Chapter 3

Temporal Analysis of Rating Datasets

The previous chapter highlighted an important problem with recommender systems: CF evaluation does not take into account that the data used to compute recommendations is subject to change over time. In this chapter, we analyse this phenomenon. We begin by introducing the rating datasets we use for this study in Section 3.1. We then split our analysis into two parts: in Section 3.2, we perform an in-depth analysis of the ratings that recommender systems receive. In Section 3.3, we investigate how these changes affect the *similarity* between users over time (and, consequently, the recommendations that CF computes). These observations lay the foundations of work we present in the following chapters; namely, how to use temporal information to improve the accuracy, augment the diversity, and secure the robustness of recommender systems.

3.1 Rating Datasets

We focus on three explicit-rating datasets: two MovieLens sets (which we refer to as ML-1 and ML-2) and the Netflix prize set. These datasets have been at the fulcrum of CF research for a number of years, and can be described as collections of 4-tuples:

$$[u, i, r_{u,i}, t_{u,i}] \tag{3.1}$$

Each tuple contains: a user id u, a movie id i, the rating $r_{u,i}$ given by the user to the movie, and the time $t_{u,i}$ when this rating was input. The motivation behind comparing these datasets extends beyond their popularity. They all provide users rating the same type of *content* (movies) with the same *scale* (1-5 stars)—thus allowing a direct comparison of each dataset's temporal rating characteristics. However, we still expect to identify temporal differences between the sets: there are significant differences in each set's size in terms of users, items, and ratings. We summarise these differences in Table 3.1. In addition to each set's relative size, there are a number of implicit reasons why temporal differences may emerge, including:

- **Motivation**: The MovieLens data is sampled from a system built for research purposes[1]. Netflix, instead, is a commercial system[2] incentivised by financial targets. The system users themselves

[1] http://www.movielens.org/login
[2] http://www.netflix.com/

Dataset	Users	Movies	Ratings	Time (Days)
MovieLens-1	943	1,682	100,000	215
MovieLens-2	6,040	3,706	1,000,209	1,036
Netflix	480,189	17,770	100,480,507	2,243

Table 3.1: Users, Items, Ratings in Each Dataset

have different relationships with each system: in the former, they are *contributing* to research by rating; in the latter, they are *customers* who are requesting and receiving DVDs when subscribed.

- **Interface**: We assume that users are more likely to rate content to which they are exposed; how and what people rate may thus be dependent on the design and usability of each system's interface.
- **Algorithm**: Similarly, we assume that there may be a relationship between what users are *recommended* and what they *rate*. The datasets therefore become subject to the CF algorithm that was in operation when the data was collected.

The rating data alone is not sufficient to understand which of these forces is at play; we are also unaware of any changes to which each system's interface or algorithm may have been subject to during the time span of ratings available. We cannot therefore explicitly discuss the causality of changes we observe in the data. This point is aggravated by the uncertainty as to whether time was taken into account when sampling each system's ratings. The ML-1 documentation states that the dataset has been "cleaned-up:" users with fewer than 20 ratings or incomplete demographic information were pruned from the set. However, there is no further mention of the subsampling technique used.

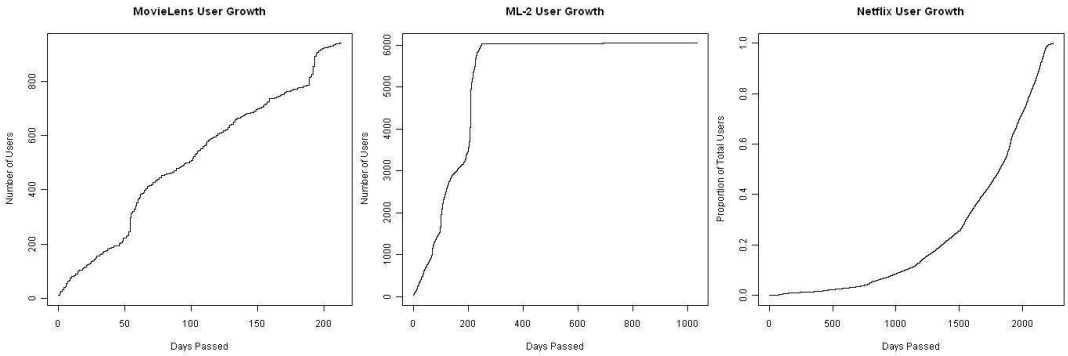
These uncertainties challenge the accuracy of hypotheses that have been verified using these datasets; in particular, it is difficult to claim that algorithms that yield improved accuracy on one of these sets will produce similar results once deployed. However, one point remains: recommender systems are *subject to change over time*, as new users join the system, new ratings are input, and new movies are released. The purpose of the following sections is to show that these changes occur, see how they are visible (in the available data), and examine their impact on conclusions drawn using current evaluative techniques that do not take them into account.

3.2 Ratings Over Time

We divide our analysis into four groups: we look at the growth of the number of users, items and ratings over time (Section 3.2.1), how this growth affects summary statistics derived from the ratings (Section 3.2.2), how user rating behaviour changes with time (Section 3.2.3), and the seasonal trends that emerge when users rate content (Section 3.2.4).

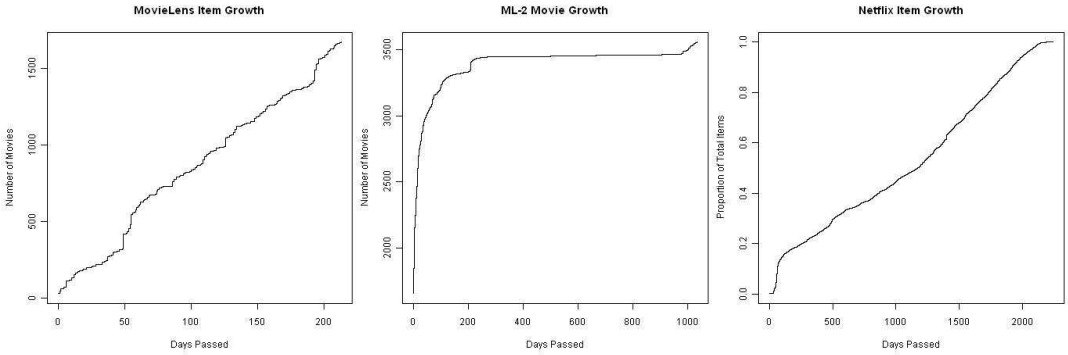
3.2.1 Dataset Growth

In Figures 3.1, 3.2 and 3.3 we visualise the cumulative growth of the number of users, movies, and total ratings over time for each dataset. In these plots we measure *daily* changes, since the Netflix timestamp data only reports the date that users input ratings. The MovieLens datasets' timestamps would allow for a finer grained analysis; however, we opt for daily views in order to consider all three sets simultaneously.



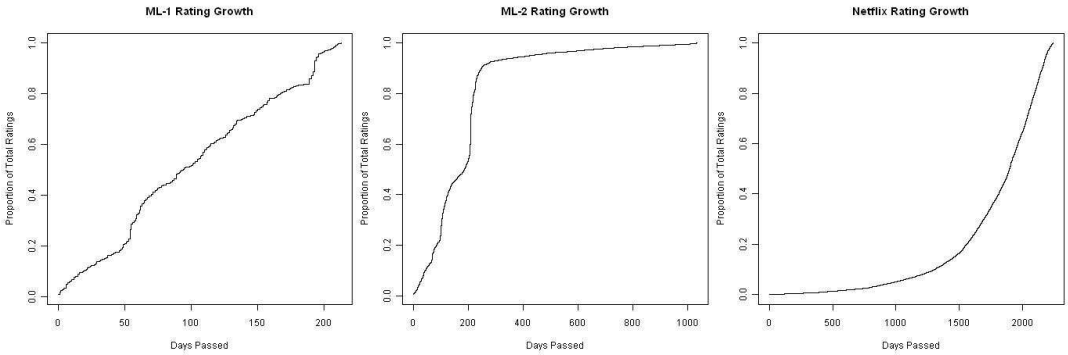
(a) ML-1 Dataset (b) ML-2 Dataset (c) Netflix Dataset

Figure 3.1: Number of Users Over Time (ML-1, ML-2, Netflix)



(a) ML-1 Dataset (b) ML-2 Dataset (c) Netflix Dataset

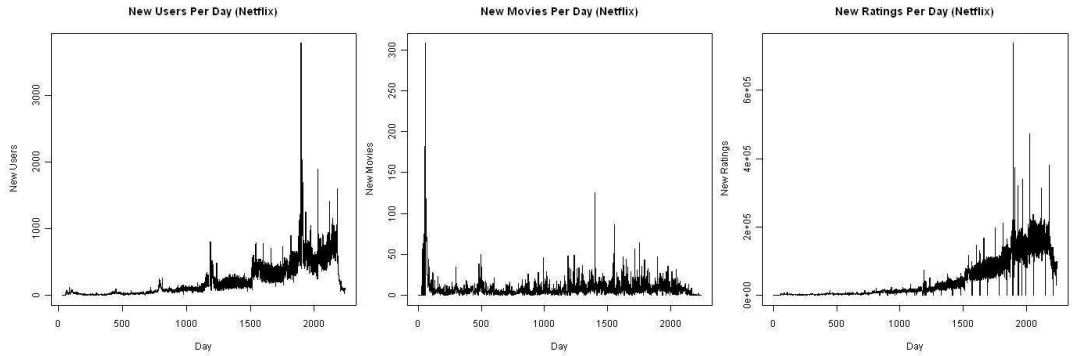
Figure 3.2: Number of Movies Over Time (ML-1, ML-2, Netflix)



(a) ML-1 Dataset (b) ML-2 Dataset (c) Netflix Dataset

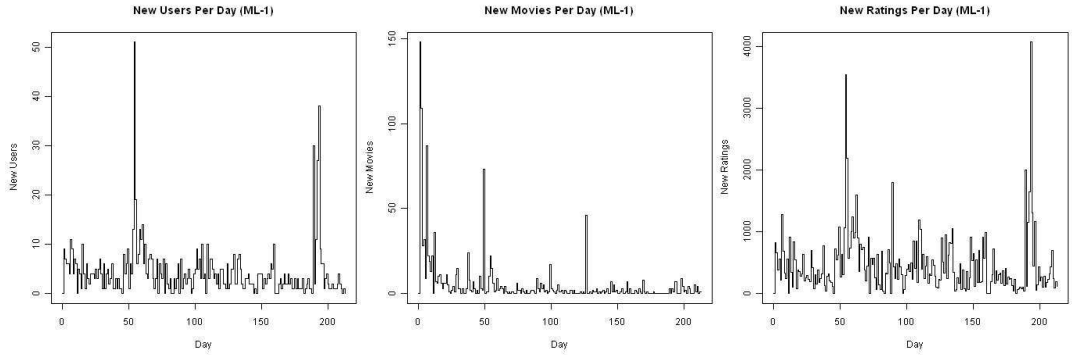
Figure 3.3: Number of Total Ratings Over Time (ML-1, ML-2, Netflix)

Since we do not have sign-up data, we consider that users "join" the system the moment they make their first rating. Similarly, a movie appears in the system when it is first rated, since we do not know when it was actually added to the movie database. We assume this to be a justifiable measure of dataset growth since CF algorithms (that do not include content information) can only compute predictions for movies that have been rated and users that have rated at least once.



(a) New Users Per Day (b) New Movies Per Day (c) New Ratings Per Day

Figure 3.4: Non-Cumulative Netflix Daily Growth: the spikes represent days when a lot of users/movies/ratings were added



(a) New Users Per Day (b) New Movies Per Day (c) New Ratings Per Day

Figure 3.5: Non-Cumulative ML-1 Daily Growth

Each dataset shows varying rates of growth. The number of Netflix users and ratings grow exponentially, while the movies appear in the system at a near-linear pace. The ML-1 set also displays near-linear growth: the number of users, items, and ratings continues to increase over each time step. The ML-2 dataset distinguishes itself from the other two by being the only one that shows a sharp change in growth over time. In fact, the majority of the users appear within the first half of the dataset; after this phase of accelerated growth, user growth halts and the rate at which new ratings and items are added to the system sharply declines. The ML-1 and Netflix sets, instead, do not exhibit this anomaly and continue to grow over time, but appear to do so at different rates.

One of the reasons for this apparent difference is the time that each dataset covers: the ML-1 set, ranging over 215 days, is less than one tenth of the time that the Netflix set (2, 243 days) spans. In order to account for this difference, we examined how much each dataset grows per day. In Figures 3.4 and 3.5 we plot how many new users, movies, and ratings appear in each day of each dataset. From this perspective, the two datasets look more similar (differences between them may be explained by the relative size of each set). Both have peaks, where a large volume of users appears in the system. Similarly, both item plots (Figures 3.4(b) and 3.5(b)) spike in the early days of the dataset, when the

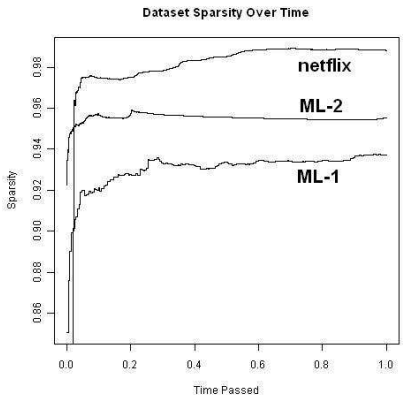
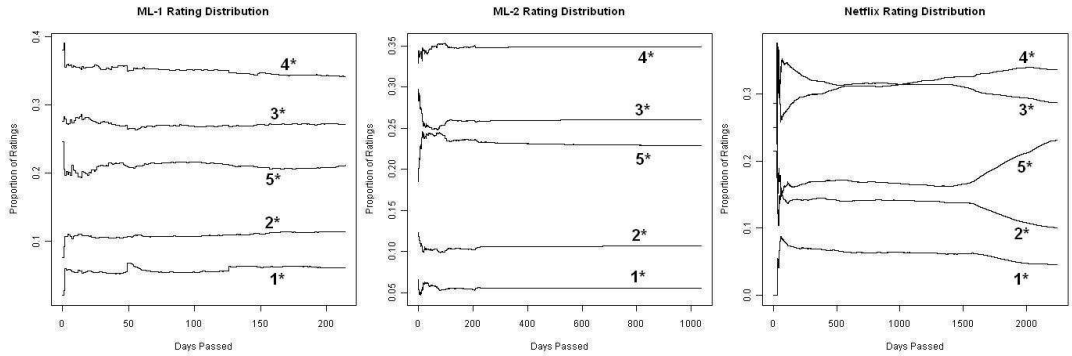


Figure 3.6: Sparsity Over Time For Each Dataset: Netflix is the most sparse dataset

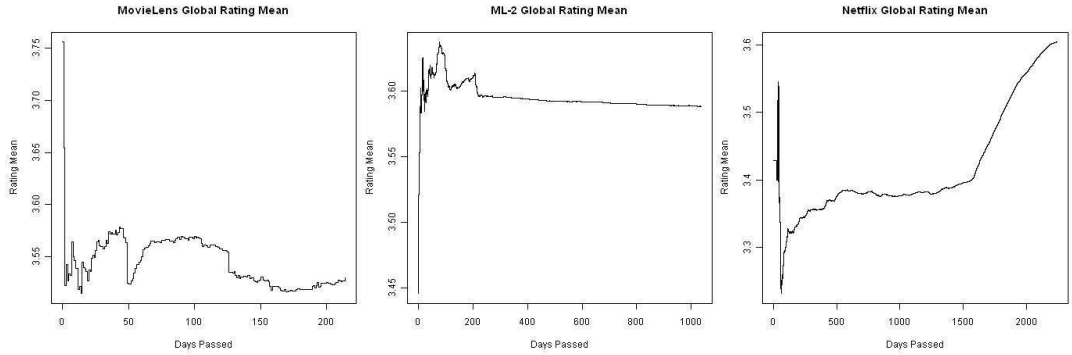
incoming ratings are going to items that have not been rated before. The Netflix data, however, continues to display accelerating growth: Figures 3.4(a) and 3.4(c) shows that the volume of incoming users and ratings tends to increase over time.

A changing volume of users, movies, and ratings will affect each dataset's sparsity and rating distribution. In Figure 3.6, we plot the sparsity over time after normalising the number of days in each dataset. All of the datasets are consistently over 90% sparse—less than 10% of the potential user-movie ratings exist—but the Netflix dataset remains the sparsest, with a maximum value near 99%. The ML-1 set, while being the smallest, is also the least sparse (potentially due to the pruning of users with fewer than 20 ratings). In Figure 3.7, we show how the rating distributions vary with time. If we consider the datasets in their entirety, the absolute ordering of ratings is equal throughout all datasets: there are more 4 star than 3-star ratings, more 3 stars than 5 star ratings, and a very small proportion of 1 and 2 star ratings. This seems to imply that people tend to rate what they already like, but tend to also avoid the "extreme" ratings (1 and 5 stars). However, the Netflix dataset's distribution (in Figure 3.7(c)) changes: in the early days of the dataset, there are more 3 stars than 4 stars. Roughly 1000 days into the dataset, the 4 star rating overtakes the 3 star rating. It seems that, at this point, users are responding more positively to their recommendations; in doing so, they shift the entire distribution of ratings towards the positive end. However, as we do not have data to know what recommendations users were given, we cannot empirically justify this claim. The main conclusion we make from these observations is that viewing rating sets from a static viewpoint does not account for the changes that real systems' data actually undergoes. In particular, user, item, and rating growth over time implies that the amount of information available to create recommendations (and thus the *value* that different users can draw from the system, and potential accuracy) at different times will be quite large. Many users, who rate items that have not been rated before, are not simply responding to recommendations but are proactively seeking to rate items, set rating trends, and respond to rating incentives [HJAK05, WH07a, BLW+04].



(a) ML-1 Dataset (b) ML-2 Dataset (c) Netflix Dataset

Figure 3.7: Rating Distribution Over Time Of Each Dataset: Netflix is the only dataset with no consistent ordering between the rating values



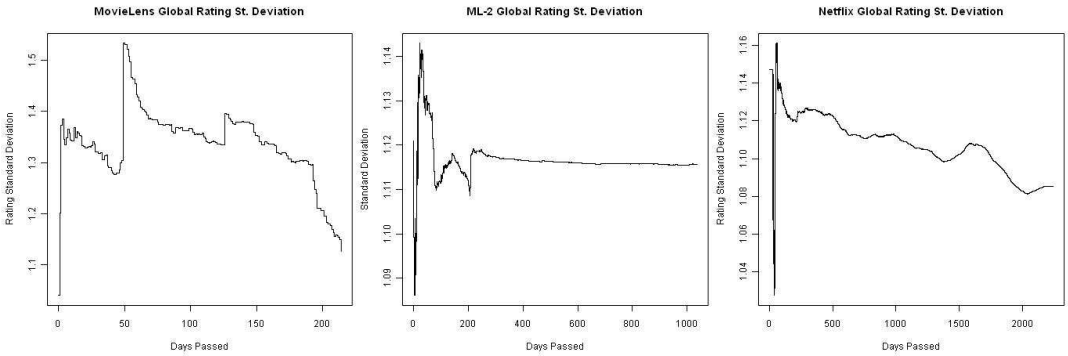
(a) ML-1 Dataset (b) ML-2 Dataset (c) Netflix Dataset

Figure 3.8: Datasets' Global Rating Mean Over Time, Again highlighting the stop in ML-2's growth

3.2.2 Changing Summary Statistics

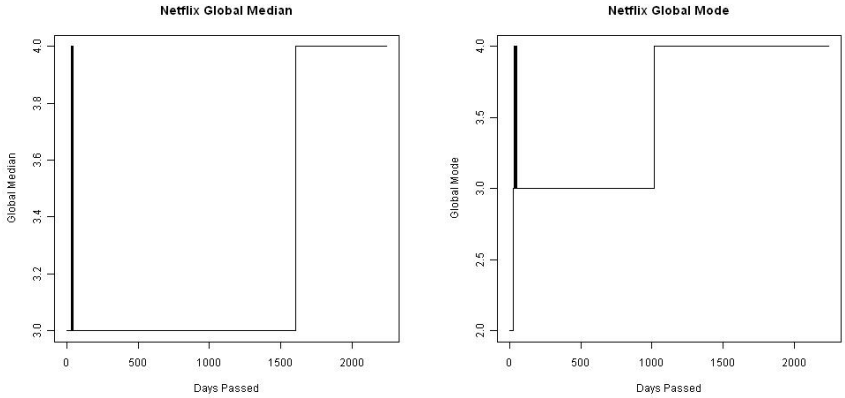
While new ratings are added, any summary statistics computed from the available data may fluctuate. In this section, we consider both *global* and *per-user* or *item* summary statistics. We begin with the global rating means, in Figure 3.8. The means are computed daily using the entire history of available ratings to date; we weight all ratings equally, regardless of when they were input (i.e., there is no time decay). All of the means consistently fall between 3 and 4 stars, but vary quite widely within this range. For the Netflix dataset, the most notable time segments are before the first 500 days, where the global mean rises sharply, falls, and then once again rises, and after the first 1,500 days have passed, where the mean begins to grow again. The ML-2 set (Figure 3.8(b)) emphasises the relationship between *growth* and *change*: when the dataset stops receiving new users (as we saw in the previous section), its global mean stabilises as well.

The standard deviations are shown in Figure 3.9. The Netflix plot (Figure 3.9(c))—like its global mean—suffers from high fluctuation in the initial days of the dataset, and then decreases from 1.14 to 1.08 in a near-linear fashion. In other words, the ratings become less dispersed around the mean over time. Given that the mean is between 3 and 4 stars, this translates to a tendency to rate more



(a) ML-1 Dataset (b) ML-2 Dataset (c) Netflix Dataset

Figure 3.9: Datasets' Global Rating Variance Over Time



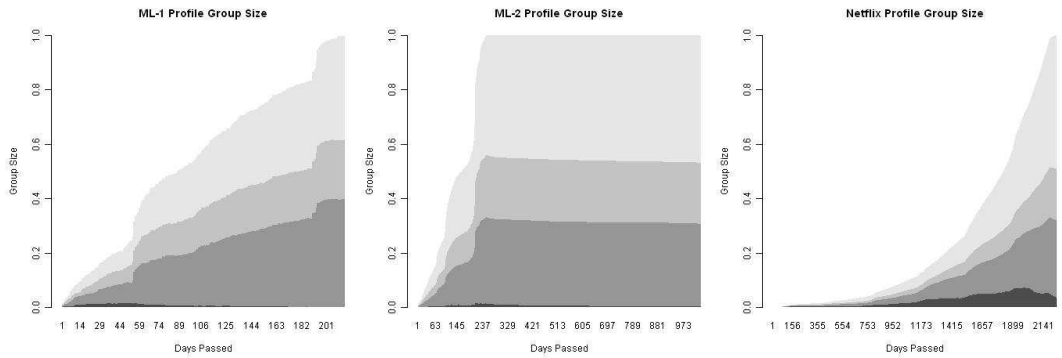
(a) Global Median (b) Global Mode

Figure 3.10: Netflix Rating Median and Mode Over Time

positively. Similarly, the ML-2 standard deviation stops changing when its mean flattens. The ML-1 dataset standard deviation is consistently higher than those we observed in the other datasets; however, excluding the edges of the 215 days, it remains relatively flat. The peak in the plot coincides with the dip in the temporal mean.

The problem here is that state of the art research does not factor in this feature of the data. For example, consider the BellKor solution to the Netflix prize competition [Kor09b]; the foundation of the ensemble of techniques they used successfully to win the competition was a *baseline* predictor, which includes the *global rating average*: a value that, as we have seen, will change over time. Although [Kor09b] does account for temporal changes at the user and item level (by binning the data into sequential windows of varying size), the global baseline prediction is used as a fixed starting value from which to build predictions. If we consider the range of values that this global mean takes over time, it seems that the accuracy of this baseline would vary significantly.

To understand why the mean and variance display such change, consider Figure 3.10, which shows the rating median and mode (i.e. the most frequent rating value) of the Netflix dataset over time. We do not plot the ML-1 and ML-2 temporal medians and modes, since they do not change: they all remain



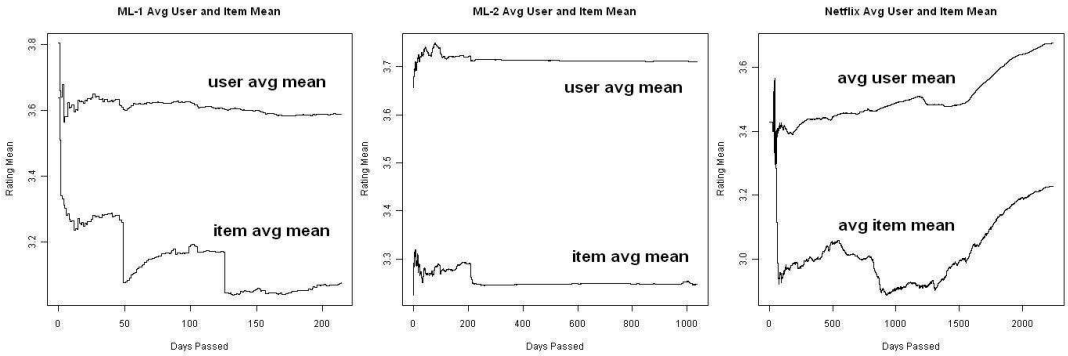
(a) ML-1 Dataset (b) ML-2 Dataset (c) Netflix Dataset

Figure 3.11: Users Binned By Profile Size Over Time

constant at 4 stars. The initial fluctuation in the Netflix mean is mirrored by a change of the rating mode from 2 to 4 stars. The mode then reverts and stabilises at 3 stars, until it again changes, and remains, at 4 stars—accounting for the rise of the rating average. The rating median behaves very similarly to the mode: days after the mode jumps to 4 stars, the median increases from 3 stars to 4 stars, reflecting the surge in the 4 and 5 star ratings that are input in this time, and accounting for the changes observed in both the mean and mode. A median of 4 tells us that *half* of the ratings in the system are 4 and 5 stars; however, more importantly, the *change* the median displays over time reflects that the distribution of ratings does not remain consistent. As above, it is impossible to deduce from the data why the global behaviour changed as we see here; changes to the Netflix interface, recommendation algorithm, user base, or combinations of these may be, but cannot be confirmed to be, the cause.

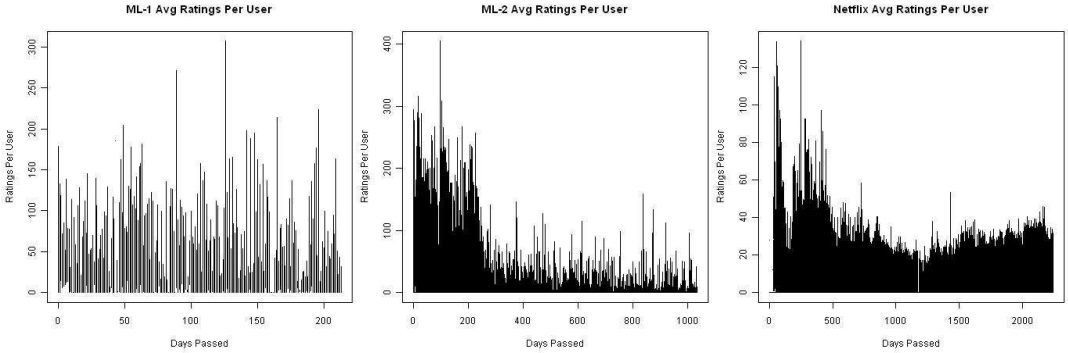
While it is possible to explore CF datasets from a global perspective, it is important to remember that the datasets represent a *collection* of individuals' profiles, and that the global state of the dataset can mask the state and changes that single profiles undergo. For example, as shown in Figure 3.11, if we first split the users into groups according to each user's number of ratings, we can then see how the group sizes fluctuate over time. In Figure 3.11, we bin users into four groups: (black) those with fewer than 10 ratings (excluding those who have yet to rate for the first time), (dark grey) those with $10 - 50$ ratings, (grey) those with $50 - 100$ ratings, and (light grey) those with more than 100 ratings. We then plot the relative group sizes as each dataset grows. These plots highlight the skewed distribution of profile sizes over time. In fact, the group of users with fewer than 10 ratings each may even be under represented in the data, although we do see that the Netflix prize data includes the highest proportion of this group.

The above analysis shows that global summary values fluctuate over time, reflecting how the overall distribution of ratings shifts as more users interact with the system. However, many algorithms that are used for CF do not use global summary statistics, but rather prefer to formulate predictions using either the item or user mean rating values. These values are also subject to change, as we show in Figure 3.12, where we plot the average item and user mean ratings over time. Each perspective (item-based or user-based) of the average means falls into a different range over time. Interestingly, the average *user* mean rating is consistently higher than the average *movie* mean rating; while users tend to rate



(a) ML-1 Dataset (b) ML-2 Dataset (c) Netflix Dataset

Figure 3.12: Average User and Item Mean Rating Over Time



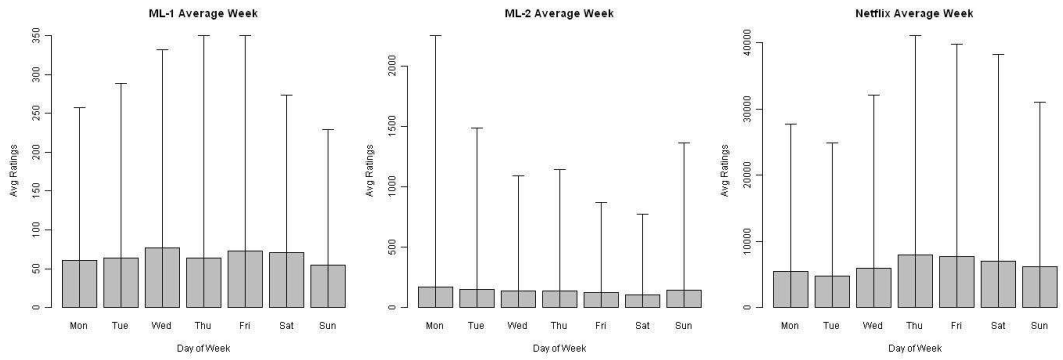
(a) ML-1 Dataset (b) ML-2 Dataset (c) Netflix Dataset

Figure 3.13: Standard Deviation of Ratings Per User Per Day

positively, there are items that are not liked (and thus rated lower), which pulls down the average item mean rating. The datasets not only remain sparse, but also do not stabilise (with the exception of ML-2, which stops growing); recommender systems continuously have to make decisions based on both *incomplete*, *inaccurate*, and *changing* data, and the range of the changes we observe in the Netflix data are likely to have a strong impact on the predictability of ratings.

3.2.3 Temporal User Behaviour

Thus far, our focus has been on the data: how the volume of users, items, and ratings grow and how summary statistics derived from them will change. Since we are dealing with explicit rating datasets, the mere act of rating also reveals how users are interacting with the system. To explore how user behaviour will vary over time, we plotted the standard deviation of the number of ratings input by returning users (i.e., users who have previously visited the sytem and input ratings at least once) per day in Figure 3.13. The plots show the high variability in how users interact with the recommender system. Both the ML-2 (Figure 3.13(b)) and Netflix (Figure 3.13(c)) datasets have high initial fluctuation in average user ratings per week; following this, the mean value flattens out. The ML-1 dataset, instead, has a more steady stream of average ratings per user, with small peaks corresponding to days that users (on average) rated more. Both of the MovieLens datasets have a much higher dispersion—many of the bars are over 100—



(a) ML-1 Dataset (b) ML-2 Dataset (c) Netflix Dataset

Figure 3.14: MovieLens: Average Number of Ratings Per Week (With Standard Deviation)

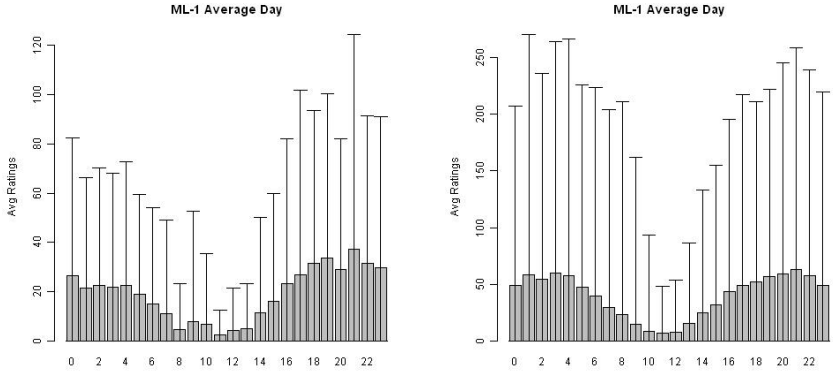
while the Netflix data (after the initial high period) falls below 50: it seems that users are proactively rating more in the MovieLens system.

3.2.4 Daily and Weekly Trends

In the previous section, we observed how user rating behaviour fluctuates over time, by looking at the entire window available for each dataset. However, this same rating behaviour can be further summarised by relating it to the day of the week when the ratings are input. In Figure 3.14 we plot the average number of ratings input per day for each dataset. Netflix sees its highest activity at the end of the week, as more ratings tend to be input on Thursdays, Fridays, and Saturdays than the other days. However, as the two MovieLens datasets show us, these results are again dependent on the subset of ratings available in the dataset: both ML datasets come from the same *system*, yet display very different rating activity. ML-1 rating trends tend to be lower during the weekend, with most ratings being input Wednesdays-Fridays, while the ML-2 dataset shows us the opposite, where more ratings are received on Mondays than any other day of the week.

Since the MovieLens timestamps allow us to know the precise moment when each rating was submitted, we can extract a finer-grained view of user activity over an average day in the system. Instead of binning ratings according to the day they were input, we binned them by hour, and plot the results in Figure 3.15. Unlike Figure 3.14, the two datasets now show very similar activity patterns: users tend to rate movies in the evenings, and the lowest volume of ratings appear roughly between 8am and 3pm; we assume this may be the cause since the majority of the system users would otherwise be occupied at work during these hours.

This analysis reflects an important aspect of recommender systems: the data is being produced by *people*, who tend to exhibit regular patterns of behaviour. The fact that people are behind the data input process also bounds the number of ratings we can expect to be input by a single person in a particular period of time. For example, it seems unlikely for a person to be able to rate 100 movies in less than a minute; moreover, if they were able to input this number of ratings, we could question the extent to which this person is providing honest (and thus, not noisy [APO09]) values. We will revisit this result and use this conclusion when we address the problem of recommender system robustness (Chapter 6).



(a) ML-1 Dataset (b) ML-2 Dataset

Figure 3.15: MovieLens: Average Number of Ratings Per Hour (With Standard Deviation)

Regardless of how collective behaviour changes over time, the focus of a recommender system is to harvest user ratings in order to then generate personalised recommendations for each user. As we have seen before, this operation relies on the assumption of persisting *like-mindedness* between users. In other words, changes to the data over time are only important if they affect the quality and accuracy of ranked recommendations. We begin to explore this facet in the following sections, where we investigate the extent to which measurable similarity persists over time.

3.3 Similarity Over Time

The various algorithms that have been applied to collaborative filtering contexts operate in different ways, but all focus on capturing the *similarity* between users or items as content is rated. For example, nearest-neighbour algorithms focus on similarity by using explicit similarity metrics and making predictions with the most similar items (or users), and factorisation methods project item pairs into feature spaces where the similar pairs will be near one another. In this section, we explore how measurable similarity changes over time. In Section 3.3.1, we redefine the similarity metrics on which we will focus. We then look at similarity from two perspectives: the *static* case (Section 3.3.2), allowing us to visualise the effects of different similarity weights, and the *temporal* case (Section 3.3.3), which explores how similarity changes over time and the consequences of it doing so.

3.3.1 Similarity Measures

We focus on three metrics: the Pearson Correlation Coefficient (PCC), the Vector (or Cosine) Similarity, and the Jaccard distance. The simplest similarity measure between two user profiles—the Jaccard distance—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.). Therefore, when two users have selected 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:

$$w_{a,b} = \frac{|R_{a,i} \cap_i R_{b,i}|}{|R_{a,i} \cup_i R_{b,i}|} \tag{3.2}$$

The Cosine similarity measure works by comparing the intersection of two users' profiles as vectors of ratings:

$$w_{a,b} = \frac{R_a \bullet R_b}{||R_a||||R_b||} = \frac{\sum_i r_{a,i} \times r_{b,i}}{\sqrt{\sum r_{a,i}^2}\sqrt{\sum r_{b,i}^2}} \tag{3.3}$$

The PCC aims to measure the degree of agreement between two users by measuring the extent to which a linear relationship exists between the two users' historical ratings [HKBR99].

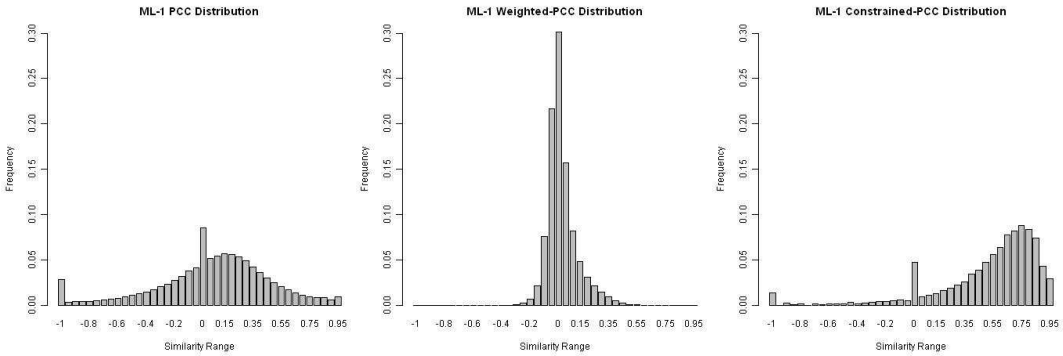
$$w_{a,b} = \frac{\Sigma_{i=1}^{N}(r_{a,i} - \bar{r}_a)(r_{b,i} - \bar{r}_b)}{\sqrt{\Sigma_{i=1}^{N}(r_{a,i} - \bar{r}_a)^2 \Sigma_{i=1}^{N}(r_{b,i} - \bar{r}_b)^2}} \tag{3.4}$$

We also include two variations of the PCC—the *weighted* PCC, where users who have co-rated n items (fewer than a threshold value $x = 50$ [HKBR99]) have their similarity scaled by $\frac{n}{x}$, and the *constrained* PCC, where user ratings are normalised with the rating scale mid point (2.5 stars) rather than each users' mean rating—making a total of five similarity measures.

3.3.2 Static Similarity

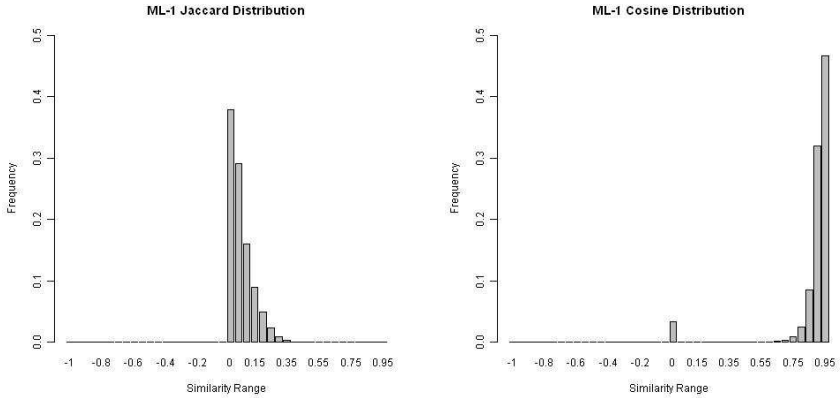
The intuition behind similarity metrics is that if they are well-suited to the problem at hand (i.e., finding good neighbours for users or items) then they will lead to better kNN predictions and, as a consequence, better recommendations. However, there is a problem with similarity measures that is best demonstrated with an example. If Alice's rating history for five items, on a five-point rating scale, is $[2, 3, 1, 5, 3]$, and Bob's rating history for the same items is $[4, 1, 3, 2, 3]$, then the Cosine similarity will be about 0.76. The PCC will return -0.50, while adding significance-weighting will produce -0.05. Other methods will result in equally different values. There is no consensus between the different methods as to how similar Alice and Bob are. Just as the relationship between Alice and Bob will change from good to bad depending on how they compute their similarity, selecting different coefficients will alter the weightings of all the user-pairs in the community. The relative ordering of similarity will also change: given three users (a, b, c), with $w_{a,b} < w_{a,c}$ when using the PCC does *not* imply that $w_{a,b} < w_{a,c}$ will remain true when using the Cosine similarity. The similarity values will, in turn, affect the prediction accuracy and coverage of the CF process.

We investigated the nature of these different similarity measures by looking at their distribution over the full range of available neighbours in the MovieLens-1 dataset. We focus on this dataset since, as we found in Figure 3.6, it is consistently the least sparse; similarity values derived from this dataset are thus assumed to be more reliable. We first computed all the coefficients between every pair of users, using all available profile information. We then plotted the proportion of the total number of coefficients that fell within a given range (in bins of size 0.05) to be able to see how these coefficients are shared out among all the available user pairs in Figure 3.16 and 3.17. The PCC distribution has two interesting peaks: one in the range of $(0, 0.05)$, and the other between $(-1.0, -0.95)$. In other words, a relatively high proportion of coefficients fall between the two ranges covered by these points: many users are either not similar or



(a) PCC Distribution (b) Weighted-PCC Distribution (c) Constrained-PCC Distribution

Figure 3.16: ML-1 PCC, Weighted-PCC & Constrained-PCC Similarity Distribution



(a) Jaccard Distribution (b) Cosine Distribution

Figure 3.17: ML-1 Jaccard & Cosine Similarity Distribution

very dissimilar to one another. Applying significance weighting to the coefficient changes the distribution drastically, by increasing the frequency of neighbours who have very low correlation. Nearly half of the user pairs are valued within $(0, 0.05)$, which implies that a high proportion of recommendations are weighted extremely lightly. The constrained-PCC skews the entire distribution toward the positive end; it seems thus that this variation of the PCC will increase the similarity between pairs of users that may otherwise have been deemed minimally similar with the standard PCC.

On the other hand, the similarity distributions based on the Jaccard distance peaks at 0, for the number of users who do not share any rated items. The rest of the user-pairs all share a positive similarity. Since this coefficient is derived using the number of co-rated items that the user-pair share, this coefficient cannot be negative, and thus a community of recommenders in this scenario will only have positive links. The Cosine distribution had the largest number of coefficients within a very high range: 0.78, or nearly 80%, of the community is weighted between 0.9 and 1.0. This is the result of summing the proportion of coefficients between $(0.9, 0.95), 0.32$, and $(0.95, 1.0), 0.46$. In other words, vector-similarity weights will favour neighbour recommendations much higher than, for example, the Jaccard distance. Finding that the majority of the population share similar coefficients may imply that the population is

neighbourhood	Co-Rated	PCC	Weighted-PCC	R(0.5, 1.0)	R(-1.0,1.0)	Constant(1.0)
1	0.9449	1.1150	0.9596	1.0665	1.0341	1.0406
10	0.8498	1.0455	0.8277	0.9595	0.9689	0.9495
30	0.7979	0.9464	0.7847	0.8903	0.8848	0.9108
50	0.7852	0.9007	0.7733	0.8584	0.8498	0.8922
100	0.7759	0.8136	0.7647	0.8222	0.8153	0.8511
153	0.7725	0.7817	0.7638	0.8053	0.8024	0.8243
229	0.7717	0.7716	0.7679	0.7919	0.8058	0.7992
459	0.7718	0.8073	0.8025	0.7773	0.7812	0.7769

Table 3.2: MAE Prediction Error, MovieLens u1 Subset

Dataset	Co-Rated	PCC	Weighted-PCC	R(0.5,1.0)	R(-1.0,1.0)	Constant(1.0)
u1	0.7718	0.8073	0.8025	0.7773	0.7812	0.7769
u2	0.7559	0.7953	0.7903	0.7630	0.7666	0.7628
u3	0.7490	0.7801	0.7775	0.7554	0.7563	0.7551
u4	0.7463	0.7792	0.7747	0.7534	0.7554	0.7531
u5	0.7501	0.7824	0.7784	0.7573	0.7595	0.7573
Average	0.7548	0.7889	0.7847	0.7613	0.7638	0.7610

Table 3.3: MAE Prediction Error For All MovieLens Subsets

full of very similar users, but following this same analysis using the PCC yielded quite opposing results. Once again, we found that the distribution given by each similarity measure does not agree with any of the others. There does not seem to be any unifying behaviour or descriptive characteristics, in terms of coefficient distribution, of the dataset, as the method for computing the coefficients is varied.

Any attempt at finding the "best" user weighting, to date, can only be done by conducting an analysis on comparative results of different techniques applied to the same dataset of user ratings; there is no way of measuring how close these algorithms are to an optimal answer. We can, however, produce a worst-case scenario: we construct a similarity matrix based on *random values*, and observe how accurately this scenario can generate predicted ratings. Random-based similarity does not use any information from the dataset to find like-minded peers; it simply is a set of uniformly distributed random values on a pre-defined range. We thus expected that the error reported on the prediction set would be devastatingly worse than when any similarity measures were used, since use of random numbers does not consider how much users have co-rated items or how much their ratings agree with each other.

In order to see how accurate predictions are with different similarity metrics, we measured the mean absolute error (MAE) of the predicted ratings *only* in the case when a prediction was made. If no information was available, typical experiments will simply return the user mean, and this value is not used when finding the MAE of the predictions. Since MAE measures the mean absolute deviation from the actual ratings, and the MovieLens dataset uses a five-point rating scale, the error measures can be expected to fall between 0, or perfect prediction, and 4.

We experimented with three ranges of random-similarity: $(-1.0, 1.0)$, or randomly assigning re-

lationships so that the distribution of coefficients over all user pairs is uniform over the full similarity scale; $(0.5, 1.0)$, i.e. giving all the user-pairs high similarity relationships; and all 1.0, giving all user pairs perfect correlation.

Table 3.2 shows the prediction error results as k is increased, when using a subset of the MovieLens data (named $u1$). However, as we have seen, prediction results are dependent on the data that is being used. We therefore cross-validate our results by averaging the prediction error across five subsets of the ML-1 dataset (named $u1, u2, u3, u4, u5$). The most accurate results were obtained when predicted ratings were derived using all of the community members' ratings; Table 3.3 shows the prediction results for all subsets, when using this value.

To our surprise, the results of the experiments using random-valued and constant relationships were not only comparable to the performance of the correlation coefficients, but on average they also performed slightly better than the tested similarity measures. Such results would be expected if there were a certain degree of homogeneity amongst the community members, regardless of whether the specific correlation values agreed or not. A simple popularity-based recommender, which returns the average rating of an item (using all available ratings of it) also produces comparable performance. The average MAE over all data subsets, in this case, is 0.8182, which is 0.04 less than the weighted-PCC's accuracy.

The datasets may be to blame for the results; they may be too small, or not representative enough of a heterogeneous set of users. The MovieLens dataset we used does comply with the "long-tailed" characteristic of user-ratings; however, little more is known of what qualifies a rating dataset as appropriate. Repeating the above experiments with the Netflix dataset produced different results. Nearly all predictions were not covered, since randomly assigning neighbours to each user did not produce useful neighbourhoods. However, if we tune the experiment to account for the larger dataset size by selecting neighbours randomly from the pool of users who have rated the item that needs to be predicted, we again see similar results to the above. These results are another sign that the dominant error measures used to compare collaborative filtering algorithms may not be sufficient. Traditional similarity-based kNN cannot be differentiated from the output of random-similarity kNN. The results further highlight the fact that the current similarity measures are not strong enough to select the best neighbours. In the following section we will see that this result persists over time.

3.3.3 Temporal Similarity

An analysis of the distribution of correlation coefficients in the community of recommenders may, at first glance, seem inappropriate, since the coefficient values will change over time, as they are recomputed with growing user profiles. In this section, we examine *how they got there*, by looking at how similarity between users changes over time.

A useful means of analysing how similarity changes over time is to consider the act of computing similarity between all users as a process that generates a graph. In this case, each user is a node. Links to other nodes are weighted according to how similar the user-pair is, and (in the case of kNN prediction) the algorithm imposes the restriction that each node can only link itself to the k most similar neighbours; the out-degree of each node is limited. From this perspective, similarity graphs are a *constrained implicit*

social network between the users in the system. The network is *implicit* since the users are not actively involved in selecting who they want to link to, and is *constrained* since the k parameter places an upper bound on the number of neighbours each user can have.

Observing similarity computation as a graph-generating process paves the way for a wide range of analysis that can be performed on recommender systems, drawing from methods described in graph theory and previous work on (explicit) social network analysis [BA02, MAA08]. The aim of analysing the graph generated by a filtering algorithm is to understand how the rating data is being manipulated in order to derive predictions. Furthermore, iterative updates of a recommender system can be viewed as re-generating the user graph. Changes in the graph when updates are computed will highlight how these systems perform over time, and give insight into why the parameters and methods that can be used to produce different accuracy results.

In the following sections, we explore the emergent properties of dynamic, temporal user-user similarity graphs, by decomposing the analysis into four separate stages:

- **Node Pairs**: Drawing from the growth of both nodes and rating information, we explore how similarity between a pair of nodes evolves over time. This analysis allows us to classify similarity measures into three groups, based on how they evolve the relationship between a pair of nodes: incremental, corrective, and near-random measures.
- **Node Neighbourhoods**: We have already mentioned that a kNN algorithm imposes restrictions on the graph, by allowing nodes to point to a pre-defined number of neighbours. We project this restriction onto the temporal scale, and observe the volatility of user neighbourhoods as profiles grow and similarities are re-computed.
- **Community Graphs**: The last section of our analysis considers the entire community of users. We computed properties such as connectness, average path length, and the in-degree distribution of links, to find that similarity graphs display the small-world, scale-free characteristic that is common to social networks. In other words, CF algorithms intrinsically favour some users over others; we refer to these as *power users*, and perform experiments that aim to collect the influence they exert on the predictive accuracy of the kNN algorithm.

User Pairs Over Time

Based on the way the datasets change over time, we first turn our attention to how the relationship between a *pair* of nodes will evolve. The primary concern of collaborative filtering, based on the user profiles explored above, is to predict how much users will rate items, in order to offer the top-N of these predictions as recommendations. As reviewed in Chapter 2, predictions are often computed as a weighted average of deviations from neighbour means [HKBR99]:

$$p_{a,i} = \bar{r}_a + \frac{\Sigma(r_{b,i} - \bar{r}_b) \times w_{a,b}}{\Sigma w_{a,b}} \tag{3.5}$$

In other words, a prediction $p_{a,i}$ of item i for user a is an average of the set of deviations $(r_{b,i} - \bar{r}_b)$ from each neighbour's mean rating $\bar{r}_b$, weighted according to the similarity $w_{a,b}$ between the user a, and

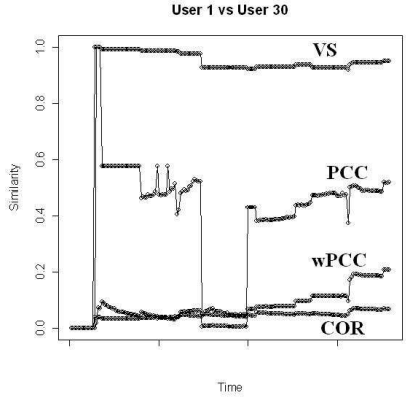


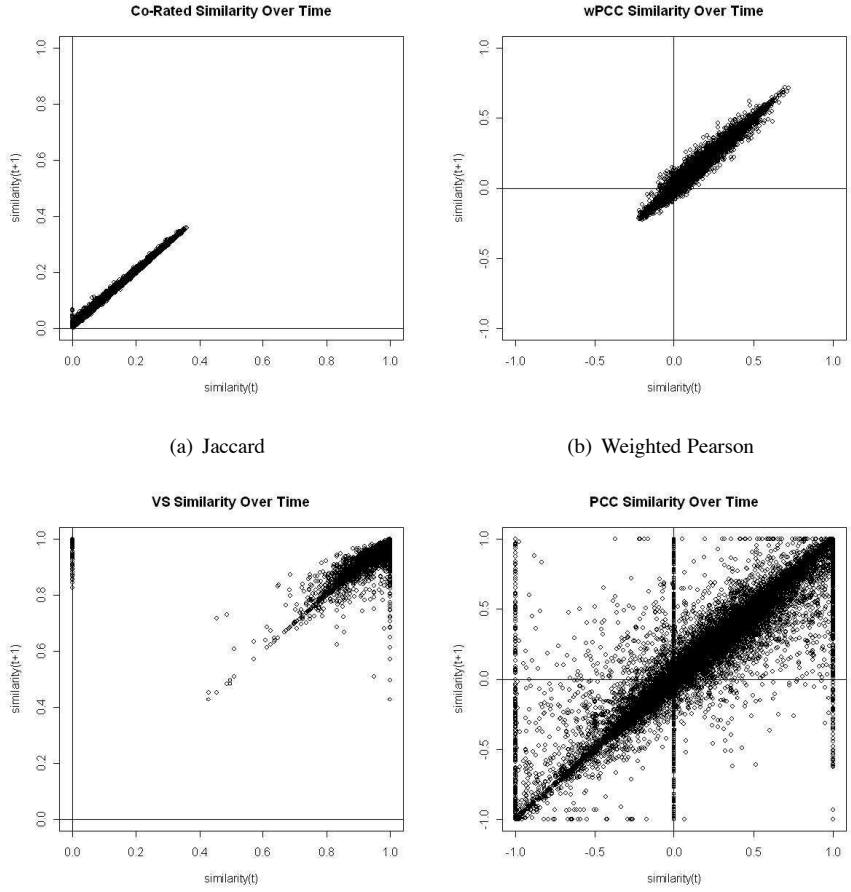
Figure 3.18: Similarity Between User 1 and 30: Similarity depends on how you measure it

neighbour b. All methods share the fact that they weight the contribution of each neighbour according to the degree of similarity shared with the current user: similarity is central to this process.

As we saw in Section 3.3.2, various similarity metrics offer different ways of computing similarity and will equally produce differing values. Despite this disagreement between similarity measures, one would expect the similarity between pairs of users to *converge*. As ratings are added to one of the two profiles, the similarity measure is computed on more information and should become more refined. However, some similarity measures do not display this behaviour.

We can consider a small example: users 1 and 30 from the ML-1 dataset. We chose this pair of users since their profiles have a large overlap over time (126 days), allowing for an extended view of their similarity's progression. If we order their profiles temporally, and then iteratively re-compute the similarity between the two as each user inputs a rating, we can observe how similarity evolves over time. Figure 3.18 shows the results of this experiment; in this case all measures return positive similarity between the users. The similarity for all measures begins at zero, when there is no overlap between the two users' profiles. Once they begin to co-rate items, the Cosine measure skyrockets to near 1.0, or perfect similarity. Over time, it very gradually degrades. The PCC measure also displays a large shift away from zero when the profile overlap begins and then both returns toward zero and jumps back up as the overlap increases. Only the wPCC and Jaccard measures grow slowly, without large shifts in similarity from one measurement to the next.

This example displays how the similarity between this particular pair of users progresses. In order to be able to generalise these results, we next aimed to analyse how the similarity of user 1's profile evolves relative to any other user in the system. There are a number of ways this evolution can be visualised; in this work we plot the similarity at time t, $sim(t)$ against the similarity at the time of the next update, $sim(t+1)$. This way we disregard the *actual* time between one update and the next, and favour focusing on how the similarity itself between a pair of users evolves. This method also allowed us to plot the similarity of one user compared to all others in the dataset, as we have done in Figure 3.19. These four images show the similarity of user 1 in the ML-1 dataset compared to the rest of the community, using different similarity measures. These results are similar to those we observed between the pair of users



(a) Jaccard

(b) Weighted Pearson

(c) Cosine

(d) Pearson

Figure 3.19: Evolution of Similarity for the Jaccard, wPCC, Cosine and PCC Similarity Masures, Comparing User 1 to All Other Users in the System

we examined before.

The first point to notice is that the range of values returned by the different similarity measures is not the same; some measures return values between 0.0 and 1.0, while others report values between -1.0 and 1.0. However, the more important aspect of these plots is the variance the points have from the diagonal, or the line $y = x$. If a point is on the diagonal it means that the similarity between the pair of users at time $(t + 1)$ is the same as it was at time t; nothing has changed. Similarly, if the point is below the diagonal then the pair is less similar that it was before, and a point above the diagonal implies that the measured similarity has grown. Therefore, the distance that these points have from the diagonal represents the extent to which similarity between the pair changed from one update to the next. As is visible in the plots of Figure 3.19, the greatest distance from the diagonal is reported in both the Cosine and PCC measures. These reflect the observations that were made when we compared user 1 and 30. Furthermore, they are representative of plots we created for other members of the community; these are not included here due to lack of space. The way that these methods evolve similarity between a pair of users follows one of three patterns. This allows for similarity measures to be classified according to their temporal behaviour:

k	COR	wPCC	PCC	Cosine
ML-1 Dataset: 943 Users				
1	2.48±2.3	2.54±2.3	4.94±6.5	10.59±16.8
10	22.22±16.3	22.18±16.2	25.15±19.9	35.80±41.4
20	42.06±28.6	42.12±27.9	41.73±26.4	49.88±45.3
100	171.80±87.9	168.99±83.9	156.27±67.4	159.03±69.9
150	237.86±109.4	230.23±104.9	216.94±88.6	221.16±87.1
ML-2 Dataset: 6040 Users				
1	1.69±1.1	1.74±1.2	3.51±3.8	3.16±5.1
10	16.75±9.3	16.85±9.4	22.75±19.0	34.68±36.2
20	33.22±17.7	33.33±18.1	40.08±28.2	60.02±54.6
100	160.26±79.1	160.93±79.9	161.68±77.4	187.02±118.3
150	236.97±113.6	237.99±114.5	231.35±102.6	255.23±142.9

Table 3.4: Average Unique Recommenders in Users' Neighbourhoods

- **Incremental**: In this case, as we observed with the Jaccard and wPCC methods, similarity begins at zero and slowly converges towards the final value. The difference between one step and the next is minimal, and therefore the relationship between a pair of nodes can be described as growing.
- **Corrective**: The Cosine method is noteworthy because similarity "jumps" from zero to near-perfect. However, once it has made this jump, the similarity between the pair tends to degrade, as can be observed by number of datapoints that fall below the diagonal on the graph. Therefore, this measure corrects its result after the initial jump.
- **Near-random**: The last class of similarity measures includes the PCC, and displays an exceeding amount of near-random behaviour. In other words, if similarity at time t is 0.0, or incomparable, and at time $(t+1)$ there is measurable similarity, the PCC returns values over the entire range of similarity. Once it has made this jump from zero in either direction, it is not guaranteed to be well-behaved; as the plot shows, it may very well make a large jump again.

Dynamic Neighbourhoods

Now that we have observed how similarity evolves between a pair of nodes, we can widen the scope of our analysis and consider *user neighbourhoods*. The importance of measuring similarity of all user pairs is to be able to create a subjective ranking for each user of everyone else, and then to pick the top-k to form the user neighbourhood. The often-cited assumption of collaborative filtering is that users who have been like-minded in the past will continue sharing opinions in the future; this assumption has thus paved the way for learning algorithms to be applied to the problem of predicting ratings. If we project this assumption onto a longer time period, we would expect groups of users to naturally emerge from the data. In particular, when applying user-user kNN CF, as we do in this work, we would expect each user's neighbourhood to converge on a fixed set of neighbours over time.

To measure this property, we ran a modified CF experiment that includes the idea of system updates. The system begins at the time of the first rating in the dataset and is updated daily. While this value

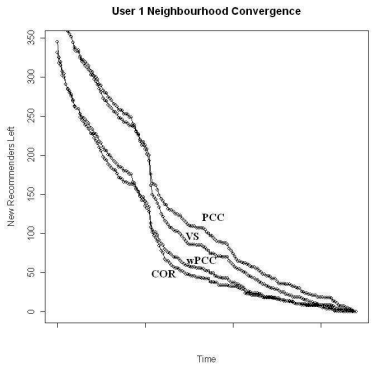


Figure 3.20: ML-1 User 1: New Relationships Left Over Time

perhaps corresponds to more frequent updates than most recommender systems can allow themselves to perform, it gives a finer-grained insight into the behaviour of the CF algorithm. At each update time, all user neighbourhoods are re-computed. In this chapter, we do not consider temporal accuracy, as we are focusing on the dynamic graph properties imposed by the algorithm.

As the users' profiles grow and neighbourhoods are recomputed, the users will be connected to a varying number of other users. The actual number of neighbours that a user will be connected to depends on both the similarity measure and neighbourhood size that is used. If, for example, $k = 1$, the user's profile is updated 10 times, and at each time step a different neighbour becomes the user's top recommender, then the user will meet 10 unique neighbours: the higher the number of unique recommenders, the higher the volatility of the user's neighbourhood. Table 3.4 displays the average unique neighbours for all users in the datasets.

The first point to note is that the number of unique recommenders is not close to k; in most cases it is nearly double the size of the allowed neighbourhood. In other words, even though a particular value of k represents the number of neighbours to use when making a prediction, the fluctuation of neighbours over time will be such that about double this value will be interacted with. For most values of k, the COR and wPCC similarity measures assign fewer unique recommenders to each user, a result that is not immediately visible when using the average number of neighbours across all users that Table 3.4 does.

Figure 3.20 shows the number of unique neighbours that user 1 has yet to meet over time when $k = 150$; it thus visualises how quickly the change within the user's neighbourhood will play out. As with the similarity plots, it is the shape of the plotted lines that gives insight into how neighbourhoods are changing over time: the steeper they are, the faster the user is meeting other recommenders. If the lines were step-shaped, the user would be meeting recommenders and staying connected to them for some time. Steeper lines, however, mean that the user's neighbourhood is converging faster, since the number of unique neighbours that have yet to be seen is decreasing. In fact, the Jaccard and wPCC similarity measures also converge to a fixed set of known recommenders faster.

Nearest-Neighbour Graphs

The last perspective we consider is the broadest view possible: the entire graph of user profiles. We have already seen that the volatility of each user's neighbourhood is quite large: this implies that the entire

k	Edges	Connected?	Max Path	Avg Path	Reciprocity
ML-1 Dataset: 943 Users					
1	1750	No	3	1.78	0.08
10	16654	Yes	4	2.63	0.13
100	148608	Yes	3	1.83	0.27
150	213260	Yes	2	1.76	0.33
200	272970	Yes	2	1.69	0.38
ML-2 Dataset: 6040 Users					
1	11406	No	5	2.58	0.06
10	109438	Yes	5	3.29	0.10
100	1055188	Yes	3	2.01	0.14
150	1568576	Yes	3	1.96	0.16
200	2076112	Yes	3	1.94	0.16

Table 3.5: wPCC-kNN Graph Properties

graph is being "re-wired" each time an update is performed. Therefore, in this section, we mainly focus on non-temporal characteristics of the dataset represented as a graph instead. Since the kNN algorithm determines where the links between users in the graph will be, the link positioning gives us the clearest insight into how the algorithm is manipulating the user-rating dataset. Table 3.5 shows a number of properties of the wPCC-kNN graph, for various values of k; we do not include results for the other similarity measures since they are very similar.

Path Length. Table 3.5 reports the maximum and average path length between any two nodes. These values were computed using Floyd's algorithm, based on an undirected representation of the kNN graph. In other words, we assume that if a link between the pair exists (regardless of its direction), then so does some measurable quantity of similarity. Another curious characteristic of the values reported in Table 3.5 is that while k increases, the maximum and average path between any pair of nodes remains small, ranging from 1.4 to 2.9 hops; in fact, the graph demonstrates small-world properties that are very similar to those measured in explicit social networks.

Connectedness. An analysis of the entire graph, generated using only positive similarity links, shows that the clusters of users appear depending on the neighbourhood size parameter k that is used. When $k = 1$, a number of small clusters of users emerge, regardless of what similarity measure is used. The different methods only vary on average intra-cluster path length (as explored above); this reflects the way that these small clusters are shaped. In some cases, such as the wPCC graph, the clusters are formed of a group of nodes that all point to the same top-neighbour. In other cases, such as in the COR graph, the clusters form small chains of nodes, which accounts for the longer intra-cluster path length between users. The majority of these characteristics disappear as soon as k is incremented above one. As soon as users are allowed to point to more than their single most similar neighbour, the graph collapses in on itself: clusters are lost and, in most cases, the graph becomes fully connected.

Reciprocity. We counted the number of edges as the number of links between nodes, whether they be directed or not. In fact, when $k = 1$, the number of edges is *less than* the $1\times$ the total number

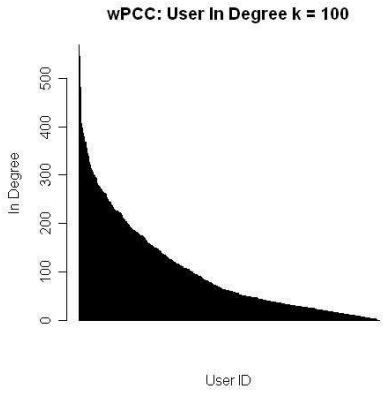


Figure 3.21: In-degree long tail of wPCC-kNN $k = 100$ ML-1 Graph

of nodes. This is due to the fact that in some cases, a pair of nodes point to each other; two directed links turn into a single undirected link, and the pair have a reciprocal relationship. Reciprocity is a characteristic of graphs explored in social network analysis [KNT06]; in our context it translates to the proportion of users who are in each other's top-k. On the one hand, reciprocity may be regarded as a desirable characteristic, since it implies that the generated graph really does pair very similar users together. On the other hand, high reciprocity can have dire consequences, as it will prevent information from being propagated over the similarity graph. The index of reciprocity that we use in Table 3.5 is the number of bi-directional links between nodes over the total number of links. The value ranges from 0, or no reciprocity, to 1, where all nodes pairs have reciprocal relationships. As the table shows, reciprocity grows as the allowed number of neighbours increases, and remains minimal when $k = 1$. However, it does not grow very much: adding a large number of links when k is incremented from 10 to 100 does very little to increase the measured reciprocity between users. This reflects the fact that although measured similarity is symmetric, this does not imply that each user will also have the same rank in the other's top-k; and this will matter when computing recommendations.

In Degree Distribution. We can further observe this phenomenon by considering the in-degree distribution of the nodes in the graph. The in-degree of a particular node n is the number of directed links that end on this node; in the context of collaborative filtering this equates to the number of users who place user n in their top-k. Figure 3.21 shows the in-degree of each user in the wPCC kNN graph, when $k = 100$. The distribution follows a power-law, much like the distribution that compares the number of ratings between different movies [LHC08b].

The in-degree distribution amongst users brings to light a new characteristic of kNN algorithms. Given a CF dataset and a nearest neighbour parameter k, there may be some users who are *not* in any other's top-k. Their ratings are therefore inaccessible and, although they will be considered when estimating the similarity between a pair of users, they will not be used in any prediction. To highlight this factor, we ran kNN prediction algorithms using the four similarity measures we are focusing on in this work on the ML-1 MovieLens subsets. Each rating in the training sets was coupled with a boolean flag, which would be set to true if the rating was used in making any prediction. We were thus able to

ML-1 Dataset				
k	COR	wPCC	PCC	VS
1	0.92	0.91	0.99	0.99
10	0.59	0.59	0.95	0.95
100	0.23	0.25	0.81	0.85
150	0.12	0.16	0.59	0.71
200	0.05	0.05	0.18	0.42

Table 3.6: Unused Proportions of the Dataset

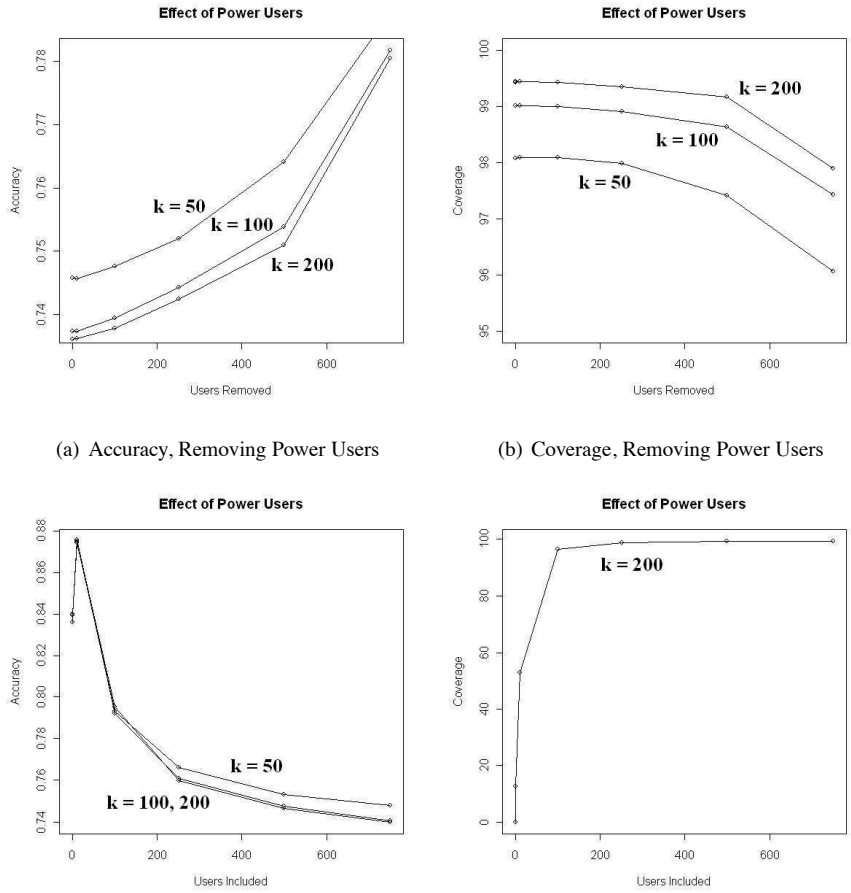
count how much of the training set remained unused after all the predictions had been completed.

Table 3.6 reports the proportions of the ML-1 dataset that are not used for varying values of k. The table does not reflect how many times individual ratings may have been used; it only counts whether the rating has ever been used or not. As the table shows, when k is very low, over 90% of the ratings are not used. In fact, these values of k generate predictions based on a very small subset of the training data, which may thus account for why they suffer from lower accuracy and impoverished coverage. As k increases, so does the use of the training data; if k were set to the total number of users in the system then the only ratings that would not be used would be those of a user who has no measurable similarity to any other in the system. However, a difference between the better-performing COR/wPCC and lower-accuracy PCC/VS similarity measures emerges once again: as k increases the former quickly use more of the dataset in predictions. When $k = 200$, only 5% of the training ratings are not used in predictions, while the VS similarity measure has barely made use of more than half of the ratings. The intuitive benefit of the COR/wPCC similarity measures may very well emerge here: they offer broader access to the ratings in the training set.

The Influence of Power Users

Another observation from Figure 3.21 is that some users will have an exceptionally high in-degree. We call this group *power* users; by being a frequently selected neighbour, they will have a stronger influence on the predictions that are made for others. These users emerge from the use of all the above similarity measures in kNN graphs. This is a characteristic that appears in other networks like the World Wide Web, movie actor collaboration graphs, and cellular networks, and is explained in terms of *preferential attachment* [BA02]. In other words, when a new node connects to the graph, the probability that it connects to another node is proportional to the in-degree of that node. In the context of collaborative filtering, therefore, it it important to understand the effect that generating a nearest-neighbour graph with power users has on the performance of the algorithm. We therefore ran two separate experiments. In the first, we forced all users' similarity with the top-P power users to be 0: in effect, removing their ability to contribute to predictions.

Figures 3.22(a) and 3.22(b) are the 5-fold cross validation mean absolute error and coverage results when removing a varying number of power users, for different values of k. As power users are removed, both accuracy and coverage worsen, although even when 750 (out of 943) profiles are made inaccessible



(a) Accuracy, Removing Power Users

(b) Coverage, Removing Power Users

(c) Accuracy, Only Power Users

(d) Coverage, Only Power Users

Figure 3.22: Results When Excluding or Exclusively Using Power Users

accuracy is still within 0.78. It seems, therefore, that the remaining non-power users can still make significant contributions to each user's predictions. These results reflect the dependency that accuracy has on the number of users in a system, another relationship that remains unexplored.

We followed this experiment by performing the inverse. Figures 3.22(c) and 3.22(d) show the 5-fold cross validation accuracy and coverage results when *only* the top-P power users are allowed to contribute to predicted ratings; if a neighbour is not a power user, a zero similarity value is set between the pair. The early spike in the plot is explained as follows: making predictions by simply returning each users' mean rating outperforms using only the topmost power user alone, but accuracy quickly returns to the same as when no users have been removed from the dataset when P increases; in other words, there are some user profiles in the dataset that do not contribute at all to the overall performance. The coverage plot shows a potential reason why these users are power users: the 10 topmost power users hold access to over 50% of the dataset.

3.4 Summary

In this chapter, we have examined the temporal characteristics of recommender system data, from the perspective of the ratings, users, and items. We have observed how the way people use recommender

systems changes over time: new users and items are added, the rating distribution and both global and per-user/item summary statistics change. In other words, *all* features of the data that are used to make predictions in state of the art algorithms will vary with time.

We also performed a graph analysis of inter-user similarity, including the changes that appear throughout these graphs as time passes. The evolution of similarity between any pair of users is dominated by the method that is used to measure similarity, and the four measures we explored can be classified into three categories (*incremental, corrective, near-random*) based on the temporal properties they show. The number of unique neighbours that a particular user will be given over time also depends on both the similarity measured and parameter k used; furthermore, the rate at which they meet these new neighbours will vary for different similarity measures. Measures that are known to perform better display the same behaviour: they are *incremental*, connect each user quicker and to fewer unique neighbours, and offer broader access to the ratings in the training set. The focus here, therefore, is on the emergent *structure* of the graph using the MovieLens dataset.

In the following chapters, we shift our focus toward the temporal *performance* of CF algorithms. Collaborative filtering algorithms have traditionally been evaluated by: (1) splitting a dataset of user ratings into training and test sets, (2) feeding the training set into the learning algorithm, and (3) querying the algorithm for predictions of items in the test set. Evaluations are then conducted by comparing the predictions to the actual ratings that were withheld in the test set. There are two problems with this setup: both the *metrics* and *methodology*, in their current form, are not suited to a context in which a sequence of updates is required. We therefore first define a methodology for performing temporal experiments and examine how the changes to the data observed here affect the accuracy of rating predictions (Chapter 4). We then evaluate the temporal diversity in recommendations produced by changing data using novel metrics (Chapter 5). Lastly, we use the regularity in users' behaviour to construct systems that are robust to attack (Chapter 6).