Evolving Communities of Recommenders: A Temporal Evaluation

Neal Lathia, Stephen Hailes, Licia Capra*
n.lathia, s.hailes, l.capra @cs.ucl.ac.uk

February 13, 2008

Abstract

Collaborative Filtering (CF) is the established algorithm that fuels the success of recommender systems. The assumption that these systems rely on is that like-minded users can successfully share the experiences they have had with the content provided; CF is thus a means of spreading information via *similarity*. There are many ways of measuring how similar two users are, and the predictive power of each method has traditionally been quantified by measuring the distance between predictions and the actual ratings provided by users. However, the data that is available to CF researchers is temporal in its nature; it grows as users respond to recommendations and rate items. The temporal nature of the data is highly influential in determining how accurate and useful any particular prediction will be in the *actual deployment* of a recommender system. In this work we perform a full evaluation of CF algorithms that *includes* time, by only considering the currently available information when making predictions. Unfortunately, traditional methods of measuring performance in this context are no longer informative. To overcome this, we introduce a classification of predictions based on *confidence*, and show that both the parameters used to tune CF and the methods used to measure user similarity will have a strong effect on the predictive confidence of the system.

1 Introduction

Recommender systems, responding to the ever growing surge of information on the web, have been designed to act as life vests for online users, who would otherwise be overwhelmed by the vastness of information available to them. Whether this content is movies, music, or the seemingly infinite catalogues of e-commerce web sites, these users are unable to dedicate the time to sift through what is available, or express their requests for new, interesting items in search queries;

*Dept of Computer Science, University College London, London, WC1H 9EB, UK they are confronted with the problem of information overload [1, 2]. Here, the recommender system comes to the rescue, offering items to users based on their profile, or historical behavior (in the form of implicit or explicit ratings). The web is ripe with examples of recommender systems, ranging from Amazon.com's product recommendations[1], to services such as Pandora[2] and Last.fm[3] for music profiling and recommendations, to the Netflix million dollar competition to improve its movie recommender system[4]. The generated recommendations aim at providing each user with a unique, personalised experience that is tailored to fit their taste, that allows users to continue dialoguing with the system as they consume the content that is being offered.

Collaborative filtering (CF) is the dominant community-based algorithmic solution that fuels recommender systems. The community of users, which may include our fictitious protagonists, Alice and Bob, provide the system with the opinions of the (usually limited) experiences they have had with the content. The system collects the set of opinions, and can make predictions of how much Alice will like items she has not encountered and rated already. It does so by combining opinions for these unrated items, following a method built upon the assumption that historically like-minded users will continue sharing similar opinions. Therefore, predictions for Alice will be based on composing opinions of other users who are similar to her, users who form a *neighbourhood* of recommenders for her. This neighbourhood may include Bob, and the contribution he makes to a predicted rating for Alice will be weighted according to how similar he is to her. The methodology described here brands this type of CF as kNN, or k-nearest neighbour, user-user CF. Other methods may compare items rather than users, but all methods aim to produce predicted ratings for items; this way, a ranking of items based on the predictions can be presented to each user as recommendations, in the hope that the predictions will closely match their actual experiences when they rate the content.

One of the main points of research into CF has been on improving the predictive accuracy of the algorithm; excellent examples can be found in [3] and [4]. This focus is echoed in the goals of Netflix prize, which aims at diminishing the current root mean squared error in the predictions by 10%. Striving for evermore accurate recommenders is grounded in the assumption that user ratings express the experiences that the users are having with the content they are consuming. Therefore, the better the system can predict a potential experience, the more likely it is to provide useful suggestions to each user. This assumption not only led to the reduction of the problem of recommending items to that of predicting ratings, but also paved the way for the traditional methods of evaluating the performance of a CF algorithm, based on measuring the distance between predicted and actual ratings.

However, to date there is very little understanding of the effect that improved mean error results will have on users, and it is very difficult to measure the effect

[1]http://www.amazon.com
[2]http://www.pandora.com/
[3]http://www.last.fm
[4]http://www.netflixprize.com

of a CF algorithm on the quality of user recommendations. On the one hand, this is due to incompleteness of current error measures. On the other hand, these evaluations do not include a separate component that is equally important to the unique experience that each user will have with the system: the time when the prediction was made. Evaluations of CF algorithms exclude the fact that the dataset that fuels the recommendation engine is dynamic, growing over time, and offers varying degrees of predictive power to users who interact with the system at different times.

In this work we inject the temporal component into the evaluation of CF algorithms. In particular, we:

- Review the current methods of user-based CF, by exploring the various methdos of measuring user similarity in Section 2, and decomposing the problem of generating predictions into three stages: neighbourhood formation, opinion aggregation, and recommendation with feedback. From this problem description, we can enumerate the assumptions that CF is built upon, which we list in Section 2.4.

- Describe current methods of evaluating a set of predictions in Section 3, and demonstrate that mean error values are not only lacking in information, as they do not consider the extent to which predictions deviate from the mean value, but also do not reflect the actual experience of the end users. We explore this issue by introducing and applying a modified standard deviation function on a set of predictions on the MovieLens dataset[5].

- Introduce the temporal characteristics of the MovieLens dataset, reporting how the dataset changes over time, in Section 4. Based on these temporal characteristics, we repeat rating-prediction experiments that *include* time, to find that the traditional error measures no longer express useful performance information at all. We compliment this analysis with a case study of one of the users in the dataset, in Section 5, where we observe how prediction error evolves over time using a time-averaged absolute error metric.

- Introduce a new measure, based on *confidence*, to observe the extent to which similarity-based CF meets these requirements in Sections 6-7. Measuring the confidence that can be achieved on a set of predictions is dependent on both the similarity measure that is used and the parameters that tune the CF algorithm; a full set of results is detailed in Section 8.

2 Collaborative Filtering

The process that is performed by the kNN user-user CF algorithm that we consider here can be decomposed into three steps: *neighbourhood formation*, *opinion aggregation*, and *recommendation and feedback*. Each component is an

[5]http://www.grouplens.org/taxonomy/term/14

additional step to the formation of recommendations, and also reveals different problems that plague the generation of good recommendations.

2.1 Neighbourhood Formation

The first step of CF entails finding a neighbourhood of recommenders for every user. Drawing from the assumption that similar users will continue to be like-minded, all of the user pairs in the system are compared, according to a predetermined measure of similarity, and each user receives a set of k neighbours. This choice of neighbours determines who is considered to be good sources of opinion information, and consequently both the value k (or size of the neighbourhood) and the method used to measure similarity will have an effect on the power the algorithm has to generate appropriate recommendations.

The simplest similarity measure between two users' profiles, or historical ratings R_a and R_b, can be derived using information that disregards the actual ratings themselves, but considers two other factors. The act of rating an item is a conscious decision made by human users, and represents a judgment on a product that has been "consumed" (viewed, listened to, etc). Furthermore, the mere problem of information overload explains that recommendation systems are built to aide product-selection. Therefore, when two users have rated the same product, they already share a common characteristic: their choice to consume and rate that product. This similarity measure disregards each user's judgment of the item, and weights users according to the proportion of co-rated items [5]:

$$w_{a,b} = \frac{|R_a \cap R_b|}{|R_a \cup R_b|} \tag{1}$$

The more commonly cited similarity measure, however, is the Pearson Correlation Coefficient (PCC) [6]. The PCC aims to measure the degree of agreement between two users, thus including the idea of "how much" a user may have liked or disliked an item. It does so by measuring the extent to which a linear relationship exists between the two users' historical ratings (Equation 2) , by comparing the ratings ($r_{a,i}$, $r_{b,i}$) that users a and b gave for item i, which are normalised with each user's mean rating $\bar{r}_a$, and $\bar{r}_b$. Similarly, the Vector Similarity (VS, Equation 3) aims at measuring the angle that exists between the two user's profiles.

$$w_{a,b} = \frac{\Sigma_{i=1}^{N}(r_{a,i} - \bar{r}_a)(r_{b,i} - \bar{r}_b)}{\sqrt{\Sigma_{i=1}^{N}(r_{a,i} - \bar{r}_a)^2 \Sigma_{i=1}^{N}(r_{b,i} - \bar{r}_b)^2}} \tag{2}$$

$$cos(R_a, R_b) = \frac{\Sigma_{i=1}^{N} r_{a,i} \times r_{b,i}}{\sqrt{\Sigma_{i=1}^{N} r_{a,i}^2}\sqrt{\Sigma_{i=1}^{N} r_{b,i}^2}} \tag{3}$$

The PCC has been subject to a number of improvements. For example, [6] was the first to introduce significance-weighting: a coefficient would be scaled by $n/50$, where n is the number of co-rated items, if the two users had co-rated less than 50 items. This extension is based on the observation that although correlation coefficients demonstrate convergent behavior over time (as they are recomputed with growing profiles), the values it takes when very few items have been co-rated varies wildly. Significance-weighting, in essence, attempts to incorporate a degree of reliability into the coefficient, and, in fact, [6, 7] reported improved prediction results. There are also other heuristics that have been applied; for example, the constrained-PCC uses the rating scale midpoint, rather than the user's mean.

Other methods do exist, and have yet to be fully explored. For example, rather than considering a user's profile to be a vector of numbers, as the above methods do, each entry can be described as a judgement based on an ordinal scale of values [8]. Two users can therefore be compared by constructing a contigency table of agreement, which includes the possibility that the two users may agree by chance [9]. Deciphering the optimal way of comparing judgements is as of yet unresolved, but all of the above methods share the common characteristic that they reduce the relationship between two users into a single numerical value.

This step relies on profile information that the user has given in order to find good sources of opinions. If the user has no historical profile, the *cold-start* problem appears, and it becomes impossible to find any neighbours [10]. Furthermore, if there is no overlap between the user's profile and those of the rest of the community members, no useful comparison can be made, and once again the user cannot be assigned any neighbours: the *sparsity* of CF datasets prevents users from participating in one another's recommendations. One of the proposed solutions is to extend the formation of user neighbourhoods from mere similarity towards the concept of trustworthiness [11]. Trust has been explored from a number of perspectives in recommender systems [11, 12, 13, 14], but continues to have a strong semantic overlap with the purpose of similarity: they are both means of identifying neighbours. In [11] the cold-start problem is overcome by asking users to provide trust scores for other users, thus allowing neighbourhoods to be formed by means of trust propagation. Other solutions, on the other hand, attempt to relieve the problem by looking to exploit available user demographic data when no ratings are available [10].

2.2 Opinion Aggregation

All of the above measures are used to determine *who* will be in each user's top-k neighbours, and to find correlation coefficients $w_{a,b}$, between all user pairs a and b, that contribute to the generation of predicted ratings by acting as *weights* on the opinions received from neighbors [6]:

$$p_{a,i} = \bar{r}_a + \frac{\Sigma(r_{b,i} - \bar{r}_b)w_{a,b}}{\Sigma w_{a,b}} \tag{4}$$

In other words, a predicted rating $p_{a,i}$ for user a's unrated item i will be created by combining the opinions $(r_{b,i} - \bar{r}_b)$ available in that user's neighbourhood, weighted according to the degree $w_{a,b}$ of similarity (or trust) in the neighbour providing the opinion, and then adjusting this weighted mean to fit the user's rating style, by centering them on the user mean $\bar{r}_a$. From this equation we can observe that the ability to make predicted ratings is heavily dependent on the weight and opinions provided by the neighbours found in the neighbourhood formation step.

However, by basing neighbourhood formation on historical profile information, even when neighbours can be found they may not be good sources of information. If, on the one hand, Alice and Bob had the exact same profile, both in terms of items rated and the actual ratings given, then they would have perfect similarity and would most likely appear in each other's neighbourhood. However, equivalent profiles does not imply that Bob will be able to provide any information to Alice. In this cases, with the two profiles being equivalent, he would not be able to contribute to any predicted ratings for Alice. On the other hand, if Bob was the only user to have rated a particular item, but he did not share any ratings with Alice, he could not be in her neighbourhood at all. In both cases, the problem of poor prediction *coverage* arises.

2.3 Recommendation and Feedback

Once predicted ratings have been computed, they can be sorted and the top-n can be presented to the user as recommendations. This step also gives the users a chance to respond to the algorithm, by rating more items and incrementing their profiles. As we will see in Section 4, these steps are repeated iteratively over time and, as user profiles grow, the community of users changes, and the amount of information available to generate predictions becomes richer.

This step of the recommender algorithm has also been subject to analysis. Questions that arise include: what are the incentives for users to rate items [15]? How does presenting information differently influence the *rating trend* for each particular item [16]? Understanding how a user reacts to an algorithm is also an open question and is equally important in order to construct systems that meet the user needs, but is beyond the scope of the work we present here. This last stage completes the *process* that continues cycling in a recommender system, and using this we can take a step back, to outline the assumptions that CF is constructed upon.

2.4 The Goals of Collaborative Filtering

The long-standing assumption of CF states that people who have been like-minded in their past opinions will probably be like-minded in the future. It is interesting to note that, stated this way, even the basic assumptions of information filtering include a temporal component, thus reinforcing the requirement that time play a significant role in the evaluation of CF algorithms. This assumption further implies that a particular user's tastes will not drammatically change from one day to the next; we can safely expect users to display a certain amount of "consistency." Although consistency is a difficult quality to measure, without them the raison d'etre of CF is void. They also imply that the opinions users put into the system will not be appropriate predictors for every other user's view of the same item. In essence, CF aims at capturing implicit relationships within a community of users and items, and use this information to subjectively guide users' choices of which further items to consume.

Given this assumption, what is the goal of CF? In the context we consider here, that of user-user CF, the goal is to create accurate predictions of unrated items, by finding the appropriate neighbours for each user. The principle of like-mindedness can therefore be extended, to form a clear picture of what a successful CF algorithm should achieve:

1. If the set of users in the system is N, then for each user a there is a subset $N_{a,k}$ of k users who are *good* neighbours for a. A good neighbour can be further specified according to three characteristics:

    - *Similarity*: the neighbour's opinions will closely match a's. In other words, there is some measurable quantity between the two user's profiles, and ranking this value compared to the other users will place it in the top-k.
    - *Reliability*: the presence of a neighbour in the top-k will be determined by the similarity measure that is used on the set of profiles. However, all similarity measures share a common characteristic: the reliability that the computed value holds can be determined by how much *profile overlap* there is between the current user and the neighbour.
    - *Participation*: the neighbour can provide information about a's unrated items.

2. A good CF algorithm will be able to quickly find these k neighbours for a; it will match similar users in order to "facilitate getting the right information to the right people." [17].

3. As time passes, a will continuously interact with these k neighbours, by using the opinions they provide to generate predicted ratings of items. Thus a will build a *temporal relationship* with these neighbours.

The main point that emerges from these assumptions is that time is a key component of the process of generating recommendations. As we will explore in Section 4, actual implementations of CF-based recommender systems undergo iterative updates. At each update, all users' neighbourhoods are recomputed using the currently available profiles, which may have significantly grown in the time that has passed since the last update. Therefore, at each update, the algorithm will have more information about different users, and has the opportunity to recompute the decisions it previously made. However, ideally it is able to make the correct neighbourhood decisions as early as possible, and may only have to revert previous decisions when temporal changes to the data (such as new users entering the system) have changed the community to such an extent that neighbours are no longer *similar* and can no longer *participate* in each other's recommendations. Otherwise, if early neighbourhood decisions not only persist, but are reinforced over time, as profiles and the amount of information available to compare users grow, then we know that our decision mechanism is achieving our goals.

The dependence that CF algorithms have on k, the frequency of update, and the method of measuring similarity all emerge from these assumptions. We can also observe that they will all be inter related; finding k neighbours quickly will depend on the update frequency, and populating a neighbourhood of k users will depend on the measure of similarity. In particular, the measure of similarity does not only tell us how much we should weight neighbour opinions, but plays the more important role of telling us whether one user should be favoured over another when being considered as candidates for a third user's neighbourhood.

Given these goals that we set out to achieve, once a CF system is built we require a means of evaluating the extent to which it meets our requirements. In the next section, we explore the current methods that are used to measure the predictive performance of various user centric CF algorithms. We demonstrate that current error measures not only carry very little information about the system's performance, but that, by ignoring the temporal aspect of these systems, also depart from realistically reflecting what users are experiencing.

3 Evaluation Methods

Evaluations of recommender systems can be broadly classified into two groups: interface and algorithmic evaluations. The purpose of evaluating an interface is to make a judgement on how well a system *presents* information to the user, and is thus perhaps the most immediate kind of evaluation that a new system requires [18]. If we were successful in creating the perfect algorithm, but could not present the information to the user in a transparent, understandable, and useful way, then the system still does not fulfill the "life-vest" role we intended for it.

We also require a means of evaluating the underlying algorithm, which can be performed parallel to any interface-level evaluations. Algorithmic evaluation aims at measuring how well a system is *generating* the information that is being provided to the user. To that end, interface concerns are ignored and the evaluation focuses purely on the ratings that users have provided. To this end, the MovieLens dataset has been widely used. The MovieLens dataset is composed of $100,000$ ratings, made by 943 users on 1682 movies. The most popular movie (in terms of rating frequency), with 583 ratings, is Star Wars. There are 142 movies that have only a single rating, but all of the movies have at least one rating. The dataset has been widely used to evaluate CF methods, addressing an equally wide range of research topics; some examples can be found in [19] and [20]. It also is available split into five disjoint 80%/20% training/test sets, ($u1$, $u2$,...$u5$), in order to favour five-fold cross validation.

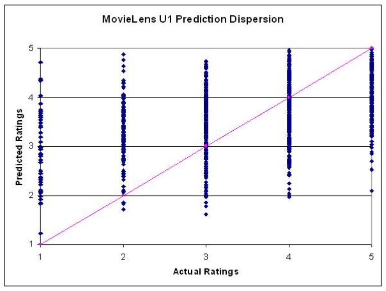


Figure 1: Actual vs. Predicted Ratings

An evaluation of a new CF method will therefore use the training set as the basis, and set any initial values in the system based on the ratings in this set. These may include user-pair similarity measures, which will be used to define each user's neighbourhood, and will determine what opinion information is available. The new method is then asked to make predictions for all the items in the test set, and we can derive the comparative performance of a method to other benchmark results [5]. The aim of this evaluation method is to measure the ability the algorithm has to create useful recommendations, disregarding the accessibility or interface issues that also influence the user's experience. Traditionally, this characterisitc has been measured with either the Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE) between the predicted and actual ratings. Both provide similar insights into the underlying performance, and thus in this work we focus on the MAE.

The quest for accuracy has already been described as a limiting factor in the advancement of recommender systems [21, 22], as it blatantly ignores any consideration of the *order* in which recommendations are given to users and there is no existing work that formalises the extent to which a 10% improvement translates into increased user satisfaction. However, it is grounded in the assumption that ratings express the experiences that the users are having with the content they are consuming, and alternative user-centric methods seem to lack the objective and empirical results offered by statistical error measures. However, reporting a simple MAE does not tell us anything about how much the predictions are dispersed around this mean. In other words, a mean error does not give us very much insight into the actual behaviour of our prediction method. To get an idea of prediction dispersion, Figure 1 shows the results of plotting the actual to predicted ratings of the MovieLens $u1$ subset, when making predictions using all available neighbours and the PCC as the similarity measure. The diagonal line across the plot is where a perfectly accurate CF algorithm would place all the predicted values. This kind of addition to traditional evaluation methods shows the tendency of our algorithm to return certain values; for example, in Figure 1 we can see that the majority of the predicted ratings fall between 2 and 4 stars, and although the overall error is about 0.8, the individual predictions sometimes do much worse than this aggregate value

Table 1: MAE Prediction Error with Standard Error Deviation

Neighborhood	Co-Rated	Weighted-PCC	PCC	VS
1	0.882±1.466	0.874±1.487	0.856±1.384	0.860±1.391
10	0.832±1.378	0.808±1.375	0.877±1.425	0.878±1.430
30	0.789±1.301	0.765±1.287	0.879±1.442	0.887±1.449
50	0.774±1.270	0.753±1.259	0.874±1.444	0.890±1.456
100	0.763±1.249	0.746±1.239	0.853±1.403	0.883±1.439

tells us (especially in the case of items the user rated 1 star).

The first additional piece of information that any evaluation of a CF method requires, therefore, is a measure of dispersion. In other words, how much does the error of each predicted rating spread from the MAE value that we have collected? We can find this value by using a modified version of traditional standard deviation:

$$\sigma = \sqrt{\frac{1}{N}\Sigma_{i=0}^{N}(|r_i - p_i| - MAE)^2} \tag{5}$$

As is visible in Equation 5, we are interested in measuring how much the error in each individual prediction deviates from the mean value that we collected, regardless of whether the prediction was too optimistic (i.e. greater than the user rating r_i) or too pessimistic (less than r_i), or whether the difference is greater or less than the mean value, although perhaps future work may consider differentiating between these outcomes as well. Given this new piece of information, we can re-run our comparative experiments to see if they shed more light on the influence of different similarity measures on the predictive power of our CF algorithm. The results of running experiments on the MovieLens $u1$ subset are shown in Table 1.

Measuring this additional statistic brings to light a new problem. Regardless of the fact that the above report is incomplete without a measure of prediction *coverage*, the standard deviations of the individual ratings from the mean errors are so big that they do not allow us, for example, to differentiate between predictions made using co-rated similarity with 30 and 50 neighbours. Intuitively, using 20 more neighbours should imply that we are using more information to create the predictions. The actual predictions, however, are equally dispersed no matter what neighbourhood size we use; it is no longer possible to measure the effect of the algorithm on the accuracy of the predictions it creates.

We can view prediction dispersion from a different perspective by considering the proportions of predictions we make that fall within certain ranges of error. For example, from Table 1 we can infer that, when using the PCC, the average error $\bar{m}$, over all neighbourhood sizes is 0.867. If we are only considering the MAE values (again, excluding prediction coverage) we would be inclined to say that the PCC prediction method performs much better with 100 neighbours

Table 2: Proportion of (PCC-based) Prediction Error Less than $\bar{m}$

Neighbourhood	$< 0.5\bar{m}$	$< \bar{m}$	$< 1.5\bar{m}$	$< 2\bar{m}$	$< 4\bar{m}$	$< 6\bar{m}$
1	0.320	0.585	0.778	0.897	0.999	1.000
10	0.317	0.578	0.769	0.886	0.998	1.000
30	0.321	0.576	0.768	0.881	0.997	1.000
50	0.323	0.578	0.770	0.880	0.997	1.000
100	0.327	0.588	0.783	0.888	0.998	1.000
Average	0.322	0.581	0.774	0.886	0.998	1.000

than it does with 1. In other words, a higher proportion of the predictions we make will have low error when we use 100 neighbours. We can compare the different proportions of the dataset, or different number of predictions with error that is within various ranges of the average value 0.867, as shown in Table 2. The table shows the various proportion of the test dataset predictions that are made with different amounts of error. The predictions with error less than $\bar{m}$ do not drastically change, regardless of the neighbourhood size we select: none of the values deviate remarkably from the average proportion within the given range. The biggest change we notice over the other results is when 100 neighbours are used, and surprisingly, when a single neighbour is allowed.

Complimenting these results with coverage statistics drastically improves the insight we gain into the performance of the algorithm. When a CF algorithm can not make a prediction, it simply returns the user mean; including unrated predictions in the MAE computation may therefore also skew the mean error value, since CF datasets are known to have a positive distribution, and perform relatively well when simply returning the user mean as a prediction [23]. One possible solution is to exclude uncovered predictions from mean error calculation [24]. However, based on all of these factors, it becomes quite apparent that an evaluation of a CF algorithm that reports mean errors is not very informative or sufficient; it does not satisfy our original goal of capturing the emergent implicit relationships between the users in the dataset. Research in this field has already identified that the user experience is also subject to time, by recognising that the new cold-start users, who do not have a profile of ratings, will not be able to receive useful recommendations. The cold-start problem already implies that the growth of a user profile over time will influence the experience the user has. Therefore, by ignoring the effect of time on predictions, this evaluation method will not reflect the actual experience of participating users.

Can we use this information to make a more informed and reliable evaluation of CF algorithms? We begin addressing this question by considering how the dataset evolves over time.

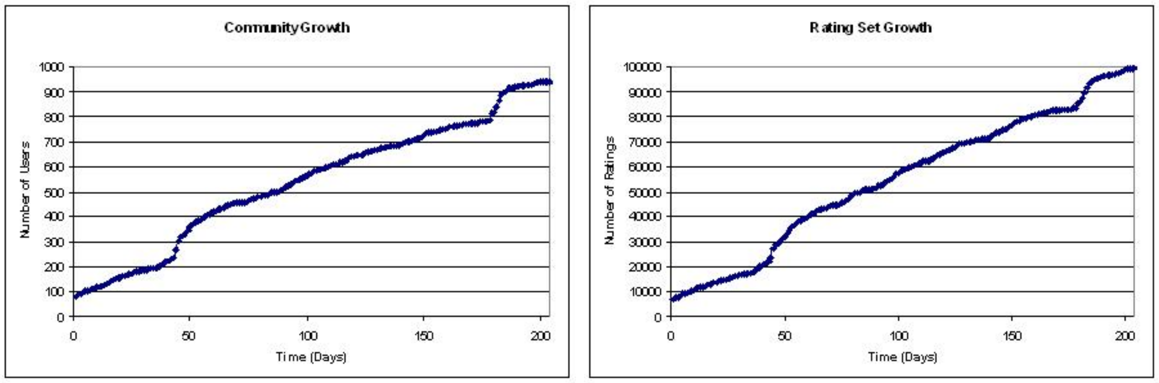


Figure 2: Community and Rating Set Growth Over Time

4 Temporal Characteristics of the Data

The MovieLens rating dataset lists four pieces of information in each entry: the user id, the movie id, the 1-5 star rating given by this user to the movie, and a timestamp, in unix seconds, showing when the user made the rating. The timestamp represents the number of unix seconds since January 1st 1970; a day in this metric is approximately 86400 unix seconds[6].

The first rating in the dataset is marked 874724710, the morning of the 20th of September 1997. The last rating is 893286638, or the evening of the 22nd of April 1998. Submitting the first rating kickstarts the data that a CF algorithm can operate on, and so we refer to this timestamp as time zero. Similarly, the last rating that we have available marks the end of the temporal view we have of this system. The dataset spans a seven month time period, and based on the submitted timestamps, we can observe both the growth of the rating set that is available to generate recommendations and the community that is using the system. In Figure 2 we have graphed the growth of the community over time. Users "join" the community the moment they make their first rating, and so there is a strong coupling between the size of the community and the growth of the rating set over time. The rating set reaches its full $100,000$-rating size at the end of the seven month period, after near-linear growth over time. The community consistently grows over time, without reaching a maximum value until the end, and so is characterised by users who have a wide variety of entry times and constant growth. Community growth over time like this, much like the qualities of a network described by Metcalfe's law, implies that the amount of information available to create recommendations (and thus the *value* that different users can draw from the system) at different times will be quite large. The set of movies that can be rated will also grow over time, but is more difficult to measure, since the dataset does not include time stamps for when a movie was added to the system. We thus consider how many of the 1682 movies remain

[6]http://en.wikipedia.org/wiki/Unix_time

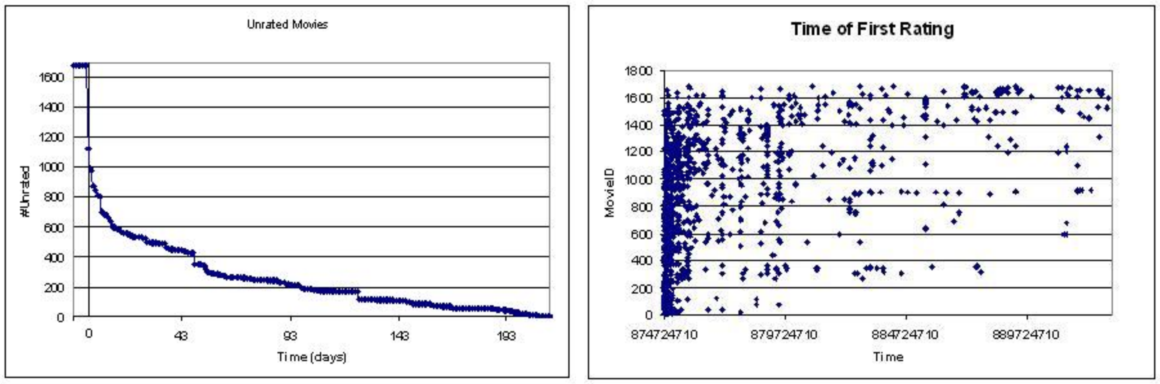


Figure 3: Unrated Movies Over Time, and Time of First Rating Per Movie

unrated over time, in Figure 3. A very large proportion of movies receive their first rating within a very brief interval from time zero; as Figure 3 shows, over half of the set of movies receives its first rating within days. Users in this group are therefore not simply responding to a set of recommendations, but are proactively seeking to rate items, set rating trends, and respond to rating incentives [15, 16, 25], although this activity seems to wane as time passes.

The first attempt at including time in the evaluation of a CF algorithm requires a very small change to traditional methods. Rather than simply breaking a user-rating matrix into training and test components, predictions are made on *all items* based on the user-rating matrix that is available *at that time*. Due to the fact that the dataset has been "cleaned up," by throwing away profiles of users who have rated less than twenty items or have not completed the demographic information and we do not know the exact times when the system was updated, this evaluation will not be a fully accurate reproduction of what happened to the MovieLens users. However, it will reflect making predictions over a dynamic rating dataset, highlight recommender system problems from new pespectives, and call for a new method of measuring the effect of CF algorithm parameters on the quality of the predictions.

4.1 Temporal Evaluation

The system begins at time zero, and will be updated at regular intervals, which could range from daily, to weekly, fortnightly, or monthly. When an update occurs, the system recomputes each user's neighbourhood based on the currently available user profiles, and then derives predicted ratings of unrated items in order to present each user with personalised recommendations. An experiment that mimics this behaviour will recompute user-similarity measures at each interval and then make kNN predictions on all items that the user has yet to rate, based only on information that is currently available. If the user has already rated the item at the current time interval, the predicted value is not updated.

Table 3: MAE Prediction Error, PCC Similarity

Neighborhood	Daily	Weekly	Fortnightly	Monthly
1	2.851±2.007	3.039±1.849	3.133±1.740	3.201±1.670
10	2.859±2.023	3.044±1.862	3.137±1.749	3.205±1.679
30	2.855±2.016	3.041±1.854	3.135±1.746	3.203±1.676
50	2.847±2.001	3.035±1.844	3.130±1.738	3.200±1.669
100	2.836±1.982	3.027±1.829	3.124±1.725	3.194±1.658

The error measures collected from the dataset will therefore be based on the last prediction made of the item before the user rated it, regardless of the time interval between the last prediction and the user rating. The inputs to this kind of experiment are the update interval, the neighbourhood size k, and the method that will be used to derive user-similarity.

We chose to work with the $100,000$-rating MovieLens dataset since it has played a key role in the historical development of CF algorithms. Other datasets exist, such as the $1,000,000$-rating MovieLens dataset and the Netflix prize set, and both not only display different temporal characteristics, but also span a larger time period and include a greater amount of data. The data we used spans a seven month time period, and although this may seem relatively short, it is an extremely long time for users, who may be expecting relevant and accurate predictions from their first encounter with the system.

The first experiment that we performed does as described above, using the PCC as the measure of user-similarity, varying both the neighbourhood size k and the frequency with which the system is updated. The results shown in Table 3 display the accumulated error over the full experiment.

The results could be used to highlight improvements in recommendation accuracy when more frequent updates are performed. This reflects the fact that when users join the system, their profile will be of size zero, and so it will be impossible to find any neighbours for them. In the worst case, they will have joined the system immediately after an update was performed, and so will have to wait the full interval before their profile will be compared to the other community members. The users begin building their profiles over time, but if they have to wait a significant period before they have a neighbourhood of non-zero weighted recommenders.

Unfortunately, these results are absolutely terrible. In fact, we repeated the above experiment and simply returned a random number that is uniformly distributed between 1 and 5, the rating scale used in this dataset, and achieved a MAE of around 1.5 ± 1.3. Why are the results so bad?

5 Case Study: User 407

We explored the effect of temporal evaluations on error measures by restricting the scope of our dataset. We selected user 407, and repeated the above experiments, but only producing predictions on the profile items of the selected user. The user's profile has 226 items and spans a number of days, with variable amounts of growth between updates, thus making user 407 an interesting candidate for inspection. Overall error statistics remain worse than random; for example, if we make predictions using daily updates, a single neighbour, and the PCC, the overall MAE is 1.87 ± 2.57. The aim of our case study was to understand why the error is so high, and to visualise the evolution of the prediction error over time.

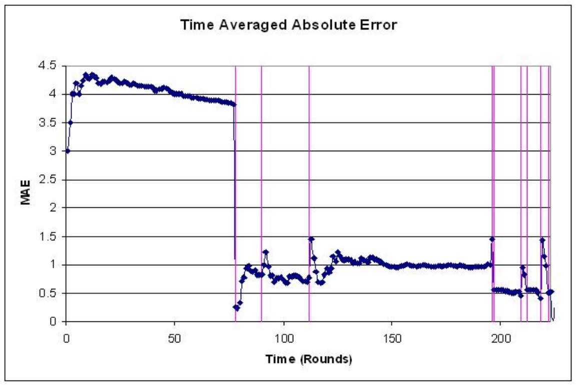
5.1 Time Averaged Absolute Error

To observe the dependence of prediction error on time, we modified the MAE calculation, similar to the Time Averaged Rank Loss that is described in [26]. We first made predictions that include the temporal state of the dataset of all the items in the user profile, and then divided the user's profile into a number of rounds. At each round, the user enters a rating, and so it is possible to calculate the error of the predictions for ratings the user has made until that time. In other words, the prediction error at round N is the sum of the absolute error between all ratings r_i and predictions p_i made up to the current round, divided by the number of rounds:

$$Error(N) = \frac{\Sigma_{i=0}^{N} |r_i - p_i|}{N} \tag{6}$$

We can similarly define the time-averaged coverage, as the number of covered prediction over the total number of ratings done to date. Given that at each update the user's neighbourhood will be recreated using profiles that have grown since the last update, we measured the time averaged absolute error from the moment of the last update. In other words, once an update has been performed, we reset the error and coverage counts to zero. The time averaged absolute error and coverage results for user 407, when daily updates and a neighbourhood of size ten ($k = 10$) is used, are shown in Figure 4. The vertical lines in the figure represent when an update was performed. From the graph, we can infer that the user does not consistently rate items; for example, the number of items that were rated before the first update are far greater than those input before the second.

The graph also gives a reason as to why *overall statistics* of the predictions made for this user are worse than a random guess. Before the first update, the error is incredibly high, and coverage is zero. In certain instances it exceeds 4, which intuitively would have been the upper bound for error for predictions made on a five-star rating scale, and the time averaged error only returns to below 1, within a range that would be expected of a CF algorithm, after the first update. However, even after the third update, when the user's profile exceeds 100 items, the time-averaged coverage does not exceed 70% of the rated items. From this we can form a clear picture of the gravity of the cold-start problem [10] for new users in recommender systems, and the continuous coverage problems



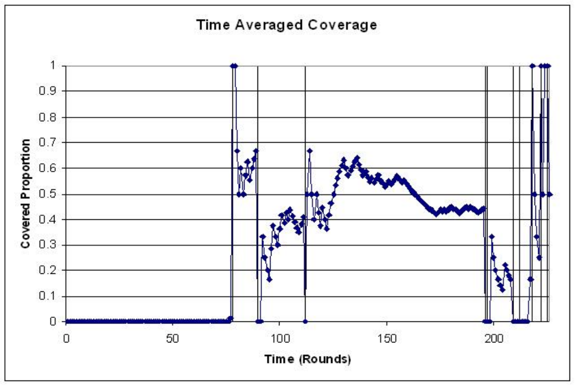


Figure 4: User 407, Time Averaged Absolute Error and Coverage ($k = 10$, Pearson Similarity)

suffered by prediction algorithms over time. Error measures in the cold-start region are so high that they will skew any overall statistics, and can be caused for a number of features of the CF algorithm, which we further explore below.

5.2 Highlighting the Cold Start Problem

As we showed in Equation 4, a predicted rating is generated for a user by collecting opinion information from neighbours and computing a weighted average of the opinions using the similarity values. Opinions are the extent by which a rating deviates from each users' mean, and the reasoning behind this method is that predicted ratings will be more suitable for user 407 by centering them around that user's mean. In a system that is dependent on time, and updated at regular intervals, a number of problems arise with this method:

- First Rater: No opinion for that item. This forms part of the traditional cold-start problem, in that if an item has not been rated by any members of the user's neighbourhood (or perhaps has not been rated by anybody) then it is very difficult to recommend it successfully to anyone else.
- Lack of User Mean. When users joins the system, their rating mean will be zero, unless a number of ratings are required of the user when they join. Any weighted average of opinions that are centred on a zero-mean will produce very high error results. For example, if the weighted average of opinions predicts a rating to be 0.5 less than the user mean, the predicted rating will be -0.5.
- Lack of Recommender Mean. Similar to above, if the only recommender(s) who has rated the item in question has a zero mean, by rating the item prior to any system update, then any deviations from this value become useless; they do not express positive or negative opinions and will distort any prediction made using them.

The relatively straightforward fix to this problem would be to centre rating deviations from the rating scale mid point or overall average user rating, rather than the user mean. This fix improves the error values that are collected from user 407's ratings by injecting numbers that mask the underlying problem. However, these results are still not expressing the full behaviour of the system; they are merely returning the difference between the predicted and actual rating, regardless of whether a prediction was at all possible or not. A closer inspection at the actual progression of user 407's set of ratings reveals that all of the items before the first update are *uncovered*; no prediction could be made. This is due to the fact that user 407's neighbourhood is empty, or full of zero-similarity users, and does not change until the next update is performed.

There are a number of other methods that could be applied in order to reduce the error we report before the first update, in this cold-start region for the user. For example, we could require the user to rate a number of items, whether these be randomly selected or the most popular movies in the system [20], as part of the user's registration process. In fact, users of the MovieLens system that appear in this dataset may have been required to complete such tasks; however, the particular sign-up procedure is not detailed in the dataset. Work has also been done to incorporate data that is external to mere ratings, such as any available demographic information [10]. The choice that is implemented in order to avoid the cold-start problem will have a strong effect on the neighbours that are selected for the user after this bootstrapping process. However, we did not include these methods in our evaluation as they are mostly ad-hoc; moreover, our focus is on evaluating the temporal progress of the system rather than the specific bootstrapping procedure. Furthermore, the successful cold-start solutions also cross into a context-specific domain, since the amount of information that is available to use may vary. For example, MovieLens users were required to input basic demographic data, whereas the Last.fm signup procedure[7] only requires an email address. The one unifying factor is that the success of each method will also be subject to time. For example, requiring users to rate 15 items when they sign up will affect the early-adopters of the system much differently than those entering at a later time. Decomposing the problem as we did above above allows us to see that cold-start does not only affect new users, but will also be a system-wide characteristic, and without implementing one of a number of proposed solutions, will only be relieved with time and proactive users. We leave a full temporal comparison of different bootstrapping methods as a topic to be explored in future work.

6 Returning To The Community

Based on what we have observed in the progression of error in user 407's set of ratings, we can return to attempting to perform an evaluation of the entire community. Figure 5 shows the time averaged absolute error results for the first $5,000$ predictions over all community members, when using 50 neighbours,

[7]http://www.last.fm/join

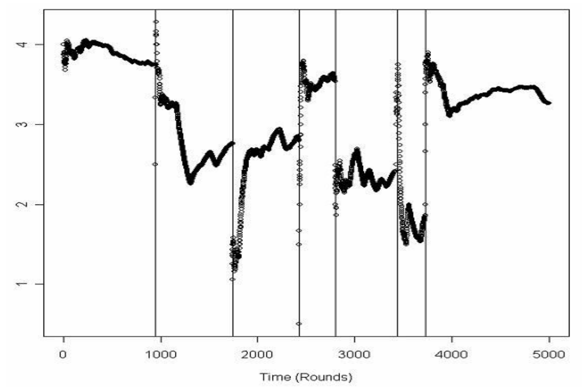


Figure 5: Dataset (first 5,000 ratings) Time Averaged Absolute Error

Table 4: Ratings Made With No Historical Profile

Update Frequency	Ratings	Proportion (%)
Daily	73,384	0.73384
Weekly	80,575	0.80575
Fortnightly	84,468	0.84468
Monthly	87,065	0.87065

the PCC to measure similarity, and daily updates. Unfortunately, the results highlight that the method we used to analyse the evolution of predictions made for user 407 does not hold for the entire community. This is due to the fact that, as we saw in Section 4, the *community itself grows over time*, and new entrants into the system will skew error measures towards what appears to be performing worse than a random guess.

To gain more insight into why such results appear, we can extract the proportion of the dataset that is composed of ratings made when the user had no historical profile. As above, this method will be dependent on the sign-up procedure required of users and the exact system update times; here we are considering the worst-case scenario, i.e. that of users not being required to rate *any* items when joining. We have already seen that user 407 inputs over 50 ratings before the first of the daily updates was performed. How does this affect the entire community? The results are shown in Table 4.

As we can see in Table 4, the proportion of the ratings in the dataset that are based on no history is extremely large. This information can be interpreted in two ways. One the one hand, it reinforces the need to address the cold-start problem, since such a huge proportion of the user experience is not covered by the CF algorithm. On the other hand, it sings the praise of the MovieLens users, who seem to have been actively participating in the rating process, and not simply responding to recommendations [15, 16, 25]. If we simply removed

Table 5: MAE Prediction Error Excluding Cold-Start Items ($k = 50$, Daily updates

Neighborhood	Co-Rated	PCC	Weighted-PCC	VS
1	0.917±1.564	0.917±1.561	0.944±1.644	0.921±1.569
10	0.901±1.525	0.945±1.626	0.894±1.555	0.943±1.616
30	0.869±1.469	0.930±1.599	0.846±1.457	0.936±1.594
50	0.859±1.451	0.901±1.542	0.828±1.419	0.913±1.549
100	0.843±1.422	0.860±1.461	0.818±1.397	0.876±1.475

these ratings from the predictions that we collect performance metrics from, then the numbers return to within ranges of what we would expect of CF algorithms, as we display in Table 5.

There is still no quantifiable difference between using varying neighbourhood sizes. Once again, our error measures do not capture the quantity or appropriateness of the information that was used to generate the predicted ratings, and it is thus very difficult to evaluate the system's performance. We therefore take a step back, and revisit the goals set by CF, in order to construct a new means of evaluating system performance over time.

7 Prediction Confidence

As we have seen above, traditional error measures hide much of the activity that is occurring within the system, and do not reflect the true experience that users will have with the algorithm. Incorporating temporal characteristics into an evaluation to compensate for the lack of realism produces error measures that appear worse than random. However, can we use the temporal progression of the system to our advantage, and use it to measure the implicit relationships that each CF algorithm is generating between the users? We require a classification scheme in order to achieve this, and in particular to divorce the effect of cold-start users from the performance of predictions made with well established user means and neighbourhoods. To do so, we define a grouping of the predictions made in the temporal dataset based on *prediction confidence*.

A prediction is made confidently if it is based on the opinions of *reliably* similar neighbours who have actively *participated* in the previous predicted ratings. In other words, we are measuring how well the algorithm has adhered to the assumptions we put forward when generating predicted ratings, that is, the extent to which the algorithm promotes the formation of temporal relationships with recommenders. For each rating we can derive a level of confidence in the prediction we provided:

$$confidence(a, i, N_{k,i}, t) = \frac{1}{|N_{k,i}|}\Sigma_{k \in N_{k,i}} interactions(k, t) \tag{7}$$

The confidence for the prediction of item i for user a, based on a set of k neighbours in $N_{k,i}$ (who provided opinions about i) at time t is the weighted mean proportion of the time that each neighbour has contributed to a prediction prior to the last update. The reason why this is a weighted mean is due to the fact that we change the value of interactions according to both the amount of information available to derive similarity, and historical participation, by incorporating the overlap (or intersection) of the profiles of users a and k, $R_{a,t}$ and $R_{k,t}$, at time t:

$$interactions(a,k,t) = participation(a,k,t) * \frac{|R_{a,t} \cap R_{k,t}|}{|R_{a,t}|} \tag{8}$$

$$participation(a,k,t) = \Sigma_{i \in R_{a,t}} 1 \text{ if } k \in N_{k,i}, \text{ else } 0 \tag{9}$$

In other words, if a neighbour has perfect profile overlap but can not participate in the prediction, then this neighbour is not part of $N_{k,i}$ and does not influence the confidence in that prediction. Similarly, if a neighbour constantly participates in a prediction but has no profile overlap (there is no measurable similarity), then the confidence in that neighbour's contribution is zero. The highest degree of confidence will come from the top-k neighbours who have historically participated and have a highly reliable measure of similarity.

Our confidence measure does not include the *degree* of similarity the users are deemed to be to their recommenders, but instead incorporates *how much information was used to create the similarity measure*. This is due to the fact that different similarity measures will disagree and return different values when input with the same pair of user profiles [5]. Therefore, in order to not a priori favour one measure over another, we incorporated the feature that is common to all measures: that a greater amount of profile information should return more reliable results [6]. Considerations on the actual value of similarity can be discarded since we are only dealing with the top-k recommenders, and therefore we are only dealing with those who have been selected as the *most* similar to the current user. Our confidence measure therefore captures the fact that confident predictions come from neighbours who have consistently been part of our neighbourhood, reflecting the similarity and participation qualities that we are looking for.

When the system can not find any useful information about a particular item, a state that would lead to an "uncovered" prediction, we say that the system makes a zero-confidence prediction. Similarly, if at the last update user Alice had not provided any rating information to the system, then the system can not make confident predictions for her; confidence will be built over time as she interacts with the neighbours that are provided to her by the CF algorithm.

Given this definition of confidence, we now proceed to measure two characteristics of the system. First, we compute what proportion of the evolving dataset has predictions that were based on any confidence at all. In the case of multiple predictions being made for the same user-item, we only consider the latest prediction before the user inputs the rating, since this is the last value

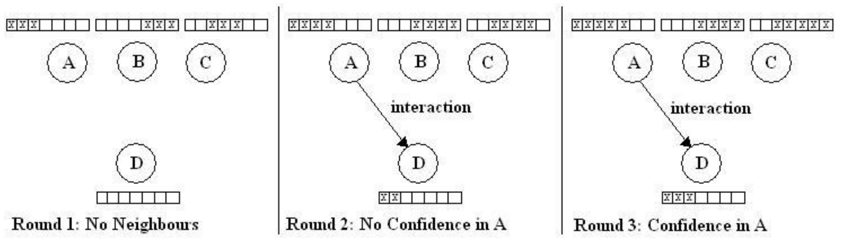


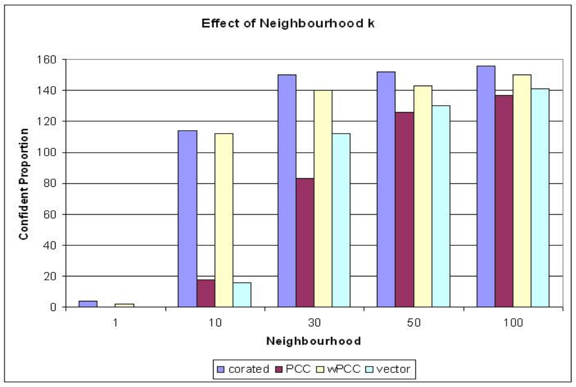
Figure 6: Developing Confidence Example

that will determine whether to recommend the item prior to the user rating it. Following this, we compare the level of accuracy that different similarity measures and methods of generating predictions achieve when making confident predictions.

7.1 An Example

To clarify what we mean by confident predictions we can use a small example. Let's pretend that our CF recommender system has a large number of users, which include Alice, Bob, Cristina, and Dave. We decide, for the sake of simplicity, that each user's neighbourhood should be of size 1, and are interested in finding the best neighbour for Dave. There is no oracle that we can consult to know what the right answer is, but the assumption of like-mindedness tells us that Dave will be more similar to some users than to others.

In the first time interval Dave has no profile; there is no way to make a decision about who his neighbour should be, and he finds himself in the cold-start region of recommender systems. No predictions can be made for him, or, in other words, any prediction that is made for him has zero confidence. At the end of the first time interval, he has rated 10 items, and the algorithm finds that he is most similar to Alice, who becomes his neighbour. In the next interval, she contributes to Dave's predicted ratings 3 times, but since Dave has not interacted with Alice in any previous interval, he still doesn't know whether she is the best neighbour for him, and thus cannot have any confidence in the predictions she contributes to. He also shares only 4 of his 10 ratings with her, and therefore the fact that she is the most similar to him may not be extremely reliable. By the end of the second interval, he has rated 15 items. Alice's profile has grown too, and he now shares 10 of his 15 ratings with her. The algorithm confirms that she will be his neighbour in the next round as well. The third round begins, and now that Dave knows that Alice has already been his neighbour before; predicted ratings that are made for him with her contributions are made with confidence. The value of the confidence will be based on Alice's historical participation (3), and the previous reliability of her presence in his top-k (4/10). If every subsequent round reconfirmed Alice as the best neighbour, and she continuously contributes to Dave's predicted



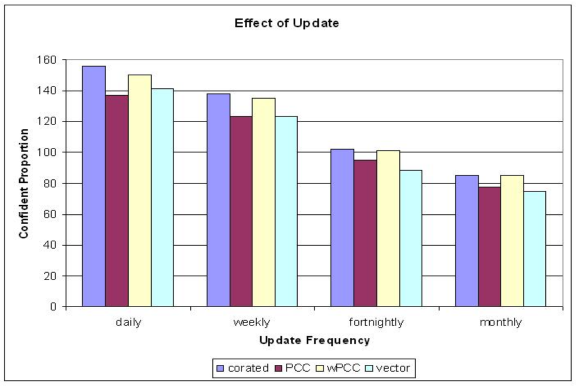


Figure 7: Effect of CF Parameters on Prediction Confidence

ratings, then at each round the level of confidence will increase, as will the overall proportion of confident predictions made for Dave. This interaction is reflected in Figure 6.

However, let's pretend that, in the fourth round, Bob suddenly becomes Dave's neighbour. Dave has never interacted with Bob before and so he cannot have any confidence in Bob's contributions. Building any confidence in Bob's opinions will take time. However, if the first few rounds were simply mistakes due to inaccurate information, or Bob's profile had a later entry into the system and then grew much faster than Alice's (to then overtake her in terms of similarity with Dave), then Bob will be the new best neighbour and he will persist over time in Dave's top-k. The main point here is that if we assume like-mindedness, then if at every round Dave receives a new neighbour (thus making neighbour selection seem like an arbitrary choice), confidence will never blossom. Of course, this is a toy example, but it highlights that like-mindedness should lead to a clustering of users over time.

8 Re-Evaluating Similarity Measures

8.1 Effect of Algorithm Parameters

We can begin evaluating the temporal performance of the CF algorithm by observing the effect of the different parameters on the amount of confident predictions that can be made. The two parameters are the neighbourhood size k and the frequency of updates. In order to consider these parameters separately, in the first experiment we kept the update frequency constant (at daily updates), and varied the neighbourhood size k from 1 to 100. In the second experiment, k was kept constant at 100 and the frequency of updates was changed, from daily to fortnightly, weekly, and monthly. The results are shown in Figure 7.

The results reflect changes we would have expected in the proportion of the dataset that confident predictions are made on as the parameters are tuned.

Table 6: Number of Recommenders Participating in Predictions, $k = 10$

Similarity	0	1-3	4-6	7-9	10
Co-rated	86,086	8,708	3,566	1,451	189
PCC	90,262	8,525	1,105	106	2
Weighted-PCC	79,094	11,891	6,570	2,263	182
Vector	91,348	7,634	900	118	0

Widening the breadth of each user's neighbourhood, by incrementing the value of k, allows the user to have historical interactions with a greater proportion of the community, thus boosting confidence. Similarly, since only historical interactions are considered, and thus only interactions *prior to* the last update are included, then less frequent updates will reduce the amount of predictions we are confident about. It is important to note that if k was the size of the entire community (in this case 943), then the confidence would be the same for all experiments, regardless of the similarity measure that was used. This is because each user would be allowed to interact with everybody else, and thus the role of the similarity measure in deciding which other community members should be neighbours is removed. However, in large-scale deployments of CF systems, it is unreasonable to assume that neighbourhood sizes will be the size of the entire community, as this would mean scaling the system to accodomate for hundreds of thousands of users in each predicted rating.

The results therefore highlight differences between the similarity measures that were explored in Section 2. In particular, both the weighted-PCC and co-rated methods of deriving similarity achieve higher proportions of confident ratings in this dataset, reconfirming their podium positions as succesful similarity measures [6, 5].

The disappointing aspect of these results is the range of values they take. In the best cases, we make just over 120 confident predictions; nowhere close to the proportion of ratings that are made outside of the cold-start region (as was shown in Table 4). Why are confidence levels so low? This must imply that the vision we set of how relationships within a community of recommenders should evolve are not being met. For there to be any confidence at all, we required participating recommenders to persist in a user's top-k from one update to the next. However, in this case, at every update the system is assigning each user a *new* set of neighbours, and very few persist over time. Therefore, it becomes nearly impossible to form long-standing implicit relationships with any recommenders.

We can take a closer look at these results by only considering recommender *participation*. If we revisit the $100,000$ ratings that are within this dataset, using daily updates and a neighbourhood of size 10, we can count how many neighbours provided an opinion in a prediction. As we show in Table 6, different methods of similarity will result in surprisingly disparate results.

The huge proportion of ratings that fall within "0 participating recom-

Table 7: Confident Predictions (Daily Updates)

Neighbours	Co-Rated	PCC	Weighted-PCC	VS
1	4	0	2	0
10	114	18	112	16
30	150	83	140	112
50	152	126	143	130
100	156	137	150	141

Table 8: Accuracy of Confident Predictions (Daily Updates)

Neighbours	Co-Rated	PCC	Weighted-PCC	VS
1	0.627±0.220	N/A	0.514±0.500	N/A
10	0.807±0.614	1.209±0.786	0.866±0.631	1.072±0.728
30	0.912±0.643	0.963±0.663	0.886±0.621	0.968±0.669
50	0.897±0.597	0.915±0.668	0.906±0.642	0.945±0.620
100	0.927±0.604	0.907±0.626	0.914±0.613	0.937±0.607

menders" reflects what we have already explored in Section 5.2, due to the cold-start problem. The table also shows that of the $100,000$ ratings, there are 26 that are made using opinions from the full neighbourhood when the weighted-PCC is implemented. Otherwise, not all of the top-k can provide information for the current item; in other words, they cannot participate in the prediction. The low levels of confidence achieved when using the VS sure are reflected in the fact that a neighbourhood of size 10 never incorporates more information than a neighbourhood of size up to 6 would have. This implies that the vector method does not find neighbours who can contribute to predictions as succesfully as the weighted-PCC and co-rated similarity measures can. If the number of contributing recommenders diminishes, then the ability to create a long-standing interaction relationships decreases as well, and so overall confidence in the predictions we are making falls towards zero.

8.2 Confident Performance

Our confidence measure defines a means of observing extend to which the CF algorithm is promoting predictions based on a history of reliably-measured, participating neighbours. However, when the system does make confident predictions, how well is it making them? To what extent is the system allowing users to build temporal relationships with recommenders who are like-minded?

We can begin by looking at the number of confident predictions that are made, shown in Table 7. Again, the influence of the neighbourhood size k emerges; by allowing users to build relationships with a larger subset of the community, there is a greater chance that repeated interactions occur. The accuracy of each number of predictions, shown in Table 8, falls within the same range of values we observed in Table 5. In other words, when we are confident about our predictions, our accuracy is high, and there is a basis for comparison across different neighbourhood sizes and similarity measures. In particular, when $k = 10$, the co-rated and weighted-PCC similarity measures achieve both a higher proportion of confident predictions *and* improved accuracy. In this work, we only considered confidence on a binary scale, looking at when there was any confidence vs. none at all. It will be interesting to observe how confidence grows over time, and the underlying relationship between different levels of confidence and accuracy.

8.3 Benchmark Results

Now that we have seen the effect of various similarity measures on the kNN CF algorithm, a natural question to ask is: how much confidence is possible within this dataset? In other words, how would other methods, designed to boost prediction confidence, perform on the same dataset? In this work we consider three separate benchmark results:

- **k Nearest Recommenders (kNR)**: Rather than limiting users to their top-k neighbours, allow them to search for the top-k users who can provide information about the current item, thus widening the breadth of recommenders that each user is allowed to interact with.

- **Oracle Recommenders (OR)**: Equation 8 weights the value of interactions according to the overlap between profiles that is shared with each neighbour. This measurement could also be used as a similarity measure itself. Furthermore, we can skip forward in time and pick the top-k neighbours that will be selected with each user's full profile; this benchmark result thus assigns each user a fixed group of recommenders, in order to maximise recommender participation.

- **Oracle Recommenders with Constant Update (ORCU)**: One of the problems we noticed while measuring confidence is that a neighbour's participation is only defined in terms of how many ratings were contributed in previous time intervals. This benchmark result computes confidence based on *any* previous participation, regardless of whether it was in the same time interval or not.

The results are shown in Table 9. Confidence is boosted by not limiting users to interactions with their top-k neighbours, although the gain is not very drastic. Allowing users to only interact with their *oracle* top-k shows little improvement, highlighting the importance of the temporal state of the data in forming confident relations: confidence can not be built if the neighbours that each user must interact with can not participate in recommendations, even if they are computed based on "perfect" information. The largest gain is the oracle with constant update; in this case the confidence does reach the proportions of the dataset that are not in the cold-start region of users. This gain is achieved since confidence is no longer built upon interactions within specific

Table 9: Confident Predictions (Daily Updates)

	kNR				Oracle	
Neighbours	Co-Rated	PCC	Weighted-PCC	VS	OR	ORCU
1	83	51	43	78	83	11,918
10	157	155	158	157	107	21,093
30	158	158	158	157	118	23,183
50	158	158	158	158	126	23,880
100	158	158	158	158	148	24,524

time intervals. Therefore, if after one update a recommender can participate 5 times, then 4 of those predictions will have confidence. Although this deviates from our original definition of confidence, it provides us with a clear upper limit that we can aim to achieve.

9 Discussion

The term confidence has been used before in recommender system research, from the point of view of quantifying the number of ratings that were used to make a prediction, a useful piece of information that can be presented to the users [27]. The main idea is that CF recommenders are decision-support systems, and when the information that is presented to users is augmented from simple predicted ratings to include a confidence metric, then user satisfaction will improve as well. Similar work aims at pushing recommender systems away from a black-box model [28], and in doing so, recognises that if users can understand where their recommendations are coming from they will trust the system more.

The work we did here pushes similar concepts down to the algorithmic level. In fact, CF algorithms mimic how people search for recommendations; by asking others who they trust will have similar opinions to them. CF algorithms thus extend this search by broadening it away from the limitations of a person's social network, to include all the system's participating users. Our definition of confidence is built upon the fact that asking an arbitrary person for a recommendation is not useful, but the utility of the information received will be reflected by the relationship with the (k-limited) subset of neighbours each user interacts with. In fact, this kind of measurement could be pushed back up to the interface level, by, for example, allowing users to see the profiles of the neighbours contributing to their predicted ratings and providing them with a "reputation" value that reflects how often these neighbours have historically contributed to predictions. Although we imagine that this would have a positive effect on the user experience, performing experiments to confirm this point is beyond the scope of our work.

The idea of confidence has also been used in trust management systems research [29, 27, 30]. The use of confidence is similar to that which we apply here; agents (operating, perhaps, in pervasive network scenarios) must make decisions to interact with others based on trust. Computational trust, however, is a value that carries a wide range of uncertainty, and only becomes more reliable as the number of interactions with, and thus the confidence in, the trusted agent increases. Our use of confidence is very similar to this. However, we use confidence as a measurable emergent property of CF algorithms that can be used to evaluate its adherence to predefined assumptions, rather than a subjective value that is used as part of a decision making process.

10 Conclusion

In this work, we have taken a first step toward temporal evaluations of CF datasets. Decomposing the problem of generating predictions into *neighbourhood formation, opinion aggregation*, and *recommendation and feedback* allowed us to outline the assumptions that CF algorithms have been built on. Revisiting current evaluation methods, based on mean errors and coverage statistics, highlighted that evaluations of CF algorithms are not only uninformative and insufficient, but they do not capture the emergent relationships between users that we had required from the outset in our assumptions or reflect the actual experience of participating users. We therefore explored the temporal characteristics of the MovieLens dataset, and, drawing from the behaviour that we expected of our algorithm, were able to define a measure of *confidence* that can be used to evaluate CF algorithms.

Our work with confidence thus far leaves a number of open questions, which can be used by the research community to identify future directions of research. In particular:

- What is the relationship between confidence and accuracy? This work has treated confidence as a binary value, although it was defined on a continuous scale, and therefore the precise relationship between confidence and accuracy was not considered. Understanding the relationship between confidence and accuracy will allow us to gain insight into the original assumptions that we made about CF algorithms.
- How does similarity evolve over time? Understanding how similarity evolves over time will allow us to refine our definition of confidence, by including an understanding of how communities of like-minded individuals form and change as profiles grow.

Although CF algorithms have been previously modelled as “missing value estimation” problems [4], observing the system from the temporal perspective extends previous work that models a CF algorithm as a network of interacting recommenders [31, 5]. CF algorithms are born from the possibility for a community of users to collaborate by sharing their experiences with each other, and thus the emergent properties of the algorithm can be visualised as a graph that links users to each other. The network model has been used to study a variety of scenarios, ranging from spreading of diseases, to human cells and social interactions [32]; in this case we are applying it to the interactions forced upon users by a filtering algorithm. The subjective opinion data that is available from the hundreds of MovieLens users, and indeed all user-rating datasets, holds within it the implicit relationships between items, between users, and the communities that are involuntarily formed by the participants who provide their feedback. The network model has yet much to reveal about the temporal formation of and changes to online communities, and will be the focus of our future work.

References

[1] J.B. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. In *Proceedings of the ACM Conference on Electronic Commerce*. ACM, 1999.

[2] G. Linden, B. Smith, and Y. York. Amazon.com recommendations: Item-to-item collaborative filtering. In *IEEE Internet Computing*, pages 76–80, 2003.

[3] R. Bell, Y. Koren, and C. Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proceedings of the 13th ACM Int. Conference on Knowledge Discovery and Data Mining (KDD'07)*. ACM, 2007.

[4] R. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *IEEE International Conference on Data Mining (ICDM'07)*. IEEE, 2007.

[5] N. Lathia, S. Hailes, and L. Capra. The effect of correlation coefficients on communities of recommenders. In *To Appear in ACM SAC TRECK*, Fortaleza, Brazil, March 2008.

[6] 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*, pages 230–237, 1999.

[7] R. McLaughlin and J. L. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 329–336, 2004.

[8] A. Agresti. *Analysis of Ordinal Categorical Data*. John Wiley and Sons, 1984.

[9] A. Agresti and L. Winner. Evaluating agreement and disagreement among movie reviewers. In *Chance*, 1997.

[10] A. Nguyen, N. Denos, and C. Berrut. Improving new user recommendations with rule-based induction on cold user data. In *Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys)*, 2007.

[11] P. Massa and B. Bhattacharjee. Using trust in recommender systems: An experimental analysis. In *iTrust International Conference*, 2004.

[12] Y. Matsuo and H. Yamamoto. Diffusion of recommendation through a trust network. In *Proceedings of ICWSM2007 Boulder, Colorado, USA*, 2007.

[13] J. O'Donovan and B. Smyth. Trust in recommender systems. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 167–174. ACM Press, 2005.

[14] G. Pitsilis and L. Marshall. A model of trust derivation from evidence for use in recommendation systems. In *Technical Report Series, CS-TR-874*. University of Newcastle Upon Tyne, 2004.

[15] F. M. Harper and Y. Chen J. A. Konstan, X. Li. User motivations and incentive structures in an online recommender system. In *Incentive Mechanisms in Online Systems, Group 2005 Workshop*, 2005.

[16] F. Wu and B. A. Huberman. Follow the trend or make a difference: The evolution of collective opinions. In *Information Dynamics Laboratory*. HP Labs, 2007.

[17] N. Glance, D. Arregui, and M. Dardenne. Knowledge pump: Community-centered collaborative filtering. In *Fifth DELOS Workshop*, 1997.

[18] P. Pu and L. Chen. Trust building with explanation interfaces. In *Proceedings of the 2006 International Conference on Intelligent User Interfaces*. ACM, 2006.

[19] S. Berkvosky, Y. Eytani, T. Kuflik, and F. Ricci. Enhancing privacy and preserving accuracy of a distributed collaborative filter. In *Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys)*, 2007.

[20] A.M. Rashid, I. Albert, D. Cosley, S.K. Lam, S.M. McNee, J.A. Konstan, and J. Riedl. Getting to know you: learning new user preferences in recommender systems. In *International Conference on Intelligent User Interfaces (IUI 2002)*, 2002.

[21] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. In *ACM Transactions on Information Systems*, volume 22, pages 5–53. ACM Press, 2004.

[22] S.M. McNee, J. Riedl, and J.A. Konstan. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *Extended Abstracts of the 2006 ACM Conference on Human Factors in Computing Systems*. ACM Press, 2006.

[23] P. Massa and P. Avesani. Trust-aware recommender systems. In *Proceedings of Recommender Systems (RecSys)*, 2007.

[24] N. Lathia, S. Hailes, and L. Capra. Trust-based collaborative filtering. In *Submitted to IFIPTM*, Trondheim, Norway, June 2008.

[25] K. Ling, G. Beenen, and P. Ludford et al. Using social psychology to motivate contributions to online communities. In *Journal of Computer-Mediated Communication*, 2005.

[26] K. Crammer and Y. Singer. Pranking with ranking. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, 2001.

[27] S.M. McNee, S.K. Lam, C. Guetzlaff, J.A. Konstan, and J. Riedl. Confidence displays and training in recommender systems. In *Proceedings of INTERACT '03 IFIP TC13 International Conference on Human-Computer Interaction*, 2003.

[28] J. Herlocker, J.A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *In proceedings of ACM 2000 Conference on Computer Supported Cooperative Work*, 2000.

[29] L. Capra. Engineering human trust in mobile system collaborations. In *Proceedings of the 12th International Symposium on the Foundations of Software Engineering (SIGSOFT 2004/FSE-12)*, pages 107–116, Newport Beach, California, USA, November 2004.

[30] S. D. Ramchurn, C. Sierra, L. Godo, and N.R. Jennings. Devising a trust model for multi-agent interactions using confidence and reputation. *International Journal of Applied Artificial Intelligence*, pages 833–852, 2004.

[31] F. E. Walter, S. Battiston, and F. Schweitzer. A model of a trust-based recommendation system on a social network. *Autonomous Agents and Multi-Agent Systems*, 2007.

[32] C. Christensen and R. Albert. Using graph concepts to understand the organization of complex systems. *International Journal of Bifurcation and Chaos*, 17(7):2201–2214, July 2007.