommendations based on historical sequences of data. This is an important advance, as people's preferences often change over time, and *TIRR* demonstrates that this change can be modelled using LSTMs or similar structures.

2. Most other RRSs in the literature predict two unidirectional preferences and then aggregate them; *TIRR* is an end-to-end algorithm that predicts the probability of a match directly. This makes it unique in the RRS literature currently, and demonstrates that a separate aggregation step is not always necessary.

3. *TIRR* outperforms not only other content-based algorithms, but also collaborative filtering RRSs, making it the state of the art in the context of the literature at the time of writing. It was able to successfully predict matches with a best F1 score of 0.87.

This thesis therefore represents a significant advance in the field of content-based reciprocal recommendation.

6.3 Themes

There are a number of themes that tie together the original contributions outlined in Section 6.2. This section describes these themes with examples from the chapters in this thesis.

The main theme of this thesis has been modernising reciprocal recommender systems. Before the work done in this thesis, many RRS algorithms were based on outdated technology relative to the techniques being used in user-item recommender systems and in modern machine learning in general. The state-of-the-art in content-based reciprocal recommendation was based on preference estimation for categorical data, and the state-of-the-art in collaborative filtering was based on nearest-neighbour algorithms. The algorithms described in this thesis use modern techniques such as LSTMs and Siamese networks, latent factor models and word embeddings to bring the results for reciprocal recommendation closer to the state of the art in user-item recommendation.

In addition to improving on the recommendation part of reciprocal recommender systems, this thesis also focuses on improving the reciprocal element, which is what makes RRSs unique. Prior to the work done in this thesis, reciprocal recommendation almost exclusively generated two unidirectional preferences and aggregated them with the harmonic mean. This thesis explored a number of alternative methods of doing this, including testing alternative aggregation functions across different algorithms, and predicting the likelihood of a match directly through a CNN.

In general, prior to the work done in this thesis, reciprocal recommender systems were often evaluated in their respective papers on relatively small datasets consisting of hundreds or thousands of users. This thesis evaluates not only current algorithms but also the previous state of the art algorithms on much larger datasets, consisting of tens or hundreds of thousands of users from a popular modern online dating service. This gives a more representative example of how these algorithms might perform if implemented into a live service.

A final major theme of this thesis is moving complexity from memory-based computations into models. Most former state-of-the-art algorithms used memory-based methods such as nearest-neighbour methods to compute recommendations in real time. With large datasets, these methods

often do not scale. The algorithms described in this thesis all have a model-based component, and the complexity of a large dataset is abstracted into the model during a training phase. This means that recommendations can be made very quickly in real time even on large datasets.

6.4 Answers to Research Questions

Section 1.3 introduced numbered research questions that this thesis would aim to address. This section clarifies the answers provided to each of these research questions throughout this thesis.

6.4.1 Can the current state of the art for reciprocal recommender systems be improved upon?

This questions is addressed throughout this thesis, but especially in Chapters 3, 4 and 5, where algorithms are demonstrated that outperform the previous state of the art for reciprocal recommendation. All algorithms were tested against and outperformed existing baseline algorithms, some such as TIRR described in Chapter 5 by very significant margins.

6.4.2 What are the most effective methods for reciprocal recommendation, and how does this contrast with the most effective methods for conventional recommendation?

The most effective methods developed as part of this thesis were the content-based methods described in Chapter 5, and esepcially TIRR, which uses historical image data to make predictions about user preferences. However, certain techniques which are particularly effective in user-item recommender systems such as deep learning-based collaborative filtering recommender systems were not explored as part of this research. It is therefore difficult to draw a definitive conclusion about the overall effectiveness of content-based and collaborative algorithms across RRSs in general.

6.4.3 Can models based on unstructured data such as photos be used to improve on current content-based RRSs?

Chapter 5 described algorithms that predict reciprocal user preference based on images. These models were able to estimate binary preference in terms of *Likes* and *Nopes* with a high degree off accuracy, and outperformed baselines such as *RECON*. The answer to this research question is therefore that unstructured data can be used to improve on current content-based RRSs. There is still scope to demonstrate the effectiveness of other types of unstructured data, such as freetext, on RRS evaluation metrics.

6.4.4 Can content-based RRSs be used to improve on the results of collaborative filtering RRSs in cold start situations?

Experiments done in Chapter 5 show that content-based algorithms can outperform collaborative filtering algorithms in cold-start situations. While these experiments represent the current state of the art in reciprocal recommendation, more advanced collaborative filtering algorithms would have to be employed to provide conclusive answers to this research question.

6.4.5 Is historical data a useful predictor of reciprocal preference in RNNs?

The algorithm TIRR presented in Chapter 5, which was designed using an RNN to capture historical preference data, gave significantly better results in terms of evaluation metrics than a similar network that incorporated a Siamese Network with the same architecture, but did not consider the relationship between historical preferences. Historical data is therefore demonstrably a useful predictor of reciprocal preferences in the RRS field.

6.4.6 Can modern techniques such as latent factor models be effectively adapted to reciprocal recommender systems?

This research question is answered in Chapter 3. An algorithm based on latent factor models, LFRR, was successfully applied to reciprocal recommendation, and demonstrated outperform the baseline algorithm RCF.

6.4.7 Can the efficiency of reciprocal recommender systems be improved over and above what's possible with current models?

Efficiency tests were performed on LFRR using *Google Cloud Platform*, and it was demonstrated to be significantly more efficient than existing neighbourhood-based methods. This was especially apparent when the number of users was greater than one million, where the neighbourhood method failed to generate a recommendation list even after 30 minutes.

6.4.8 Does the aggregation function applied have a significant impact on the effectiveness of the recommender system?

Experiments with four aggregation functions - the *Arithmetic Mean*, *Geometric Mean*, *Harmonic Mean* and *Cross-Ratio Uninorm* - demonstrated that aggregation functions do have an impact on the evaluation metrics of RRSs. In particular, the harmonic mean was consistently effective, but effectiveness of the function differed depending on the algorithm used. More research is needed to discover exactly what influences the success of different functions in different situations.

6.4.9 Can hybrid systems be used to improve on the results of content-based and collaborative filtering in reciprocal recommender systems?

As shown in Chapter 4, hybrid systems can be used to improve on the results of individual RRSs. A hybrid system designed with a content-based component and a reciprocal collaborative filtering component outperformed latent factor model-based collaborative filtering. The answer to this research question is therefore also that hybrid systems can be used to improve on the results of individual systems.

6.5 Further Work

This section describes potential future work and other interesting directions that potentially emerge from the research in this thesis. First, future work related to each of the individual areas is described. Finally, general areas for future research in the RRS field are outlined.

6.5.1 Content-Based Filtering

Chapter 5 described algorithms related to content-based filtering with a focus on images as predictors of mutual preference in online dating services. The algorithms presented in the section used images to successfully predict mutual preference, but there is significant scope for breaking down these models to determine why they are successful. Embeddings generated by intermediate layers of the networks used might shed light on specific elements of photos that cause users to feel preference for them, and these embeddings might also be usefully clustered to show specific groups of users who like each other.

Neural networks and in particular machine learning based on facial recognition is notorious for learning biases in the data, especially racial biases [32]. It is not impossible that the networks described in Chapter 5 have a similar problem. This was difficult to test with the data that was available, as an extremely high percentage of the faces in this dataset are Japanese, but it would be interesting to investigate bias and methods for reducing this bias in these algorithms using a more diverse dataset.

6.5.2 Collaborative Filtering

Chapter 3 focused on the development of LFRR, an algorithm using latent factor models to make reciprocal recommendations. This algorithm used baseline methods for learning a latent factor representation to establish the usefulness of this technique in the reciprocal recommendation field, but a number of more advanced techniques are available. Latent factor models based on deep learning [34] and graph neural networks [157] have been particularly effective in terms of evaluation metrics in offline testing, and as social networks can be modelled as graphs, the latter might be a particularly interesting solution for reciprocal recommender systems.

This chapter also included an exploration of the effects of various aggregation functions on reciprocal recommendation. While the results from LFRR demonstrated that the choice of aggregation function had an impact on the evaluation metrics of the algorithm, further research across multiple datasets is needed to determine exactly what should guide the choice of aggregation function in a given situation.

6.5.3 Hybrid Filtering

Chapter 4 described a hybrid filtering algorithm that used unstructured text data and the LFRR collaborative filtering algorithm to make reciprocal recommendations. This was successful, but there is scope for further investigations of hybrid reciprocal recommender systems. This could include the use of more advanced collaborative or content-based filtering solutions (such as the TIRR algorithm described in Chapter 5), or other methods of combining hybrid algorithms, such as a switching algorithm which emphasises content-based filtering during cold-start periods.

6.5.4 General

All the algorithms in this thesis were evaluated using offline testing. This is a normal method of evaluation in the recommender systems domain, as online testing is often costly, and implementing algorithms into real services takes significant time and effort. However, research suggests that offline tests may not always be representative of recommender system effectiveness in real services [21]. Some research has already been conducted by private companies using the algorithms described in this thesis. Engineers at *Tapple*[1], a popular online dating service, found LFRR produced positive results on their service and published their results [116]. However, there is further scope for evaluating all of the algorithms in this thesis in online environments, and measuring their performance in these cases.

This thesis used binary indicators of preference for both training and evaluation. In particular, the algorithms related to online dating used *Likes* and *Nopes* as their primary indicators of positive and negative preference. While this is sufficient for establishing successful or unsuccessful recommendations and a common way to evaluate RRSs, user interactions are more complicated than this, and also include searches, profile views, exchanges of messages and eventually offline meetings. User objectives also vary significantly, with some users looking for more serious relationships than others. Future RRSs might usefully look to include more different aspects of user interactions in developing and evaluating models.

6.6 Summary

This thesis covered a broad variety of contributions to reciprocal recommender systems research. Starting from the introduction of what recommender systems and reciprocal recommender

[1] https://tapple.me/

systems are in Chapter 1, they were then classified into three main types: content-based, collaborative filtering and hybrid systems. Chapter 2 described a machine learning base for the technologies used in recommender systems, and then outlined the progress of the field of both standard user-item recommender systems and reciprocal systems.

Chapters 5, 3 and 4 described original contributions in content-based, collaborative and hybrid systems respectively. These contributions advance the field of reciprocal recommendation, both by adapting modern technologies from conventional recommmender systems and by specifically adapting original techniques from machine learning to design algorithms that outperform the existing state-of-the-art in all three areas.

This thesis is important because prior to this work having been done, there had been very little research into reciprocal recommendation using modern algorithms and techniques. This work significantly improves on previous work, and incorporates a number of original techniques specific to reciprocal systems.

Reciprocal recommendation is an extremely useful field. It is academically interesting because the inherent additional complexity over and above user-item recommender systems requires creative algorithmic solutions. It is also highly valuable from a societal standpoint, because increasingly friendships and romantic connections are being formed online. This has been especially poignant over two of the three years during which the work in this thesis was conducted, where a global pandemic significantly limited real-world social contact for many people. Effective reciprocal recommendation is a tool which can facilitate these relationships, and the author hopes that the work contained in this thesis represents not only a contribution to its academic field, but also to human relationships worldwide.

APPENDIX A

DATA

Throughout this thesis, the primary data source was a popular online dating service in Japan. In Chapter 4, data from a recipe sharing service was used instead. This appendix describes the datasets used during the thesis in each chapter, and explains why decisions were made with regard to data curation and any preprocessing that was done.

A.1 Online Dating Dataset

The primary dataset used for algorithms in Chapter 3 and Chapter 5 was provided by a popular Japanese online dating service. This section describes the form that this data was received in, as well as preprocessing work that was done on the data so that it could be used in training and evaluating algorithms.

A.1.1 Service Description

The online dating service which provided the data for the experiments described in this thesis is primarily based in Japan, with branches in Korea and Taiwan. For this paper, our experimental study focuses on the Japanese service, where the vast majority of users are. By default, the service displays users of a similar age and living in a similar area to the active user. Users can search for other users using attributes such as body type and smoker or non-smoker. The dataset used for our experimental evaluation contained only interactions between members of opposite genders, and this assumption was used in designing all algorithms.

After searching, users can view other users' profiles, which have both selectable attributes such as age and income, and a free text introduction. Users can also provide pictures of themselves. When a user finds another user they want to communicate with, they can send a *Like*. The

receiving user sees a notification, and can choose to either return the *Like* or send a *Nope*, indicating that they are not interested. Once two users have *Liked* each other, they can exchange messages and arrange to meet. This process is visualised in Figure A.1.

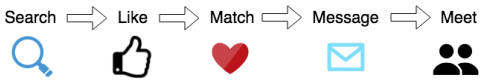


Figure A.1: The usage flow for the online dating service

The service therefore has various indicators of preference that are saved to a database. In increasing strength of interest: 1) viewing a profile; 2) sending a *Like*; 3) sending an initial message; 4) a message exchange; 5) arranging to meet. In most previous studies on reciprocal recommender systems, there was no system of *Likes* and messages were used as an indicator of preference, with a reply being used as a indicator of mutual preference. There are two reasons why a *Like* might be a more useful preference indicator for recommender system design. Firstly, *Likes* have a binary value - it was either sent or not. A message may have positive, strongly positive or even negative value, and extracting this value from free text might be a challenging task. Secondly, a *Like* and a response takes only the effort required to press the button, whereas users may wish to express preference by sending or replying to a message but decide they don't have the time.

Note that the number of *Likes* a user can send is limited by their subscription level, which also adds to the overall sparse nature of the user-user data. It is not possible for users to negatively impact the recommender system by sending *Likes* indiscriminately to users they have no interest in.

The service from which the data was taken has several million users, and send several million *Likes* and *Nopes* every week. Most datasets that have been used to test collaborative filtering-based reciprocal recommender systems in the past have been relatively small in comparison, comprising only a few thousand interactions at most. To the best of our knowledge, there is no example in the literature of a reciprocal recommender system being tested on a dataset of the size of this dataset. This represents a novel contribution to the field but also a novel challenge: users are widely distributed over geographical areas, and also have varying motivations for using the service. Some users create profiles with the intent of recruiting users into dubious schemes or selling products to them. Data curation was therefore an important part of training a useful model.

The following list outlines the users that were excluded from the test data for all models, as well as the reasons for those exclusions.

- Users who live outside Tokyo were excluded from the dataset. Existing heuristics on the service ensure that users are much more likely to see users who are geographically near them. These heuristics skew the distribution of *Likes*, which would impact any models

trained using this data. The majority of users on the service live in Tokyo, so this restriction does not significantly reduce the quantity of data available.

- Users who had not confirmed their identity through uploading a photo of an accepted ID card were excluded from the dataset. Users cannot exchange messages until they have done this, and users who fail to do this are much more likely to be either very new users with no data, or people using the service for reasons besides online dating.
- Users who were marked by the customer service team as being dubious or banned from the service were excluded for similar reasons: their interactions are not likely to be representative of users who are using the service for its intended purpose, and would therefore negatively affect the model.
- Users who had not published any photos were excluded. Users with no photos are statistically extremely unlikely to send or receive many *Likes*, and often leave the service very quickly, making them a subgroup who would skew the accuracy of the final model for active users.
- Users who had voluntarily deactivated their profiles were excluded from the dataset, as they are considered to no longer consent to the use of their data for research.

The following sections discuss specific data collection and curation methods for the algorithms discussed in the main chapters of the thesis.

A.1.2 Data Curation for Collaborative Filtering

This section describes the data used for the collaborative filtering algorithm LFRR from Chapter 3. Users were excluded from the data based on the criteria above, and based on two additional criteria:

- Users who had expressed fewer than ten indicators of preference were excluded. Collaborative filtering without any hybrid elements tends to perform more reliably on users with more data, and this algorithm was not intended to represent a solution to the *Cold-Start Problem*.
- Data was restricted to the past three months. Experiments found that using data from before this point reduced the effectiveness of both LFRR and RCF, possibly because of a change in the service's user interface, or because of drift in user preferences over time.

The data used for training and testing LFRR was in the form of a table with three columns: *user_a_id*, *user_b_id* and *preference*, with an example of this shown in Table A.1. These three columns represent an expression of preference from a user a to a user b, where the preference is either 1.0 for positive preference and 0.0 for negative preference.

user_a_id	user_b_id	preference
1	2	1.0
1	3	1.0
2	3	0.0
3	1	1.0

Table A.1: Example dataframe for collaborative filtering training and testing.

Datasets of varying sizes up to 10^7 were used during the experiment; however, training for the algorithms in Chapter 3 were done with 20000 users and approximately 280000 expressions of preference, because the algorithm used as a baseline (RCF) had training times that were infeasible past this point.

A.1.3 Data Curation for Content-Based Filtering

This section describes the data used for the content-based filtering algorithms *ImRec* and *TIRR* in Chapter 5. In addition to the users excluded based on the criteria listed in Section A.1.1, the following users were excluded from the data for content-based filtering models:

- Users who had not posted a photo containing their face were excluded. Face detection using the *OpenFace*[1] library was used to determine which photos contained faces. The machine learning models were designed to determine reciprocal attractiveness of people to each other, and while other commonly used photos (such as of landscapes and food) might also play a part in determining attractiveness, experiments showed that including them reduced the evaluation metrics of the models.

- Users whose photos were not published or were removed due to infringement of the service's rules were not included. Importantly, the service rules do not permit showing the faces of users besides the owner of the profile, and require users to blur or block these faces themselves, so photos were guaranteed to contain the target user and no others. (Service rules also forbid lewd photos or unlawful photos such as those that infringe copyright.)

Before training of models commenced, the photos themselves were also preprocessed to make them more uniform. This was done through the following process, using Python libraries from *OpenFace* [18] and *ScikitLearn* [109]:

1. Photos were cropped such that the borders of the photo surrounded the detected face. The cropping was done such that all the borders of the face were included in the photo, including hair and neck.

[1] https://cmusatyalab.github.io/openface/

2. Photos were re-sized to be 100x100. This allowed for model training to happen in reasonable time with standardised sizes (where it was much slower with larger photos for very little increase in evaluation metrics).

3. Faces were affine transformed to 2 dimensions. This normalisation process has been shown to increase the accuracy of facial recognition systems [33].

anchor	**positive**	**negative**
1	2	3
2	5	3
3	1	2

Table A.2: Example dataframe for training a Siamese network.

Training of the content-based algorithms was done with two dataframes. The first was a dataframe of tuples, which is the common method of training Siamese Networks, containing an *anchor* (user who is expressing preferences), *positive* (user for whom the anchor has expressed a positive preference) and *negative* (user for whom the anchor has expressed a negative preference). An example of this dataframe is shown in Table A.2. A second dataframe mapped user IDs to their profile images.

A.1.4 Dataset Characteristics and Limitations

A significant advantage of the online dating dataset described in this section is that it was very large, with over 10 million users sending on average over 9 million indicators of preference every week at the time of testing these algorithms. This means that there is a lot of data available for training and testing machine learning models. This also gave a relative freedom to exclude dubious users from the data on the criteria described above, and still retain enough data to effectively train and test a variety of models effectively.

Unlike many online dating services, which are often heavily numerically skewed towards users of one gender, the dataset used in this thesis was well balanced, with a similar number of male and female users. This is a significant advantage in training, as it means that the evaluation metrics are not negatively affected by the sparsity of the data for one gender.

The main limitation of this dataset is that it lacked records of intermediate stages of preference besides binary *Likes* and *Nopes*. While these are the main methods that users use to communicate their preference to each other, and a necessary step before they can exchange messages, there are a number of intermediate stages, such as viewing each other on search pages and viewing profiles, and a number of subsequent stages such as exchanging messages and agreeing to meet. For various technological and ethical reasons, it was not possible to capture all of these intermediate stages, which could potentially have been used either as intermediate predictors of preference between 0.0 and 1.0, or as input to a new model.

Users also have a limited number of *Likes* per month before they are required to purchase more. The decision of whether or not to use a *Like* might be influenced or prevented by the number of *Likes* a user has remaining that month.

A.2 Recipe Sharing Dataset

The dataset used for the hybrid algorithm in Chapter 4 was a dataset from a recipe sharing service. This section describes that dataset, its characteristics and limitations.

A.2.1 Service Description

This data was provided by the international recipe sharing website Cookpad Inc., based in Japan [2]. On Cookpad, users share recipes with titles, pictures and textual information describing the ingredients and descriptions of those recipes. Other users can demonstrate their results when making those recipes via "Cooksnaps", which are mini reviews of the recipes with a picture of their own results.

Users have a number of ways for indicating preference for each other on the site. They can *Follow* each other, where the follower is notified of the followee's public actions. They can also bookmark other users' recipes (an implicit indicator of preference used in the non-reciprocal part of the HRSS approach) and send messages to each other. Users and recipes share very little information about themselves in quantifiable form - for instance, users do not give their age, nationality or explicit preferences such as ratings on other users' recipes, therefore only implicit preferences are used. However, the recipes shared and bookmarked by users in the form of a title, list of ingredients and steps for making the dish, provide a wealth of freetext information about the users taste.

Cookpad's data is quite different to, and in many ways more complex than, the data from a dating service, where users often have a list of attributes and demonstrate clear, direct preferences. However, this data is more representative of many general social networks - including skillsharing platforms - than online dating sites in two ways:

1. The data includes only a single class of users who have to be matched with each other. Dating site data is generally divided into two distinct classifications (male and female), and to the best of our knowledge, no research work has been done on RRSs for single sex dating as of yet.

2. Most of the attribute data for users is unstructured freetext data, as opposed to well structured datasets that have been used in existing RRS models.

[2] https://cookpad.com/

A.2.2 Data Curation for Hybrid Filtering

The hybrid filtering model developed in Chapter 4 was built up of two parts that were initially trained individually: the collaborative filtering model and the content-based filtering model.

A.2.2.1 Collaborative Filtering Data

user_a_id	user_b_id	preference
1	2	4.0
1	3	3.0
2	3	0.0
3	1	2.0

Table A.3: Example dataframe for hybrid collaborative filtering algorithm.

As described in its respective chapter, the collaborative filtering data for the hybrid algorithm was built from $Follows(a,b)$ (whether or not user a follows user b) and $Bookamarks(a,b)$ (the number of times user a had bookmarked recipes by user b). Using data from these two tables, a dataframe was constructed with a preference score between user a and user b as shown in Table A.3.

A.2.2.2 Content-Based Filtering Data

recipe_id	vector
1	[1,6,2]
2	[1,4,5]
3	[4,2,3]

Table A.4: Example vector representations of recipes.

The data for content-based filtering used *Word2Vec* to calculate vectors that represented the contents of recipes based on their title and descriptions. These vectors were considered representations of those recipes in a dataframe similar to the example shown in Table A.4.

As described in Chapter 4, these vectors and user preferences for recipes were used to calculate user preferences for each other based on shared interest in recipes.

A.2.3 Dataset Characteristics and Limitations

Data for 5300 users and 45000 recipes was received from Cookpad, which was enough to train useful models and provide a baseline for reciprocal recommendation based on hybrid filtering using this novel method. Using more data would have meant going back to a time when users were fewer and the data was less relevant due to changes in the service. However, it would be interesting to test this model on a significantly larger dataset and examine whether this would allow higher performance on evaluation metrics.

Due to privacy concerns, user IDs were hashed and no identifying information was included as part of the dataset. However, user profile data is significant to reciprocal recommendation, and if identifying information such as geographical location were able to be used, this might potentially have been useful in the design of the content-based part of the hybrid filtering algorithm.

APPENDIX B

EXPERIMENTAL PROCEDURES

Chapters 3, 4 and 5 described model training and experiments that allowed conclusions to be drawn about those models. This appendix goes into more detail about the training and experiments such that they can be more easily reproduced and validated by readers, including validation methods, hyperparameters used and reasons for their choice.

B.1 LFRR

LFRR, as described in Chapter 3, is a collaborative filtering algorithm based on training a latent factor model. The model is trained using *Stochastic Gradient Descent*. The equations for this are covered in its chapter. The hyperparameters for LFRR are as follows:

1. γ is the learning rate. The objective of SGD is to minimise the error function by computing its gradient and taking steps down it. The learning rate determines the size of these steps. A larger learning rate means that the system may train more quickly, but increases the likelihood of overshooting the minimum with too large steps. SGD will converge to a minimum, but this may not be the global minimum, and other methods are often required to find the optimal solution.

2. λ is the regularisation parameter. Regularisation is used to reduce the chance of the model overfitting to the dataset (fitting excessively to outliers that do not represent the dataset as a whole), by penalising increasing complexity in models. A very complex model is more likely to have overfitted to the training data, and regularisation helps to prevent this.

3. k is the number of latent factors which becomes the size of the vectors in U and V. A small number of latent factors means discarding more information, and therefore potentially

reducing the accuracy of recommendations. A large number of latent factors increases the space complexity of the model, and increases the time required to make individual recommendations. It is therefore important to balance the number of latent factors such that it is the smallest it can be without significantly reducing the accuracy of the model.

4. Finally, the number of iterations represents how many times the model is trained on the dataset. With each training iteration, the error decreases by a smaller amount until the minimum is reached and minor fluctuations are seen instead of decreases, so generally iterations are continued until this point is reached.

K-Fold Cross Validation was used to select hyperparameters. In K-fold cross validation, the data is split into groups. Each group is alternately used as the validation group, with the model trained on the remaining groups. This process is repeated until each group has been used as the validation set. The error of the model is then calculated as the average of the errors across the individual models. K-fold cross validation helps to avoid overfitting hyperparameters to a particular training set. In LFRR, 10 fold were used during cross-validation.

Ranges for hyperparameters were initially determined by a manual search, training the model using cross validation to determine reasonable ranges for them. Tuning was then done by grid search, with ranges of hyperparameters chosen to find the combination of values that gave the best results. The following values for hyperparameters were used to generate the final results:

- $\gamma = 0.01$
- $\lambda = 0.2$
- $k = 5$
- $iterations = 30$

The graphs in Chapter 3 were generated through a separate test dataset which was not used as part of the validation process.

B.2 HRRS

HRRS, described in Chapter 4, is a hybrid filtering algorithm based on a combination of a latent factor model and a Word2Vec model used to predict mutual preference.

The latent factor model was built using the same process as LFRR, so the parameters and the method for refining them is the same as described in Section B.1. The parameters chosen for HRRS by this method were slightly different to LFRR, and are listed below:

- $\gamma = 0.01$

- $\lambda = 0.3$
- $k = 3$
- $iterations = 30$

In addition to the latent factor model, a content-based score was also generated using a Word2Vec model. This model was trained on the *Google News 300* dataset[1]. This is a standard method for training Word2Vec models, so no particular parameter tuning was needed in this case.

The results of the two models in HRRS were combined using an arithmetic mean function. Informal testing was done with harmonic and geometric means, but the arithmetic mean appeared to yield the highest accuracy on the test dataset.

B.3 ImRec

ImRec, described in Chapter 5, is a content-based RRS based on a Siamese Network that, given an anchor image which was liked by a user, predicts preference for a second image. The design of the network itself was arrived at experimentally. Initial designs of ImRec used a convolutional neural network to predict attractiveness of for individual users using examples of previously Liked and Noped users. Insufficient data per user meant that this approach gave relatively low accuracy even for very active users. Unsupervised learning using an autoencoder followed by clustering embeddings, aiming to predict preference for clusters, also did not produce satisfactory results, as users formed a uniform spread and clusters were not predictive of preference. The Siamese Network architecture was tested because of its strong predictive power for one-shot learning (making predictions based on very little data) and its success with facial recognition, and initial informal experiments demonstrated its effectiveness.

Several resources were used as part of preprocessing the images before training the network. A face detector was applied to photos to determine which ones contained faces, and the locations of those faces for centering. This was done with the Python dlib package, and pretrained CNN weights downloaded from the Internet[2]. Faces were flattened using landmarks from the Python dlib face detector, which implements Kazemi et al.'s method [66], with 68-point annotations based on the *iBUG 300-W dataset*[3].

The structure of the CNN used as the symmetrical part of the Siamese Network was based on the commonly used face recognition architecture *Deep Face* [105], with a slightly reduced number of layers, as the complexity of this network meant that it was too slow to train with the quantity of data that was available. The layers within the network as well as the final convolutional layers were adjusted based on initial RMSE results from K-fold cross validation, which was performed

[1] https://huggingface.co/fse/word2vec-google-news-300
[2] http://arunponnusamy.com/files/mmod_human_face_detector.dat
[3] https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/

with five folds. Most experimentation was done manually and based on results from network structures in papers with similar research, because limitations on the hardware available and training times of over an hour meant that exhaustive grid searches over the network structure were not practical. The network was trained over 30 epochs while tuning hyperparameters, and 200 epochs for the final model.

ImRec uses a random forest model to combine results from a user's previously Liked and Noped images in chronological order with a new potential image in order to predict mutual preference. The random forest model uses default parameters except for the number of trees, which was chosen as 64 based on results from K-fold cross validation with 10 folds. The random forest model was trained using a separate dataset from the set used to train the Siamese Network. Final testing for the results presented in the chapter was done with a third dataset.

B.4 TIRR

TIRR, described in Chapter 5, is a model for predicting mutual preference based on the positive and negative preferences for photos of those users over time. It uses the Siamese Network from ImRec as a base, with the results from this model over time fed to an LSTM.

The initial Siamese network and its associated preprocessing is the same as the one used in ImRec, and the testing and data is therefore the same as described in Section B.3. The Siamese Network component of TIRR was pre-trained and tested using the same data structure (although due to time passing between the development of the two algorithms, a fresh dataset was downloaded and processed).

The LSTM used was the default implementation from Tensorflow, with dropout of 0.5, which improved performance on cross-validation. Following pre-training of the Siamese Network on data from 100000 users, the entire network was subsequently trained on data from a further 10000 users for 50 epochs, with cross-validation performed with five folds.

BIBLIOGRAPHY

[1] H. ABDOLLAHPOURI, G. ADOMAVICIUS, R. BURKE, I. GUY, D. JANNACH, T. KAMISHIMA, J. KRASNODEBSKI, AND L. PIZZATO, *Multistakeholder recommendation: Survey and research directions*, User Modeling and User-Adapted Interaction, 30 (2020), pp. 127—-158.

[2] G. ADOMAVICIUS AND A. TUZHILIN, *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 (2005), pp. 734–749.

[3] G. ADOMAVICIUS AND A. TUZHILIN, *Context-aware recommender systems*, Recommender Systems Handbook, (2010).

[4] D. AGARWAL AND B.-C. CHEN, *Regression-based latent factor models*, in Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '09, New York, NY, 2009, ACM, pp. 19–28.

[5] C. AGGARWAL, *Recommender Systems: The Textbook*, Springer, London, England, 1st ed., 2016.

[6] C. C. AGGARWAL AND S. PARTHASARATHY, *Mining massively incomplete data sets by conceptual reconstruction*, in Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '01, New York, NY, 2001, ACM, pp. 227–232.

[7] E. AHMED, M. JONES, AND T. MARKS, *An improved deep learning architecture for person re-identification*, in Proceedings of the 2015 Conference on Computer Vision and Pattern Recognition, CVPR 2015, IEEE, 2015, pp. 3908–3916.

[8] J.-W. AHN, P. BRUSILOVSKY, J. GRADY, D. HE, AND S. SYN, *Open user profiles for adaptive news systems: Help or harm?*, in Proceedings of the 16th International Conference on World Wide Web, WWW 2007, New York, NY, 2007, ACM, pp. 11–20.

[9] A. AJESH, J. NAIR, AND P. S. JIJIN, *A random forest approach for rating-based recommender system*, in Proceedings of the 2016 International Conference on Advances in

Computing, Communications and Informatics, ICACCI 2016, IEEE, 2016, pp. 1293–1297.

[10] J. AKEHURST, *A probabilistic reciprocal recommender with temporal dynamics*, in PhD Thesis, 2011.

[11] J. AKEHURST, I. KOPRINSKA, K. YACEF, L. PIZZATO, J. KAY, AND T. REJ, *Ccr — a content-collaborative reciprocal recommender for online dating*, in Twenty-Second International Joint Conference on Artificial Intelligence, 2011.

[12] M. AL-ZEYADI, F. COENEN, AND A. LISITSA, *User-to-user recommendation using the concept of movement patterns: A study using a dating social network*, SCITEPRESS - Science and Technology Publications, (2017).

[13] A. ALANZI AND M. BAIN, *A people-to-people content-based reciprocal recommender using hidden markov models*, in Proceedings of the 7th ACM conference on Recommender systems, RecSys '13, New York, NY, 2013, ACM, pp. 303–306.

[14] N. D. ALMALIS, G. A. TSIHRINTZIS, AND N. KARAGIANNIS, *A content based approach for recommending personnel for job positions*, in Proceedings of the 5th International Conference on Information, Intelligence, Systems and Applications, IISA 2014, Orlando, FL, USA, 2014, IEEE.

[15] D. ANAND AND K. BHARADWAJ, *Utilizing various sparsity measures for enhancing accuracy of collaborative recommender systems based on local and global similarities*, Expert Systems with Applications, 38 (2011), pp. 5101–5109.

[16] O. APPEL, F. CHICLANA, J. CARTER, AND H. FUJITA, *Cross-ratio uninorms as an effective aggregation mechanism in sentiment analysis*, Knowledge-Based Systems, 124 (2017), pp. 16–22.

[17] M. BALABANOVIC AND Y. SHOHAM, *Combining content-based and collaborative recommendation*, in Communications of the ACM, ACM 1997, New York, NY, 1997, ACM.

[18] T. BALTRUSAITIS, A. ZADEH, Y. C. LIM, AND L.-P. MORENCY, *Openface 2.0: Facial behavior analysis toolkit*, in 2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018), IEEE, 2018, pp. 59–66.

[19] T. BANSAL, D. BELANGER, AND A. MCCALLUM, *Ask the gru: Multi-task learning for deep text recommendations*, in Proceedings of the 10th ACM Conference on Recommender Systems, Recsys 2016, New York, NY, 2016, ACM, pp. 107–114.

[20] J. BASILICO AND T. HOFMANN, *Unifying collaborative and content-based filtering*, in Proceedings of the twenty-first international conference on Machine learning, ICML '04, New York, NY, 2004, ACM, pp. 09–09.

[21] J. BEEL, M. GENZMEHR, S. LANGER, A. NÜRNBERGER, AND B. GIPP, *A comparative analysis of offline and online evaluations and discussion of research paper recommender system evaluation*, in Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation, RecSys '13, New York, NY, 2013, ACM, pp. 7–14.

[22] R. BELL AND Y. KOREN, *Lessons from the netflix prize challenge*, ACM SIGKDD Explorations Newsletter - Special issue on visual analytics, 9 (2007), pp. 75–79.

[23] R. BELL, Y. KOREN, AND C. VOLINSKY, *Matrix factorization techniques for recommender systems*, Computer, 42 (2009), pp. 30–37.

[24] J. BENESTY, J. CHEN, Y. HUANG, AND I. COHEN, *Pearson correlation coefficient*, Noise Reduction in Speech Processing, 2 (2009), pp. 1–4.

[25] L. BERTINETTO, J. VALMADRE, J. HENRIQUES, A. VEDALDI, AND P. TORR, *Fully-convolutional siamese networks for object tracking*, in Proceedings of the 2016 European Conference on Computer Vision, ECCV 2016, Springer, 2016, pp. 850–865.

[26] D. BILLSUS, M. PAZZANI, AND J. CHEN, *A learning agent for wireless news access*, in Proceedings of the International Conference on Intelligent User Interfaces, ICIU 2002, 2002, pp. 33–36.

[27] J. BREESE, D. HECKERMAN, AND C. KADIE, *Empirical analysis of predictive algorithms for collaborative filtering*, in Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, UAI '98, San Francisco, CA, 1998, Morgan Kaufmann Publishers Inc., pp. 24–26.

[28] R. BURKE, *Integrating knowledge-based and collaborative-filtering recommender systems*, in In proceedings of Artificial Intelligence for Electronic Commerce: Papers from the AAAI Workshop, AAAI '99, 1999, pp. 69–72.

[29] ——, *Hybrid recommender systems: Survey and experiments*, User Modeling and User-Adapted Interaction, 12 (2002), pp. 331–370.

[30] ——, *Multisided fairness for recommendation*, Arxiv.org, (2017).

[31] P. CAMPOS, F. DÍEZ, AND I. CANTADOR, *Time-aware recommender systems: A comprehensive survey and analysis of existing evaluation protocols*, User Modeling and User-Adapted Interaction, 24 (2013), pp. 67–119.

[32] J. G. CAVAZOS, P. J. PHILLIPS, C. D. CASTILLO, AND A. J. O'TOOLE, *Accuracy comparison across face recognition algorithms: Where are we on measuring race bias?*, IEEE Transactions on Biometrics, Behavior, and Identity Science, 3 (2021).

[33] X. CHAI, S. SHAN, AND W. GAO, *Pose normalization for robust face recognition based on statistical affine transformation*, in Fourth International Conference on Information, Communications and Signal Processing, ICS 2003, IEEE, 2003.

[34] H.-T. CHENG, L. KOC, J. HARMSEN, T. SHAKED, T. CHANDRA, H. ARADHYE, G. ANDERSON, G. CORRADO, W. CHAI, M. ISPIR, R. ANIL, Z. HAQUE, L. HONG, V. JAIN, X. LIU, AND H. SHAH, *Wide and deep learning for recommender systems*, in Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS 2016, New York, NY, 2016, ACM, pp. 7–10.

[35] Y. H. CHO, J. K. KIMB, AND S. H. KIM, *A personalized recommender system based on web usage mining and decision tree induction*, Expert Systems with Applications, 23 (2002), pp. 329–342.

[36] K. CHOI, G. FAZEKAS, M. SANDLER, AND K. CHO, *Convolutional recurrent neural networks for music classification*, in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, Montreal, QC, Canada, 2017, IEEE.

[37] P. COTTER AND B. SMYTH, *Ptv: Intelligent personalized tv guides*, Twelfth Conference on Innovative Applications of Artificial Intelligence, (2000).

[38] P. COVINGTON, J. ADAMS, AND E. SARGIN, *Deep neural networks for youtube recommendations*, in Proceedings of the 10th ACM Conference on Recommender Systems, Recsys '16, New York, NY, 2016, ACM, pp. 191–198.

[39] P. CREMONESI, Y. KOREN, AND R. TURRIN, *Performance of recommender algorithms on top-n recommendation tasks*, in Proceedings of the fourth ACM conference on Recommender systems, RecSys '10, New York, NY, 2010, ACM, pp. 39–46.

[40] J. DENG, W. DONG, R. SOCHER, L.-J. LI, K. LI, AND L. FEI-FEI, *Imagenet: A large-scale hierarchical image database*, in Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, CCVPR 2009, Miami, FL, 2009, IEEE, pp. 248–255.

[41] J. DEVLIN, M.-W. CHANG, K. LEE, AND K. TOUTANOVA, *Bert: Pre-training of deep bidirectional transformers for language understanding*, arXiv, (2019).

[42] Y. DING, Y. ZHANG, L. LI, W. XU, AND H. WANG, *A reciprocal recommender system for graduates' recruitment*, in International Conference on Information Technology in Medicine and Education, ITME 2016, New York, NY, 2016, IEEE.

[43] E. EZIN, I. PALOMARES, AND J. NEVE, *Group decision making with collaborative-filtering 'in the loop': interaction-based preference and trust elicitation*, in 2019 IEEE Interna-

tional Conference on Systems, Man and Cybernetics, IEEE SMC 2019, New York, NY, 2019, IEEE.

[44] Y.-Y. FAN, S. LIU, B. LI, Z. GUO, A. SAMAL, J. WAN, AND S. Z. LI, *Label distribution-based facial attractiveness computation by deep residual learning*, IEEE Transactions on Multimedia, 20 (2018), pp. 2196–2208.

[45] M.-L. FEN, *Choosing online partners in the virtual world: How online partners' characteristics affect online dating*, ProQuest Dissertations Publishing, (2005).

[46] Y. FREUND AND R. SCHAPIRE, *A decision-theoretic generalization of on-line learning and an application to boosting*, Journal of Computer and System Sciences, 55 (1997), pp. 119–139.

[47] D. GALE AND L. SHAPLEY, *College admissions and the stability of marriage*, Am. Math. Monthly, 69 (1962).

[48] R. GARCIA AND X. AMATRIAIN, *Weighted content based methods for recommending connections in online social networks*, in Proceedings of the fourth ACM conference on Recommender systems, RecSys 2010, New York, NY, 2010, ACM, pp. 68–71.

[49] M. GE, C. DELGADO-BATTENFELD, AND D. JANNACH, *Beyond accuracy: evaluating recommender systems by coverage and serendipity*, in Proceedings of the fourth ACM conference on Recommender systems, RecSys '10, New York, NY, 2010, ACM, pp. 257–260.

[50] D. GOLDBERG, D. A. NICHOLS, B. OKI, AND D. B. TERRY, *Using collaborative filtering to weave an information tapestry*, Communications of the ACM, 35 (1992), pp. 61–70.

[51] S. HA, *Digital content recommender on the internet*, IEEE Intelligent Systems, 21 (2006), pp. 70–77.

[52] H. A. M. HASSAN, G. SANSONETTI, F. GASPARETTI, A. MICARELLI, AND J. BEEL, *Bert, elmo, use and infersent sentence encoders: The panacea for research-paper recommendation?*, in Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, New York, NY, 2019, ACM.

[53] J. HE AND W. CHU, *A social network-based recommender system (snrs)*, Data Mining for Social Network Data, 12 (2010), pp. 47–74.

[54] K. HE, X. ZHANG, S. REN, AND J. SUN, *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*, in Proceedings of the IEEE International Conference on Computer Vision, ICCV 2015, Montreal, QC, Canada, 2015, IEEE, pp. 1026–1034.

[55] J. HERLOCKER, J. A. KONSTAN, L. TERVEEN, AND J. RIEDL, *Evaluating collaborative filtering recommender systems*, ACM Transactions on Information Systems, 22 (2004).

[56] J. L. HERLOCKER, J. A. KONSTAN, A. BORCHERS, AND J. RIEDL, *An algorithmic framework for performing collaborative filtering*, in Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '99, New York, NY, 1999, ACM, pp. 230—-237.

[57] T. K. HO, *Random decision forests*, in Proceedings of 3rd International Conference on Document Analysis and Recognition, CDAR 1995, Montreal, QC, Canada, 1995, IEEE.

[58] S. HOCHREITER, *The vanishing gradient problem during learning recurrent neural nets and problem solutions*, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6 (1998), pp. 107–116.

[59] S. HOCHREITER AND J. SCHMIDHUBER, *Long short-term memory*, Neural Computation, 9 (1997), pp. 1735—-1780.

[60] W. HONG, S. ZHENG, H. WANG, AND J. SHI, *A job recommender system based on user clustering*, Journal of Computers, 8 (2013), pp. 1960–1967.

[61] R. JAHANDIDEH, A. T. TARGHI, AND M. TAHMASBI, *Physical attribute prediction using deep residual neural networks*, arXiv, (2018).

[62] J.BOBADILLA, F.ORTEGA, A.HERNANDO, , AND A.GUTIÉRREZ, *Recommender systems survey*, Knowledge-Based Systems, 46 (2013), pp. 109–132.

[63] C. F. JEKEL AND R. T. HAFTKA, *Classifying online dating profiles on tinder using facenet facial embeddings*, Arxiv.org, (2018).

[64] M. KAMINSKAS AND D. BRIDGE, *Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems*, ACM Transactions on Interactive Intelligent Systems, 7 (2016).

[65] M. KARIMIA, D. JANNACHB, AND M. JUGOVACC, *News recommender systems – survey and roads ahead*, Information Processing & Management, 54 (2018), pp. 1203–1227.

[66] V. KAZEMI AND J. SULLIVAN, *One millisecond face alignment with an ensemble of regression trees*, in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, IEEE, 2014.

[67] D. KHATTAR, V. KUMAR, V. VARMA, AND M. GUPTA, *Weave&rec: A word embedding based 3-d convolutional network for news recommendation*, in Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, New York, NY, 2018, ACM, pp. 1855–1858.

[68] J. KIM, B. LEE, M. SHAW, H. CHANG, AND W. NELSON, *Application of decision-tree induction techniques to personalized advertisements on internet storefronts*, International Journal of Electronic Commerce, 5 (2001), pp. 45–62.

[69] A. KLEINERMAN, A. ROSENFELD, AND S. KRAUS, *Providing explanations for recommendations in reciprocal environments*, in Proceedings of the 12th ACM Conference on Recommender Systems, RecSys '18, New York, NY, 2018, ACM, pp. 22–30.

[70] A. KLEINERMAN, A. ROSENFELD, F. RICCI, AND S. KRAUS, *Optimally balancing receiver and recommended users' importance in reciprocal recommender systems*, in Proceedings of the 12th ACM Conference on Recommender Systems, RecSys '18, New York, NY, 2018, ACM, pp. 131–139.

[71] G. KOCH, R. ZEMEL, AND R. SALAKHUTDINOV, *Siamese neural networks for one-shot image recognition*, in Proceedings of the 2015 ICML Deep Learning workshop, ICML, 2015.

[72] I. KOPRINSKA AND K. YACEF, *People-to-people reciprocal recommenders*, Recommender Systems Handbook, (2015), pp. 556–578.

[73] Y. KOREN, *Factorization meets the neighborhood: a multifaceted collaborative filtering model*, in Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '08, New York, NY, 2008, ACM, pp. 426–434.

[74] A. KRIZHEVSKY, I. SUTSKEVER, AND G. E. HINTON, *Imagenet classification with deep convolutional neural networks*, Communications of the ACM, 60 (2017).

[75] A. KRZYWICKI, W. WOBCKE, X. CAI, A. MAHIDADIA, M. BAIN, P. COMPTON, AND Y. S. KIM, *Interaction-based collaborative filtering methods for recommendation in online dating*, in Proceedings of the International Conference on Web Information Systems Engineering, WISE 2010, Springer, 2010, pp. 342–356.

[76] X. N. LAM, T. VU, T. D. LE, AND A. D. DUONG, *Addressing cold-start problem in recommendation systems*, in Proceedings of the 2nd international conference on Ubiquitous information management and communication, ICUIMC '08, New York, NY, 2008, ACM, pp. 208–211.

[77] A. LAMPROPOULOS, P. LAMPROPOULOU, AND G. TSIHRINTZIS, *A cascade-hybrid music recommender system for mobile services based on musical genre classification and personality diagnosis*, Multimedia Tools and Applications, 59 (2012), pp. 241–258.

[78] K. LANG, *Newsweeder: Learning to filter netnews*, in Proceedings 12th International Conference on Machine Learning„ ICML 1995, 1995, pp. 331–339.

[79] Y. LECUN, L. BOTTOU, Y. BENGIO, AND P. HAFFNER, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE, 86 (1998), pp. 2278–2324.

[80] C. LEI, D. LIU, W. LI, Z.-J. ZHA, AND H. LI, *Comparative deep learning of hybrid representations for image recommendations*, in The IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, IEEE, 2016, pp. 2545–2553.

[81] Y. LEWENBERG, Y. BACHRACH, S. SHANKAR, AND A. CRIMINISI, *Predicting personal traits from facial images using convolutional neural networks augmented with facial landmark information*, in Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI 2016, AAAI, 2016, pp. 4365–4366.

[82] J. LIN, K. SUGIYAMA, M.-Y. KAN, AND T.-S. CHUA, *Addressing cold-start in app recommendation: latent user models constructed from twitter followers*, in Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, SIGIR '13, New York, NY, 2013, ACM, pp. 283–292.

[83] W. LING, C. DYER, A. W. BLACK, AND I. TRANCOSO, *Two/too simple adaptations of word2vec for syntax problems*, in Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL '15, Denver, Colorado, 2015, Association for Computational Linguistics, pp. 1299–1304.

[84] Q. LIU, S. WU, AND L. WANG, *Deepstyle: Learning user preferences for visual recommendation*, in Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2017, New York, NY, 2017, ACM, pp. 841–844.

[85] G. M. LUNARDI, *Representing the filter bubble: Towards a model to diversification in news*, in Proceedings of the 23rd international conference on Advances in Conceptual Modeling, EC 2019, Springer, 2019, pp. 239–246.

[86] G. MACHADO, L. GUILHERME, M. MACHADOB, V. M. JOSÉ, AND P. DE OLIVEIRA, *A metric for filter bubble measurement in recommender algorithms considering the news domain*, Applied Soft Computing, 97 (2020).

[87] L. MCINNES, J. HEALY, AND J. MELVILLE, *Umap: Uniform manifold approximation and projection for dimension reduction*, Arxiv.org, (2018).

[88] P. J. MCPARLANE, Y. MOSHFEGHI, AND J. M. JOSE, *Nobody comes here anymore, it's too crowded; predicting image popularity on flickr*, in Proceedings of International Conference on Multimedia Retrieval, ICMR 2014, ACM, 2014, pp. 385–391.

[89] T. MIKOLOV, K. CHEN, G. S. CORRADO, AND J. DEAN, *Efficient estimation of word representations in vector space*, ICLR, 0 (2013), p. 0.

[90] D. MISHKIN, N. SERGIEVSKIY, AND J. MATAS, *Systematic evaluation of convolution neural network advances on the imagenet*, Computer Vision and Image Understanding, 161 (2017), pp. 11–19.

[91] R. J. MOONEY AND L. ROY, *Content-based book recommending using learning for text categorization*, in Proceedings of the fifth ACM conference on Digital libraries, DL 2000, New York, NY, 2000, ACM, pp. 195–204.

[92] J. NEVE AND R. MCCONVILLE, *Imrec: Learning reciprocal preferences using images*, in Proceedings of the Fourteenth ACM Conference on Recommender Systems, Recsys '2020, New York, NY, 2020, ACM, pp. 170–179.

[93] ——, *Photos are all you need for reciprocal recommendation in online dating*, Arxiv.org, (2021).

[94] J. NEVE AND I. PALOMARES, *Aggregation strategies in user-to-user reciprocal recommender systems*, in Proceedings of the 2018 IEEE International Conference on Systems, Man and Cybernetics, IEEE SMC 2018, New York, NY, 2018, IEEE, pp. 2299–2304.

[95] ——, *Arikui - a dubious user detection system for online dating in japan*, in 2018 IEEE International Conference on Systems, Man, and Cybernetics, IEEE SMC 2018, Miyazaki, Japan, 2018, IEEE, pp. 853–871.

[96] ——, *Latent factor models and aggregation operators for collaborative filtering in reciprocal recommender systems*, in Proceedings of the 13th ACM Conference on Recommender Systems, RecSys '19, New York, NY, 2019, ACM.

[97] ——, *Hybrid reciprocal recommender systems: Integrating item-to-user principles in reciprocal recommendation*, in 4th International Workshop on Mining Actionable Insights from Social Networks, WebConf 2020, 2020.

[98] J. NGIAM, A. KHOSLA, M. KIM, J. NAM, H. LEE, AND A. Y. NG, *Multimodal deep learning*, in Proceedings of the International Conference on Machine Learning, ICML 2011, 2011.

[99] P. NGUYEN, J. DINES, AND J. KRASNODEBSKI, *A multi-objective learning to re-rank approach to optimize online marketplaces for multiple stakeholders*, Arxiv.org, (2017).

[100] T. T. NGUYEN, P. HUI, F. M. HARPER, L. TERVEEN, AND J. A. KONSTAN, *Exploring the filter bubble: the effect of using recommender systems on content diversity*, in Proceedings of the 23rd international conference on World wide web, WWW 2014, Orlando, FL, USA, 2014, IEEE, pp. 677–686.

[101] O. OTAKORE AND C. UGWU, *Online matchmaking using collaborative filtering and reciprocal recommender systems*, The International Journal of Engineering and Science (IJES), 7 (2018), pp. 07–21.

[102] O. OZGOBEK, J. GULLA, AND R. ERDUR, *A survey on challenges and methods in news recommendation*, in Proceedings of the 10th International Conference on Web Information Systems and Technologies, WEBIST 2014, New York, NY, 2014, ACM, pp. 278–285.

[103] M. O'CONNOR AND J. HERLOCKER, *Clustering items for collaborative filtering*, in Proceedings of the ACM SIGIR Workshop on Recommender Systems, SIGIR 1999, New York, NY, 1999, ACM.

[104] I. PALOMARES, J. NEVE, C. PORCEL, L. PIZZATO, I. GUY, AND E. HERRERA-VIEDMA, *Reciprocal recommender systems: Analysis of state-of-art literature, challenges and opportunities towards social recommendation*, Information Fusion, 69 (2021), pp. 103–127.

[105] O. PARKHI, A. VEDALDI, AND A. ZISSERMAN, *Deep face recognition*, Proceedings of the British Machine Vision, 1 (2015), p. 6.

[106] A. PATEREK, *Improving regularized singular value decomposition for collaborative filtering*, in Proceedings of KDD cup and workshop, KDD '07, New York, NY, 2007, ACM.

[107] M. PAZZANI, *A framework for collaborative, content-based and demographic filtering*, Artificial Intelligence Review, 13 (1999), pp. 393–408.

[108] M. PAZZANI, *A framework for collaborative, content-based and demographic filtering*, Artificial Intelligence Review, 13 (1999), pp. 393–408.

[109] F. PEDREGOSA, G. VAROQUAUX, A. GRAMFORT, V. MICHEL, B. THIRION, O. GRISEL, M. BLONDEL, P. PRETTENHOFER, R. WEISS, V. DUBOURG, ET AL., *Scikit-learn: Machine learning in python*, the Journal of machine Learning research, 12 (2011), pp. 2825–2830.

[110] M. E. PETERS, M. NEUMANN, M. IYYER, M. GARDNER, C. CLARK, K. LEE, AND L. ZETTLEMOYER, *Deep contextualized word representations*, arXiv, (2018).

[111] L. PIZZATO, T. CHUNG, T. REJ, I. KOPRINSKA, K. YACEF, AND J. KAY, *Learning user preferences in online dating*, Technical Report 656, Univeristy of Sydney, (2010).

[112] L. PIZZATO, T. REJ, J. AKEHURST, I. KOPRINSKA, K. YACEF, AND J. KAY, *Recommending people to people: the nature of reciprocal recommenders with a case study in online dating*, User Model User-Adap Inter, 23 (2013), pp. 447–488.

[113] L. PIZZATO, T. REJ, T. CHUNG, I. KOPRINSKA, AND J. KAY, *Recon: a reciprocal recommender for online dating*, in Proceedings of the fourth ACM conference on Recommender systems, RecSys '10, New York, NY, 2010, ACM, pp. 207–214.

[114] S. PRABHAKAR, G. SPANAKIS, AND O. ZAIANE, *Reciprocal recommender system for learners in massive open online courses (moocs)*, in International Conference on Web-Based Learning, ICWL 2017, New York, NY, 2017, Springer, pp. 157–167.

[115] Y. QU, H. LIU, Y. DU, AND Z. WU, *Reciprocal ranking: A hybrid ranking algorithm for reciprocal recommendation*, in 15th Pacific Rim International Conference on Artificial Intelligence, Nanjing, China, 2018.

[116] R. RAMANATHAN, N. SHINADA, AND S. PALANIAPPAN, *Building a reciprocal recommendation system at scale from scratch: Learnings from one of japan's prominent dating applications*, in Proceedings of the fourteenth ACM Conference on Recommender Systems, Recsys 2020, New York, NY, 2020, ACM, pp. 566–567.

[117] A. M. RASHID, S. K. LAM, G. KARYPIS, AND J. RIEDL., *Clustknn: a highly scalable hybrid model-& memory-based cf algorithm*, in WebKDD-06, KDD Workshop on Web Mining and Web Usage Analysis, KDD '06, 2006.

[118] C. V. RIJSBERGEN, *Foundation of evaluation*, Journal of Documentation, (1974).

[119] M. RODRIGUEZ-GARCIA, R. VALENCIA-GARCIA, R. PALACIOS, AND J. GOMEZ-BERBIS, *Blinddate recommender: A context-aware ontology-based dating recommendation platform*, Journal of Information Science, 45 (2019), pp. 573–591.

[120] X. RONG, *word2vec parameter learning explained*, axIv, 0 (2011).

[121] R. ROTHE, R. TIMOFTE, AND L. V. GOOL, *Some like it hot - visual guidance for preference prediction*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Orlando, FL, USA, 2016, IEEE, pp. 5553–5561.

[122] S. SAFAVIAN AND D. LANDGREBE, *A survey of decision tree classifier methodology*, IEEE Transactions on Systems, Man, and Cybernetics, 21 (1991), pp. 660–674.

[123] R. SALAKHUTDINOV, A. MNIH, AND G. HINTON, *Restricted boltzmann machines for collaborative filtering*, in Proceedings of the 24th international conference on Machine learning, ICML '07, New York, NY, 2007, ACM, pp. 791–798.

[124] B. SARWAR, G. KARYPIS, J. KONSTAN, AND J. RIEDL, *Application of dimensionality reduction in recommender system - a case study*, Technical Report - University of Minnesota, (2000).

[125] ——, *Application of dimensionality reduction in recommender system - a case study*, in Workshop on Web Mining for e-Commerce: Challenges and Opportunities, WebKDD '00, New York, NY, 2000, ACM.

[126] B. SARWAR AND J. RIEDL, *Item-based collaborative filtering recommendation algorithms*, in Proceedings of the 10th International World Wide Web Conference, WWW '01, New York, NY, 2001, ACM, pp. 285–295.

[127] B. M. SARWAR, G. KARYPIS, J. KONSTAN, , AND J. RIEDL, *Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering*, in Proceedings of the fifth international conference on computer and information technology, 2002.

[128] A. SCHCLAR, A. TSIKINOVSKY, L. ROKACH, A. MEISELS, AND L. ANTWARG, *Ensemble methods for improving the performance of neighborhood-based collaborative filtering*, in Proceedings of the third ACM conference on Recommender systems, ACM Recsys '09, New York, NY, 2009, ACM, pp. 261—-264.

[129] A. I. SCHEIN, A. POPESCUL, L. H. UNGAR, AND D. M. PENNOCK, *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, New York, NY, 2002, ACM, pp. 253–260.

[130] U. SHARDANAND AND P. MAES, *Social information filtering: Algorithms for automating 'word of mouth*, Human Factors in Computing Systems, (1995).

[131] B. SHETH AND P. MAES, *Evolving agents for personalized information filtering*, in Proceedings of 9th IEEE Conference on Artificial Intelligence for Applications, AIA 2002, Orlando, FL, USA, 2002, IEEE, pp. 345–352.

[132] Z. SITING, H. WENXING, Z. NING, AND Y. FAN, *Job recommender systems: A survey*, in Proceedings of the 7th International Conference on Computer Science & Education, ICCSE '12, Melbourne, VIC, Australia, 2012, IEEE, pp. 920–924.

[133] B. SMYTH AND P. MCCLAVE, *Similarity vs. diversity*, in Proceedings of the International Conference on Case-Based Reasoning, ICCBR 2001, Springer, 2001, pp. 347–361.

[134] L. STAFFORD AND J. R. RESKE, *Idealization and communication in long-distance premarital relationships*, Family Relations, 39 (1990), pp. 274–279.

[135] P. SYMEONIDIS, A. NANOPOULOS, A. PAPADOPOULOS, AND Y. MANOLOPOULOS, *Collaborative recommender systems: Combining effectiveness and efficiency*, Expert Systems with Applications, 34 (2008), pp. 2995–3013.

[136] T. TRAN AND R. COHEN, *Hybrid recommender systems for electronic commerce*, Knowledge-Based Electronic Markets, AAAI Press, (2000).

[137] E. TSOUROUGIANNI AND N. AMPAZIS, *Recommending who to follow on twitter based on tweet contents and social connections*, Social Networking, 2 (2013), pp. 165–173.

[138] K. TU, B. F. RIBEIRO, D. W. JENSEN, D. F. TOWSLEY, B. LIU, H. JIANG, AND X. WANG, *Online dating recommendations: matching markets and learning preferences*, in Proceedings of the 23rd International Conference on World Wide Web, WWW 2014, New York, NY, 2014, ACM, pp. 787–792.

[139] B. TWARDOWSKI, *Modelling contextual information in session-aware recommender systems with neural networks*, in Proceedings of the 10th ACM Conference on Recommender Systems, RecSys '16, New York, NY, 2016, ACM, pp. 273–276.

[140] G. TYSON, V. C. PERTA, H. HADDADI, AND M. C. SETO, *A first look at user activity on tinder*, in 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM '16, Melbourne, VIC, Australia, 2016, IEEE, pp. 920–924.

[141] R. VAN METEREN AND M. VAN SOMEREN, *Using content-based filtering for recommendation*, in Proceedings of the ECML 2000 Workshop: Maching Learning in Information Age„ ECML 2000, 2000, pp. 47–56.

[142] A. VASWANI, N. SHAZEER, N. PARMAR, J. USZKOREIT, L. JONES, A. N. GOMEZ, L. KAISER, AND I. POLOSUKHIN, *Attention is all you need*, arXiv, (2017).

[143] O. VINYALS, C. BLUNDELL, T. LILLICRAP, K. KAVUKCUOGLU, AND D. WIERSTRA, *Matching networks for one shot learning*, Advances in Neural Information Processing Systems, 29 (2016).

[144] F. VITALE, N. PAROTSIDIS, AND C. GENTILE, *Online reciprocal recommendation with theoretical performance guarantees*, Arxiv.org, (2018).

[145] H.-F. WANG AND C.-T. WU, *A mathematical model for product selection strategies in a recommender system*, Expert Systems with Applications, 36 (2009), pp. 7299–7308.

[146] C. WOTIPKA AND A. HIGH, *An idealized self or the real me? predicting attraction to online dating profiles using selective self-presentation and warranting*, Communication Monographs, 83 (2016), pp. 281–302.

[147] C.-Y. WU, A. AHMED, A. BEUTEL, A. SMOLA, AND H. JING, *Recurrent recommender networks*, in Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM '17, New York, NY, 2017, ACM, pp. 495–503.

[148] F. WU, Y. QIAO, J.-H. CHEN, C. WU, T. QI, J. LIAN, D. LIU, X. XIE, J. GAO, W. WU, AND M. ZHOU, *Mind: A large-scale dataset for news recommendation*, in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, New York, NY, 2020, ACL, pp. 3597–3606.

[149] B. XIA, J. YIN, J. XU, AND Y. LI, *We-rec: A fairness-aware reciprocal recommendation based on walrasian equilibrium*, Knowledge-Based Systems, 182 (2019).

[150] P. XIA, B. LIU, Y. SUN, AND C. CHEN, *Reciprocal recommendation system for online dating*, in Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM '15, New York, NY, 2015, ACM, pp. 234–241.

[151] P. XIA, S. ZHAI, B. LIU, Y. SUN, AND C. CHEN, *Design of reciprocal recommendation systems for online dating*, Social Network Analysis and Mining, 6 (2016), pp. 1–13.

[152] S. XIAO, Z. LIU, Y. SHAO, T. DI, AND X. XIE, *Training microsoft news recommenders with pretrained language models in the loop*, arXiv, (2021).

[153] L. XU, J. XIANG, AND X. YUAN, *Transferring rich deep features for facial beauty prediction*, arXiv, (2018).

[154] R. YAGER AND A. RYBALOV, *Uninorm aggregation operators*, Fuzzy Sets and Systems, 80 (1996), pp. 111–120.

[155] Z. YANG, L. XU, Z. CAI, AND Z. XU, *Re-scale adaboost for attack detection in collaborative filtering recommender systems*, Knowledge-Based Systems, 100 (2016), pp. 74–88.

[156] Z. YIN, T. XU, H. ZHU, C. ZHU, E. CHEN, AND H. XIONG, *Matching of social events and users: a two-way selection perspective*, in World Wide Web, WWW '20, New York, NY, 2020, Springer, pp. 853–871.

[157] R. YING, R. HE, K. CHEN, P. EKSOMBATCHAI, W. L. HAMILTON, AND J. LESKOVEC, *Graph convolutional neural networks for web-scale recommender systems*, in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '18:, New York, NY, 2018, ACM, pp. 974–983.

[158] H. YU, C. LIU, AND F. ZHANG, *Reciprocal recommendation algorithm for the field of recruitment*, Journal of Information & Computational Science, 8 (2011), p. 0.

[159] H.-R. ZHANG AND F. MIN, *Three-way recommender systems based on random forests*, Knowledge-Based Systems, 91 (2016), pp. 275–286.

[160] M. ZHANG, J. MA, Z. LIU, J. SUN, AND T. SILVA, *A research analytics framework-supported recommendation approach for supervisor selection*, British Journal of Education Technology, 47 (2016), pp. 403–420.

[161] S. ZHANG, L. YAO, A. SUN, AND Y. TAY, *Deep learning based recommender system: A survey and new perspectives*, ACM Computing Surveys, 52 (2019), pp. 1–38.

[162] Y. ZHENG, T. DAVE, N. MISHRA, AND H. KUMAR, *Fairness in reciprocal recommendations: A speed-dating study*, in Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization, UMAP 2018, 2018, pp. 29–34.

[163] Y. ZHOU, D. WILKINSON, R. SCHREIBER, AND R. PAN, *Large-scale parallel collaborative filtering for the netflix prize*, in Proceedings of the 4th international conference on Algorithmic Aspects in Information and Management, AAIM '08, Berlin, Heidelberg, 2008, Springer-Verlag, pp. 337–348.