kNN CF: A Temporal Social Network

Neal Lathia
Dept. of Computer Science
University College London
London, WC1E 6BT, UK
n.lathia@cs.ucl.ac.uk

Stephen Hailes
Dept. of Computer Science
University College London
London, WC1E 6BT, UK
s.hailes@cs.ucl.ac.uk

Licia Capra
Dept. of Computer Science
University College London
London, WC1E 6BT, UK
l.capra@cs.ucl.ac.uk

ABSTRACT

Recommender systems, based on collaborative filtering, draw their strength from techniques that manipulate a set of user-rating profiles in order to compute predicted ratings of unrated items. There are a wide range of techniques that can be applied to this problem; however, the k-nearest neighbour (kNN) algorithm has become the dominant method used in this context. Much research to date has focused on improving the performance of this algorithm, without considering the properties that emerge from manipulating the user data in this way. In order to understand the effect of kNN on a user-rating dataset, the algorithm can be viewed as a process that generates a graph, where nodes are users and edges connect similar users: the algorithm generates an *implicit social network* amongst the system subscribers. Temporal updates of the recommender system will impose changes on the graph. In this work we analyse user-user kNN graphs from a temporal perspective, retrieving characteristics such as dataset growth, the evolution of similarity between pairs of users, the volatility of user neighbourhoods over time, and emergent properties of the entire graph as the algorithm parameters change. These insights explain why certain kNN parameters and similarity measures outperform others, and show that there is a surprising degree of structural similarity between these graphs and explicit user social networks.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering

General Terms

Algorithms, Performance, Theory

Keywords

Similarity, Recommender Systems, Power Users, Temporal Graph Analysis

RecSys'08, October 23–25, 2008, Lausanne, Switzerland.

1. INTRODUCTION

Recommender systems are blossoming; they are no longer simply addressing information overload [1], but are now an important component, or even core technology, of online business. k-Nearest Neighbour Collaborative Filtering (kNN CF; from here on referred to as kNN) has surfaced amongst the most popular underlying algorithms of recommender systems. The wide success that it has experienced is due to the fact that it automates the process of collecting and combining human judgements of content. In doing so, it can compute recommendations that inherently include a sense of quality, and not merely the closeness of the descriptions between two items, as content-based approaches attempted [2]. The process of computing recommendations is therefore reduced to a problem of *predicting* the correct rating that users would apply to unrated items; this assumption is reflected in the ongoing Netflix prize competition[1].

The race for evermore accurate predictors continues, and the leaders are found using mean error measures. However, mean error values do not highlight the underlying behaviour that the algorithm applies onto the community of user profiles; they mask the process and focus only on the results. Little has been explored of the emergent properties of the algorithms applied to this problem, and, furthermore, current evaluation metrics do not reflect how the system will change over time, even though all deployed recommender systems will have a dynamic user base and growing user-rating dataset.

An alternative means of analysing a kNN algorithm is to consider it as a process that generates a graph [3]. For example, the algorithm could be an implementation of user-user CF, based on the Pearson similarity [4], where $k = 10$. In this case, each user is a node in the graph. Links to other nodes are weighted according to how similar the user-pair is, and the algorithm imposes the restriction that each node can only link itself to the 10 most similar neighbours; the out-degree of each node is limited. From this perspective, kNN algorithms are constructing a *constrained implicit social network* between the users in the system, by assigning relationships according to the methods and parameters that it operates upon. 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 a kNN filtering algorithm as a graph-generating process paves the way for a wide range of analysis that can be performed on these systems, drawing from methods de-

[1] http://www.netflixprize.com

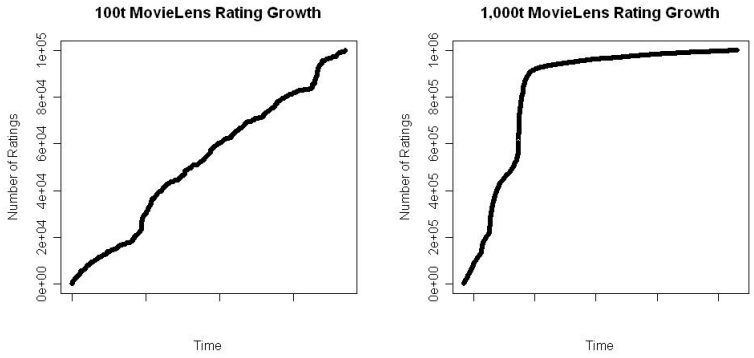

Figure 1: Rating Set Growth

scribed in graph theory and previous work on (explicit) social network analysis [5, 6]. 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.

We therefore propose to explore the emergent properties of dynamic, temporal user-user kNN graphs, by decomposing the analysis into four separate stages:

- **Individual Nodes**: Section 2 describes how user-rating datasets are temporal in their nature; as time passes, new nodes will be added to the graph, and the amount of information (or ratings) that individual nodes hold will also change.

- **Node Pairs**: Drawing from the growth of both nodes and rating information, Section 3 explores 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. Section 4 projects this restriction onto the temporal scale, and allows us to 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 kNN graphs display the small-world, scale-free characteristic that is common to social networks. In other words, kNN 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 algorithm.

2. USER PROFILES OVER TIME

As described above, the graph generated by a kNN algorithm represents the relationships between a set of user profiles. The number of profiles is variable over time, as the user base of the system changes. These profiles are subject to change; as users enter more ratings, they broaden the breadth of the information the system holds on their likes and dislikes, and change the amount of information available that can be used to compute recommendations.

In this work we focus on the two MovieLens datasets[2]. The first, which we refer to as $100t$ MovieLens, contains $100,000$ ratings of 1682 movies by 943 users. The earliest rating of the dataset has the Unix time stamp 874724710, the morning of the 20^{th} of September 1997 and the last rating is marked 893286638, the evening of the 22^{nd} of April 1998; this dataset is a sub-sample of the MovieLens recommender system spanning approximately 7 months. The second dataset, refered to with the pseudonym $1,000t$ MovieLens, consists of approximately 1 million ratings of 3900 movies by 6040 users. It is both larger and spans a longer time period, ranging from the evening of the 25^{th} of April 2000 until the 28^{th} of February 2003 (approximately 2 years and 10 months).

The user profiles in each dataset differ in size, rate of growth, and time span. The first characteristic that we extracted, therefore, is how the dataset changes over time; Figure 1 shows the growth of the number of ratings. The $100t$ MovieLens dataset has a near-linear growth over time, while the larger $1,000t$ MovieLens dataset has explosive initial growth, but then flattens out over the rest of the time period.

In order to understand why the datasets display different growth patterns, we consider the number of *active* users in the system over time. A user is active if they have already contributed ratings to the dataset and will also contribute in the future. Figure 2 shows the fluctuation of active users over time for each dataset (computed on daily iterations). The $100t$ dataset never has more than about 140 active users, a value that remains relatively constant over time, and accounts for the linear growth observed in the dataset. The larger dataset has an early peak of about $1,200$ users and then decays back to zero. The flattening of the rating set growth can hence be clarified: after the initial peak, the number of active users shrinks, therefore hindering further

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

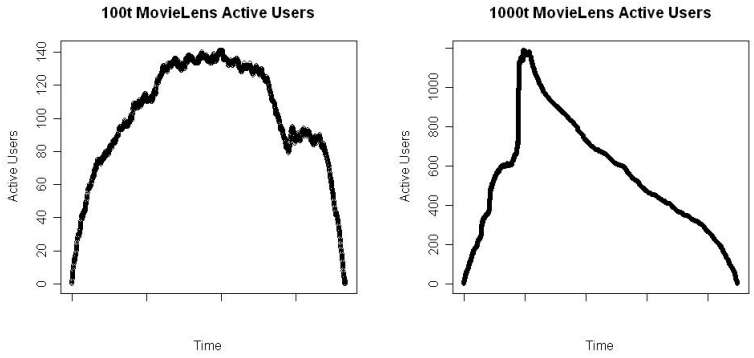

Figure 2: Active Users in Time

growth of the dataset. Only a subset of the nodes, or user profiles, in the kNN graph will be changing at a particular instance. As we will see in Section 5, the effect of a change of a subset of nodes resounds over the entire community.

3. USER PAIRS OVER TIME

Based on the way the datasets change over time, we 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. Predictions are often computed as a weighted average of deviations from neighbour means [7]:

$$p_{a,i} = \bar{r}_a + \frac{\Sigma(r_{b,i} - \bar{r}_b) \times w_{a,b}}{\Sigma w_{a,b}} \quad (1)$$

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 neighbour b. Other methods only consider the ratings themselves, rather than the deviations [8]; however, 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.

In this work, we focus on four highly cited methods [3, 7]:

- **COR**, the proportion of co-rated items:

$$w_{a,b} = \frac{|R_{a,i} \cap_i R_{b,i}|}{|R_{a,i} \cup_i R_{b,i}|} \quad (2)$$

- **VS**, the vector similarity:

$$w_{a,b} = \frac{a * b}{|a| * |b|} = \frac{\Sigma_{i=1}^{N} a_i b_i}{\sqrt{\Sigma_{i=1}^{N} a_i^2}\sqrt{\Sigma_{i=1}^{N} b_i^2}} \quad (3)$$

- **PCC**, the Pearson correlation coefficient:

$$w_{a,b} = \frac{\Sigma_{i=1}^{N}(r_{a,i} - \bar{r}_a)(r_{b,i} - \bar{r}_b)}{\sqrt{\Sigma_{i=1}^{N}(r_{a,i} - \bar{r}_a)^2 \Sigma_{i=1}^{N}(r_{b,i} - \bar{r}_b)^2}} \quad (4)$$

- w**PCC**, the weighted Pearson correlation coefficient, where the PCC is scaled by the number of co-rated items $n/50$ if n is less than 50:

$$w_{a,b} = \frac{n}{50} \times \frac{\Sigma_{i=1}^{N}(r_{a,i} - \bar{r}_a)(r_{b,i} - \bar{r}_b)}{\sqrt{\Sigma_{i=1}^{N}(r_{a,i} - \bar{r}_a)^2 \Sigma_{i=1}^{N}(r_{b,i} - \bar{r}_b)^2}} \quad (5)$$

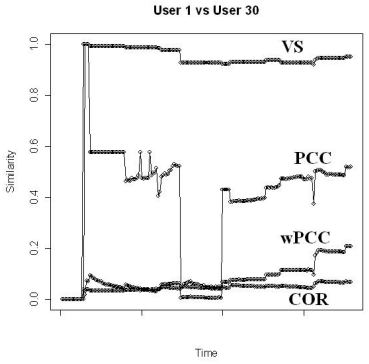

Figure 3: Similarity Between User 1 and 30

Each method offers a different way of computing similarity, and will result in different values. In other words, when the VS claims that two users are highly similar, the PCC may result in dissimilarity, and the wPCC may return a near-zero value: the distribution of similarity will vary across the entire graph [3]. Despite this odd behaviour, and based on the assumption that the better measures of accuracy become, the better they will be at finding the "right" neighbours for each user, 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 becomes more refined. However, some similarity measures do not display this behaviour.

We can consider a small example: users 1 and 30 from the 100t MovieLens dataset. We chose this pair of users since their profiles have a large overlap over time, 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 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

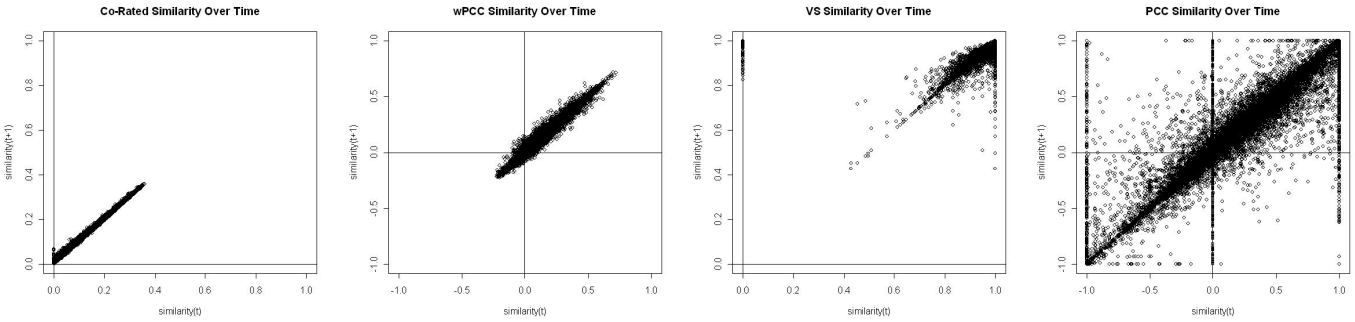

Figure 4: Evolution of Similarity for the COR, wPCC, VS and PCC Similarity measures

between the two users' profiles. Once they begin to co-rate items, the VS 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 COR measures slowly grow, 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 at analysing 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 4. These four images show the similarity of user 1 in the 100t MovieLens dataset compared to the rest of the community, using different similarity measures.

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 4, the greatest distance from the diagonal is reported in both the VS and PCC measures. These reflect the observations that were made when we compared a 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:

- **Incremental**: In this case, as we observed with the COR 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 VS 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.

4. 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 the underlying assumption of like-mindedness, used to justify how collaborative filtering algorithms are designed, onto a longer time period, then 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 to 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 (given in Section 2), and will be updated daily. While this value

k	COR	wPCC	PCC	VS
100t MovieLens: 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
1,000t MovieLens: 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 1: Average Unique Recommenders in Users' Neighbourhoods

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

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 1 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 less unique recommenders to each user, a result that is not immediately visible when using the average number of neighbours across all users that Table 1 does.

Figure 5 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 COR and wPCC similarity measures, which outperform the PCC and VS methods (as discussed in [3]), also converge to a fixed set of known recommenders faster.

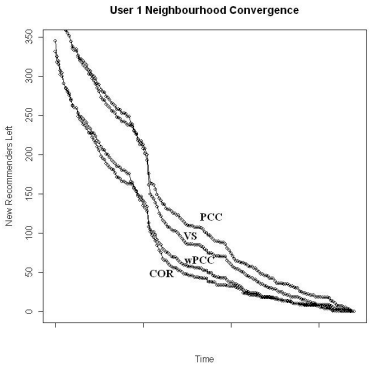

Figure 5: New Relationships Left Over Time

5. 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 users' neighbourhood is quite large: this implies that the entire 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 2 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 due to lack of space.

Path Length. Table 2 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 2 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 the clusters

k	Edges	Connected?	Max Path	Avg Path	Reciprocity
100t MovieLens: 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
1,000t MovieLens: 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 2: wPCC-kNN Graph Properties

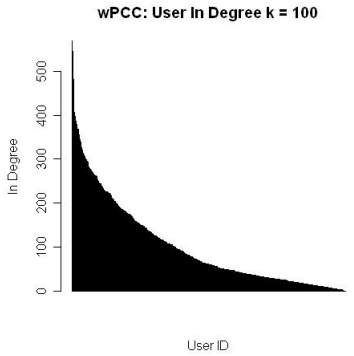

Figure 6: In-degree long tail of wPCC-kNN $k = 100$ 100t MovieLens Graph

of users do 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. 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 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 [9]; 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 a desired characteristic, since that 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 able to be propagated over the similarity graph. The index of reciprocity that we use in Table 2 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 an explosive 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, it does not imply that each user will also have the same rank in the other's top-k.

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 6 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 number of ratings between different movies [3].

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 inaccesible 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 100t 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 count how much of the training set was not used once all the predictions had been completed.

Table 3 reports the proportions of the 100t MovieLens 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

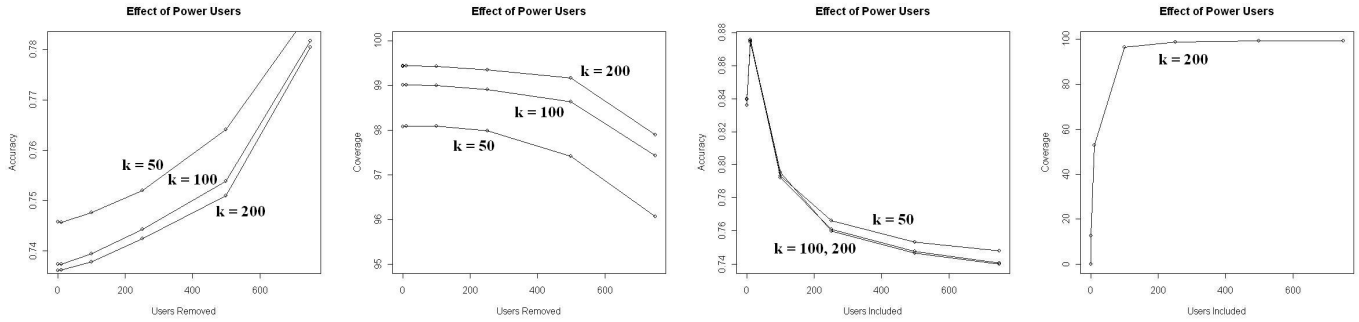

Figure 7: Effect of power users on the 100t MovieLens dataset

100t MovieLens				
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: Unused Proportions of the Dataset

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 ratings. The benefit of these similarity measures may very well emerge here: they offer broader access to the ratings in the training set.

Another observation from Figure 6 is that some users will have an incredibly 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* [5]. 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.

The left plots of Figure 7 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. When no power users are removed, the accuracy and coverage results are as expected. However, as power users are removed both accuracy and coverage worsen, although even when 750 (out of 943) profiles are made inaccessible accuracy is still within 0.78. It seems, therefore, that the remaining non-power users can still make significant contributions to each users' predictions.

These results reflect the dependence that accuracy has on the number of users in a system, another relationship that remains unexplored.

We followed this experiment by perfoming the inverse. The right plots of Figure 7 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. Making predictions by simply return 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.

6. DISCUSSION AND RELATED WORK

In this work we have performed a graph analysis of the emergent properties of user-user kNN collaborative filtering algorithms, including the changes that appear throughout these graphs as time passes. The datasets grow over time, but only a subset of the community are *active* users at each moment. 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 less unique neighbours, and offer broader access to the ratings in the training set. The focus here, therefore, is on the emergent *structure* of the kNN graph using the MoviLens dataset; our future work aims at com-

paring these characteristics on different datasets. We also plan on repeating the above analysis for item-based kNN graphs; examining the similarities between the two graphs may provide insight into how to achieve better results when the two methods are fused together [10]. These kinds of insights offer the potential to improve kNN algorithms in a number of ways. In this section we discuss these applications and related work.

The Cold-Start Problem. We explored the effect that emergent *power users* have on predictive accuracy and coverage and validated that only using them still achieves comparable accuracy results. Cold-start users, who have no profile, may therefore be given a neighbourhood of power users until their profiles are sufficiently large to compute a separate neighbourhood. The term "power users" has been used before to describe users who have rated a lot of items [4] and those who exert a high degree of "influence" in predictions for others [11]. Our work veers away from the former, heuristic-based definition towards the latter, algorithmic-based identification of power users. It would be interesting to see if there is a correlation between these users (who, according to the user study in [4], do not rate to improve their recommendations) and the power users that emerge from the algorithm.

However, in this work we did not consider a number of characteristics of power users' profiles, or how they change over time. Being able to identify users who will be future power users may help finding the sources of *serendipitous* information; a method like this will find users with profiles that are growing towards the power-user status. On the other hand, information on new, unknown items may be found in the opposite users, who have little to no in-degree. We therefore plan on comparing the profiles of each user with the in-degree they are awarded. Richer datasets may also offer finer-grained insights. In particular, recent work on multidimensional recommender systems may show why power-users emerge, and how they can best be used [12].

Scalability. The mainstream approaches used to tackle the large number of users involves dimensionality reduction techniques [8]. Temporal patterns in neighbourhoods, however, can be taken advantage of to both reduce the complexity of recomputing the similarity between every user pair. Identifying the active users at a particular instance can potentially be used to reduce the time complexity of computing recommendations, and to determine which users are the sources of the most novel, recent (and potentially serendipitous) information. Furthermore, as discussed above, power users also offer good predictive power for the entire community. Their presence implies that often large proportions of the dataset are not used to generate predictions at all; and, given their performance, they may be used exclusively to make predictions. Only requiring a small set of power users to generate accurate predictions would vastly reduce the scalability issues that recommender systems face.

Parameter Selection In this work we defined kNN graphs as *constrained*: the out-degree of each node is set to k. Finding the optimal k is a problem relating to the broader field of parameter selection in machine learning techniques. However, from the perspective of a graph, we can see that k need not be the same for each user: in fact, the parameter may serve its fuction best (that is, providing the most accurate predictions) if it is *customised* for each user. kNN can thus be approached as a per-user, rather than a global, optimisation problem.

Online Learning. Further study of the temporal progression of the kNN algorithm is also required; understanding how historical interactions between users affect future ones can shed light on long-term emergent relationships between users in the community. This paves the way for online learning algorithms to be applied to the problem: extending the vast pool of available solutions that can be applied to collaborative filtering, and leading to systems that respond positively to user's needs as they interact with the recommender system.

7. REFERENCES

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

[2] M.J. Pazzani and D. Billsus. Content-based recommendation systems. *The Adaptive Web*, pages 325–341, 2007.

[3] N. Lathia, S. Hailes, and L. Capra. The effect of correlation coefficients on communities of recommenders. In *Proceedings of the* 23^{rd} *ACM SAC TRECK Track*, Fortaleza, Brazil, March 2008.

[4] 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.

[5] A. Barabasi and R. Albert. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, 2002.

[6] M. Maia, J. Almeida, and V. Almeida. Identifying user profiles in online social networks. In *Proceedings of the* 1^{st} *International Workshop on Social Network Systems (EuroSys 2008)*. ACM Press, 2008.

[7] 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.

[8] 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.

[9] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *International Conference on Knowledge Discovery and Data Mining*. ACM, 2006.

[10] J. Wang, A.P. de Vries, and M.J.T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th ACM SIGIR Conference*, pages 501–508. ACM Press, 2006.

[11] A. M. Rashid, G. Karypis, and J. Riedl. Influence in ratings-based recommender systems: An algorithm-independent approach. In *Proceedings of SIAM 2005 Data Mining Conference*, 2005.

[12] G. Adomavicius, R. Sankaranarayanan, and S. Sen. Incorporating contextual information in recommender systems using a multidimensional approach. In *ACM Transactions on Information Systems (TOIS)*, 2008.