

User-based Collaborative Filtering: Sparsity and Performance

Jennifer Redpath, David H. Glass, Sally McClean and Luke Chen¹

Abstract. It is generally assumed that all users in a dataset are sparse and hence addressing the problem of data sparsity should result in improved performance. However, although all users may be members of a sparse dataset, they do not all suffer from the data sparsity problem. This indicates that there is some ambiguity as to which users should be identified as sparse users and targeted with new recommendation improvement strategies. This paper defines sparsity in terms of number of item ratings and average similarity with nearest neighbours and then goes on to look at the impact of sparsity so defined on performance. Counter intuitively, it is found that in top- N recommendations sparse users actually perform better than some other categories of users when a standard approach is used. These results are explained, and empirically verified, in terms of a bias towards users with a low number of ratings. The link between sparsity and performance is also considered in the case of predictions rather than top- N recommendations. This work provides the motivation for targeting improvement approaches towards distinct groups of users as opposed to the entire dataset.

1 INTRODUCTION

Recommender systems [1] are typically employed to assist users with the discovery of items that may be of interest. A commonly-used approach for generating personalised recommendations is User-Based Collaborative Filtering (CF) [2], in which a database of user ratings profiles is searched for the closest matches to the active user, the user's Neighbourhood (NN). The process is modelled on how people innately interact with each other i.e. sharing opinions. These algorithms are evaluated in various ways, with a popular approach being the Leave- N -Out (also known as All-But- N) protocol [3]. Any items that are withheld from the recommender algorithm for the purpose of evaluation are referred to as the *hidden* set (H). The items remaining after H has been removed is referred to as the training set (T); this is the information that the system has available regarding the active user a . Either the Top- N items in the neighbourhood are selected to form the recommendations or predictions are generated for H .

The central components of a recommender system are the dataset and the recommender algorithm. The dataset can be thought of as a set of items that are available for recommendation to a set of users. Information stored on the users is usually in the form of ratings collected, either implicitly or explicitly, although there could also be other information, such as user demographics. If many of the hidden items occur in the recommendations or the algorithm can

accurately predict ratings for unknown items, the system is considered to be successful. Some researchers suggest that the only means for assessing a system is via on-line user studies [4]. These can record how the system recommendations are used by the individual, in other words, did the user continue to use the system? Unfortunately these studies are expensive in terms of time and finding a diverse test group. It must also be noted that a test group of users can be more sympathetic than real users and this must be taken into account.

This paper demonstrates that a problem in the evaluation of Top- N recommendations using traditional evaluation protocols means that the sparse users, as identified in the literature [5; 6], are not the worst performing group. Research focused on improving Top- N recommendations [7; 8] has not identified this problem, but has instead concentrated on addressing issues with the traditional evaluation metrics. Those users who perform worst when an algorithm has been evaluated are the users who require attention. The paper also shows that this problem does not affect evaluation of rating predictions for H .

The problem identified is a result of the protocols being biased towards users with a low number of item ratings. This problem is compounded by problems with common evaluation metrics; for instance, recall only measures how many previously viewed items are found in the hidden set, but not how likely a user is to rate these items highly [9]. In contrast, mean absolute error (MAE) provides the error rate for items that are rated but ignores recommended items that have no user rating; therefore they have no effect on the MAE result (this is only an issue when predicting ratings for items other than H).

The remainder of this paper is organised as follows; Section Two provides a definition for sparse users. This is followed by a detailed description of the methodology in Section Three for both Top- N recommendation and prediction generation; including the methods and metrics for evaluation. Section Four presents the results obtained from running the experiments, demonstrating well that the sparse users are not the worst performers for Top- N recommendation. A discussion, explanation and verification of the results are provided in Section Five, followed by a description of related work in Section Six. Finally, Section Seven completes the report with conclusions and suggestions for future work.

¹Computer Science Research Institute, University of Ulster,
Newtownabbey, Northern Ireland, email: redpath-j@email.ulster.ac.uk

2 SPARSITY IN USER-BASED CF

While there is generally agreement that sparsity is a problem in recommender systems, the term itself is often used in slightly different ways. An obvious way to characterise sparsity is in terms of number or ratings so that a user who has only a few ratings is considered to be a sparse user. The same approach can be applied not only to individual users, but to a dataset as a whole so that if a high proportion of user-item pairs have not been rated the dataset is sparse. The Sparsity Level measure [10], which will be considered later, is of this kind.

It also seems clear, however, that a user could have a low number of ratings and yet a high proportion of the items rated might also have been rated by other users. In this case the user might well have a high similarity with other users. By contrast, another user might have a high number of ratings yet only a small proportion of the items rated might have been rated by other users. In this case the user might have a low similarity with other users. In order to take this into account, the approach in this paper will be to consider both the number of ratings and the average similarity with nearest neighbours as contributing to sparsity. This will be discussed further in section 4.

In terms of addressing sparsity, a common approach is to consider the performance of an algorithm while the ratio of the number of items used in training to that used in testing is varied [10; 11]. The idea is that sparsity corresponds to the case where this ratio is low and so approaches for alleviating sparsity can be assessed in this case. This approach corresponds to the idea that sparsity is considered as a property of the dataset and so the focus is on the dataset as a whole rather than distinctly identified sparse users.

The approach adopted here is to identify sparse users in terms of their number of ratings and average similarity with nearest neighbours as noted above. The goal then is to investigate the performance of such users in standard recommendation tasks and to compare it with the performance of less sparse users.

3 METHODOLOGY

The core recommendation generation tasks are to suggest good items to users and/or make predictions that indicate the expected level of user satisfaction [12]. This paper concentrates on these two tasks, namely *Recommending Some Good Items* and *Predicting Ratings*. The well known MovieLens dataset [13] was used. There were 6,040 visitors with 1,000,199 ratings on 3,706 movies.

There is a wide variety of metrics used in the literature to assess the accuracy of a recommender algorithm. In fact, [14] reports that this abundance of metrics is what makes it so difficult for researchers to evaluate recommender systems. It is important to select the most appropriate metric for evaluating the corresponding recommendation task [15]. Recommender Algorithms can be evaluated in terms of predictive, classification or rank accuracy [12]. In this report, recall is the choice for evaluating the classification of *Recommending Some Good Items* and MAE was chosen for the *Predicting Ratings* task.

3.1. Recommending Some Good Items

It is assumed that there are many good items that could be of interest to the user. A user-centric approach to recommendation is to provide a small number of good suggestions, to prevent the user from having to search through a large set in order to find a handful of good ones. The recommended items at the top of the list should

be the items of most interest to the user, Top- N . In this report N is set to 20.

3.1.1. Collaborative Filtering Algorithm

The neighbourhoods throughout this paper are formulated using a standard user-based collaborative filtering approach with cosine as the similarity measure, which is defined as follows,

$$\text{sim}(u, v) = \frac{u \cdot v}{\|u\| \|v\|} \quad (1)$$

Cosine is the measure of the cosine angle between two user rating vectors, in this case u and v . When including non co-rated items [9] the denominator is able to account for the number of movies each user has rated individually. This ensures that users with a high number of ratings do not swamp the neighbourhood selections.

The approach adopted uses the Most-Frequently-Rated (*MFR*) [16] recommendation generation method. *MFR* is a simple incremental count of movie occurrences; movies are counted regardless of the rating attributed to them and all *NN* contribute equally, the Top- N *MFR* items that do not occur in the active user's test set are then recommended.

3.1.2. Evaluation of top- N

Recall is classified as a *Successful Decision Making Capacity* [17] metric, as its role is to measure the ability of a recommender algorithm to make successful decisions. It is the ratio of true positives to the number of hidden items H (Equation 2), i.e. this metric measures the proportion of successful recoveries from H . The number of true positives (TP) is a count of the number of times a recommendation has been found in the hidden items.

$$\text{recall} = \frac{TP}{H} \quad (2)$$

The interpretation of the evaluation method's result depends on whether the intention is to maximise the prediction of previously watched items, or to uncover the greatest number of movies the user has expressed a favoured opinion towards. In order to distinguish between these two different assessments, two evaluation methods are used: *Complete (COM)*, where any recommendation that corresponds to a hidden item is considered a *TP* and *Positive (POS)*, where only items found with a rating above a predefined threshold [8] contribute to the measurement of accuracy. In this paper the rating threshold is set to 3.

An additional measure used in the assessment of the recommender algorithm's decision-making success was User Coverage (UC), which is the proportion of users that receive one or more true positive recommendations.

3.1.3. Evaluation Protocols

The commonly used Leave- N -Out evaluation protocol can be applied to the dataset in two ways, (1) select a fixed number of items from every user to withhold, or (2) conceal a fixed percentage of each user profile. In this report these different dataset divides are referred to as *10Hidden* and *80/20*, respectively.

In *10Hidden*; the hidden items are selected by randomly withholding 10 items from each user's profile. The remainder of the data is used for training purposes. Each user receives a set number of recommendations. This approach is biased towards users

with small rating sets (explained further in section 5). When using the *10Hidden* evaluation protocol and *COM* evaluation technique there were 60,400 possible *TPs* to find, whereas with the *POS* approach only 37,575 movies were available. While using the *POS* approach, 27 users were completely omitted as none of these had any highly rated movies in their hidden set.

For *80/20*; 80 % of the data is retained for training and 20% of each user's profile is hidden from the algorithm. This approach deals with users with a high number of ratings more equally than *10Hidden*. As before, each user receives a set number of recommendations. The problem with this approach is apparent when the hidden set is greater than the recommendation set (explained further in section 5).

3.2. Predicting Ratings

For this particular task, a set of predicted ratings needs to be generated for items not rated by the active user. The use of predicted ratings is common when generating a set of recommendations. In this report, predictions are generated for all the hidden items.

Predictions are calculated as a weighted average deviation from the neighbour's mean:

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^n (r_{u,i} - \bar{r}_u) \times S_{a,u}}{\sum_{u=1}^n S_{a,u}} \quad (3)$$

where $p_{a,i}$ is the prediction for the active user a for item i ; $r_{u,i}$ is the rating for item i by neighbour u ; $\bar{r}_u$ and $\bar{r}_a$ are the mean rating of user u and a , respectively; $S_{a,u}$ is the similarity between users a and u ; and n is the number of users in the neighbourhood. For the experiments in this paper the top 30 neighbours who had rated the item i were considered. If less than 10 neighbours had rated the item then the average rating for that item in the dataset was used instead, known as the background rating.

3.2.1. Evaluating Predictions

Mean Absolute Error (*MAE*) is a prediction evaluation metric used for measuring the average absolute deviation from each predicted rating to each actual rating. It is formulated in Equation 4. *MAE* concentrates on the accuracy of a predicted rating as opposed to the accuracy of the recommendation [8]. Other metrics related to *MAE* are *Mean Squared Error*, *Root Mean Squared Error* and *Normalised Mean Absolute Error*.

$$\frac{\sum_{i=1}^N |f_i - y_i|}{N} \quad (4)$$

Where, f_i is the predicted or background rating, y_i is the actual rating the active user assigned to item i and N is the number of hidden items. The lower the *MAE* score obtained, the better the performance of the algorithm.

[7] argue that because *MAE* evaluates each predicted recommendation independently the results are biased towards algorithms that predict all items well, as opposed to predicting the top items well. The top items, usually 10 or 20, are the items of most interest to the user. This report suggests that if the algorithm can accurately predict some hidden items then this is an indication

of the quality of the recommendation set, even if they are not found in the hidden set.

4 RESULTS

A sparse user is defined as a user with few overlapping ratings and a small number of items rated. To identify these users the dataset was divided into quadrants, see figure 1, with the sparsest users in the low average nearest-neighbour similarity and low number of ratings quadrant, Quadrant 1, and the least sparse users in the high average nearest-neighbour similarity and high number of ratings quadrant, Quadrant 4. Quadrants 2 and 3 represent intermediate levels of sparsity.

Quadrant One: Low NN Similarity Low Number of Ratings	Quadrant Three: High NN Similarity Low Number of Ratings
Quadrant Two: Low NN Similarity High Number of Ratings	Quadrant Four: High NN Similarity High Number of Ratings

Figure 1. Quadrants for Categorising Users

The ranges for the quadrants were defined using the similarity results from *COSINE*, and the number of movies in the active user's training set. For example when using *10Hidden*, quadrant one is users with an average similarity value below 0.3475 and number of test ratings below 86.

Sarwar et al. define the Sparsity Level [10] as follows,

$$1 - \frac{\text{nonzero entries}}{\text{total entries}} \quad (5)$$

It measures the proportion of zero entries in the user-ratings matrix and so takes into account the number of ratings, but it does not take into account the similarity between users. The sparsity level of the entire MovieLens dataset is calculated as $1 - 1,000,199 / 6040 * 3706$, which is 0.96.

4.1. Recommendation Results

COM and *POS* recall and *UC* (as defined in section 3.1.2) is shown in table 1 and 2, respectively.

Table 1. *COM* and *POS* Recall using 10 Hidden, 20 Recommendations and *COSINE*

Quadrant 1 (2425 : 0.99)		Quadrant 3 (594 : 0.97)	
COM	POS	COM	POS
0.3159	0.2521	0.3980	0.3012
Quadrant 2 (595 : 0.96)		Quadrant 4 (2426 : 0.91)	
COM	POS	COM	POS
0.2311	0.1931	0.2596	0.2115

Table 2. *COM UC* and *POS UC* using 10 Hidden, 20 Recommendations and *COSINE*

Quadrant 1 (2425 : 0.99)		Quadrant 3 (594 : 0.97)	
COM UC	POS UC	COM UC	POS UC
96%	89%	99%	94%
Quadrant 2 (595 : 0.96)		Quadrant 4 (2426 : 0.91)	
COM UC	POS UC	COM UC	POS UC
86%	77%	91%	81%

The number of users followed by the sparsity level of each quadrant is shown in brackets after the quadrant number.

The results shown in tables 1 and 2 would indicate that the worst performing users are in Quadrants 2 and 4; users with a high number of test ratings. Surprisingly, it is Quadrant 3 that does best overall. Quadrant 1, assumed to be the sparse users, is the second best group of users. In fact the users that would be expected to perform best, Quadrant 4, are in 3rd place. It is worth noting that sparsity level scores correspond with the ordering of the quadrants as noted earlier, i.e. Quadrant 1 is the most sparse, Quadrant 4 least sparse and Quadrants 2 and 3 are intermediate. Table 2 supports the findings in table 1 demonstrating that more users in Quadrants 1 and 3 are receiving *TPs* than users in Quadrants 2 and 4.

The evaluation results when using *80/20* are displayed in tables 3 and 4. The quadrants for this data split are defined as $<0.3370 \geq$ for the average similarity, and $<77 \geq$ for the numbers of ratings.

Table 3. *COM* and *POS* Recall using *80/20*, 20 Recommendations and *COSINE*

Quadrant 1 (2235 : 0.99)		Quadrant 3 (784 : 0.98)	
COM	POS	COM	POS
0.2990	0.3453	0.3986	0.4574
Quadrant 2 (780 : 0.97)		Quadrant 4 (2241 : 0.91)	
COM	POS	COM	POS
0.1813	0.2237	0.1791	0.2322

Table 4. *COM UC* and *POS UC* using *80/20*, 20 Recommendations and *COSINE*

Quadrant 1 (2235 : 0.99)		Quadrant 3 (784 : 0.98)	
COM UC	POS UC	COM UC	POS UC
92%	84%	98%	94%
Quadrant 2 (780 : 0.97)		Quadrant 4 (2241 : 0.91)	
COM UC	POS UC	COM UC	POS UC
99%	97%	100%	99%

The results in table 3 are similar to table 1 in that the users with smaller rating vectors perform better with recall. The main differences are that when using *COM* Quadrant 4 becomes the worst group, but as shown by the sparsity level these are the users in the dense area of the dataset. Table 4 is different from table 2, as it is shown that the users in Quadrants 2 and 4 are covered better than users with smaller rating vectors (1 and 3). This would suggest that fixing the size of the hidden set has an effect on the number of users that will receive *TPs*.

4.2. Prediction Results

Table 5 shows the results when using *MAE* as the evaluation metric.

Table 5. *MAE* using *10Hidden*

Quadrant 1 (2425 : 0.99)		Quadrant 3 (594 : 0.97)	
0.7902		0.7402	
Quadrant 2 (595 : 0.96)		Quadrant 4 (2426 : 0.91)	
0.7649		0.7194	

When using predictions and evaluating the hidden items using *MAE* it is found that the sparse users (Quadrant 1) do perform worse. As expected the users in Quadrant 4, high similarity and high number of items rated, receive the best *MAE* results, followed

by the users with high similarity and low number of ratings (Quadrant 3).

5 DISCUSSION

As noted in section 4, Quadrant 1 represents the sparsest users within the dataset and Quadrant 4 the least sparse (or most dense) users. Correspondingly, we would expect performance to be worst for the former and best for the latter and this is indeed the case for prediction results evaluated using *MAE*. Quadrants 2 and 3 are intermediate cases with the former having a low average neighbourhood similarity and high number of ratings and the latter a high average neighbourhood similarity and low number of ratings. And as would be expected, the performance for the prediction experiments yields intermediate values for these quadrants. Interestingly, the results are slightly better for Quadrant 3 than Quadrant 2 even though the sparsity level as defined in equation (5) indicates that Quadrant 3 is sparser than Quadrant 2. This shows that a greater degree of sparsity does not necessarily correspond to poorer performance in the case of prediction if sparsity is quantified in terms of the sparsity level. However, the sparsity level only takes into account the proportion of non-zero entries in a dataset and does not incorporate the degree of similarity between users. The results in section 4 suggest that the latter has a greater impact on performance since the average similarity in Quadrant 3 is greater than that in Quadrant 2. Hence, if sparsity is to be correlated with performance, sparsity must be taken to include degree of similarity and not just number of ratings.

Much more surprising are the results for recommendation of the Top-*N* items. In particular, performance, as measured by recall, is best for Quadrant 3, even though it is sparser than Quadrant 4. Worse still, Quadrant 1 performs second best even though it is the sparsest quadrant by any reckoning. The expected correlation between degree of similarity and performance does hold since Quadrants 3 and 4 outperform Quadrants 1 and 2 respectively in the case of *10Hidden* presented in table 1. In the case of *80/20* and *COM* Quadrant 4 performs worse than Quadrant 2, but this may be an artefact of the number of hidden items being greater than the number of recommendations in some cases.

However, since Quadrants 1 and 3 outperform Quadrants 2 and 4 respectively in both *10Hidden* and *80/20* it is clear that a greater number of ratings *adversely* affects performance. This is an extremely counterintuitive result and deserves explanation.

In an attempt to make sense of these results, it is worth recalling that for an active user *u* the approach involves selecting items from the set of items rated by the nearest neighbours of *u*, the candidate set of *u*, CS_u , in the hope that they will also be in *u*'s hidden set, H_u . For this reason it is instructive to consider $P(i \in H_u | i \in CS_u)$, the probability that an item *i* is in the hidden set given that it is in the candidate set. Of course, in reality items are not selected randomly from the candidate set, but on the basis of how frequently they have been rated. Nevertheless, this probability helps to shed light on the problem since it draws attention to the difficulty of finding hidden items in large candidate sets. Via Bayes' theorem, the probability of interest can be expressed as

$$P(i \in H_u | i \in CS_u) = \frac{P(i \in CS_u | i \in H_u)}{P(i \in CS_u)} \times P(i \in H_u). \quad (6)$$

Since $P(i \in H_u)$, the probability that a randomly selected item will be in the hidden set depends only on the size of the hidden set, this probability will be the same for all users and so need not be considered further. $P(i \in CS_u)$, the probability that a randomly selected item will belong to the candidate set, is just the ratio of the number of items in the candidate set to the total number of items. Finally, $P(i \in CS_u | i \in H_u)$, can be estimated in terms of the probability that an item will be in the set of items rated by the nearest neighbours of u given that it is in the non-hidden items rated by u , which in turn can be estimated by ratio of the corresponding frequencies.

Consider now a typical user in Quadrant 1, v say, and a typical user in Quadrant 2, w say. Since the cosine similarity behaves in a similar way to a simple overlap measure [see 9] and since the average cosine similarity would be the same for these two users, the ratio of items co-rated by v and v 's neighbours to the number of items rated by v would be similar to the corresponding ratio for w . This means that the estimates for $P(i \in CS_v | i \in H_v)$ and $P(i \in CS_w | i \in H_w)$ would be similar. By contrast, the candidate set for w will be much larger than that for v since w has a larger number of ratings and so w 's neighbours will also have a larger number of ratings than v 's neighbours since v and w have the same average similarity. This means that $P(i \in CS_w)$ will be greater than $P(i \in CS_v)$ and so, via equation (6), $P(i \in H_v | i \in CS_v)$ will be greater than $P(i \in H_w | i \in CS_w)$. That is, the probability of finding an item in v 's hidden set given that it is in v 's candidate set is greater than the corresponding probability for w . See table 6 for an illustration of the above example using the mean values of the users in Quadrants 1 and 2 to estimate the relevant probabilities.

Table 6. Estimated Probabilities for Quadrants 1 and 2

	Quadrant 1 Mean Values	Quadrant 2 Mean Values
$P(i \in CS_u i \in H_u)$	32.69/35.09 = 0.9316	127.78/131.73 = 0.97
$P(i \in CS_u)$	902/3706 = 0.2434	1918/3706 = 0.5175
$P(i \in CS_u i \in H_u)$	0.9316 / 0.2434	0.97 / 0.5175
$P(i \in CS_u)$	= 3.83	= 1.87

This goes some way to explaining why the performance is better for Quadrant 1 than Quadrant 2. Basically, since users in Quadrant 2 have a greater number of ratings than users in Quadrant 1 they have a larger candidate set and hence it is much more difficult to find the 10 hidden items. Exactly the same kind of reasoning helps explain why Quadrant 3 outperforms Quadrant 4. The size of the candidate set is not so crucial when comparing Quadrants 1 and 3 since they have approximately the same number of ratings. In this case, it is the greater similarity in Quadrant 3 which makes it easier to find a hidden item from a given user's candidate set and hence Quadrant 3 outperforms Quadrant 1. The same reasoning explains why Quadrant 4 outperforms Quadrant 2.

In summary, the greater number of ratings in the case of Quadrants 2 and 4 results in a larger candidate set and hence makes it more difficult to find a small number of hidden items. In effect, there is a bias towards users with a small number of ratings. This is closely related to research by Redpath et al [9] who argued that using standard evaluation metrics when recommending the Top- N items indicates how good algorithms are at identifying items that

users have previously rated rather than how likely users are to rate items highly.

6 RELATED WORK

In this section, relevant research concerning collaborative filtering (CF), the data sparsity issue and evaluation of recommender algorithms are presented.

Early CF systems were semi-automated, requiring the individual user to discriminate between useful and interesting items and have knowledge about the community of people. These systems were either pull-active [2] or push-active [18]. Recent systems use either memory-based or model-based learning algorithms [3].

Providing user-centric predictions has proved difficult for many reasons, chiefly, that certain areas of the dataset are very sparse. This occurs as the dataset increases, as even the most active of users will only have rated a small percentage of the product catalogue. As a result, user preferences and overlaps are difficult to infer. Approaches to alleviate the problem of sparse users have been researched extensively with many different approaches being suggested: including item-based [10, 11], graph-based [19], clustering [20], hybrid recommenders [5] and various dimensionality reduction techniques [21]. Dimensionality reduction methods, such as Singular Value Decomposition (SVD) and Latent Semantic Indexing (LSI) tend to lead to a reduction in accuracy, compared to using the entire dataset, as valuable information is discarded. [5] suggests a hybrid approach: using content to "boost" the CF by predicting ratings for empty spaces in the user-ratings matrix and then applying CF to the dense pseudo-user ratings matrix. This approach still suffers from the sparsity problem, as users that have fewer than 50 items rated produce less reliable pseudo-user rating vectors.

The evaluation of a recommender algorithm should ideally be left to the subjective opinion of the end user. Unfortunately, this is not always possible due to lack of time and resources. In addition there is an abundance of common metrics, which are uncorrelated [12], not standardised for ease of comparison and not widely available as reusable resources [14]. Comparative studies have been conducted [12; 22; 7; 3] to evaluate different collaborative recommender algorithms. The metrics used have differed and therefore the degree of improvement in the end results is not transparent.

The research community recognises that the problems with current evaluation methods and measures must be addressed [23]. In response to this, new and modified metrics have started to appear in the literature [7, 24]. [7] introduces a modified precision, whereby any recommendation generated which is not rated by the test user counts negatively towards the result. [24] introduced the Intra-List Similarity Metric to measure the diversity of recommendations within a list.

7 CONCLUSIONS AND FUTURE WORK

A widely accepted assumption in the research community is that because sparsity is a property of the ratings matrix and all users are considered sparse, that it follows that all users suffer from the sparsity problem. The results in this paper demonstrate that this is not always the case. By defining sparse users in terms of low average similarity with nearest neighbours and low number of ratings, it has been demonstrated that while sparse users do perform

worse for prediction generation, they do not perform worse for Top- N recommendation when using the current evaluation protocols. This counterintuitive result has been explained and empirically verified in terms of a bias towards users with smaller rating vectors.

This result has extremely important consequences for attempts to address the problem of sparsity. In the context of Top- N recommendation when using the current evaluation protocols, attempts to improve results for sparse users are unlikely to be very successful since sparse users actually perform quite well. This, of course, highlights a problem with current approaches to Top- N recommendation which needs to be addressed. In particular, steps are required to develop evaluation protocols and metrics that treat all users fairly.

Future work includes evaluating the quality of the recommendation sets from different algorithms in terms of, not just how many known items or ratings can be predicted correctly but how good the items that were not in the hidden set were.

The conclusion of this report, that not all users suffer from the data sparsity problem, provides the motivation for targeting improvement approaches towards distinct groups of users as opposed to the entire dataset. Work is underway to provide predictions and Top- N recommendations to the appropriate sparse users using content-based, item-based and hybrid approaches.

A combined content and collaborative similarity measure that uses the ratings for co-rated items and calculates a content divergence for all other movies rated, is being developed. It is hypothesized that the combined measure will provide a more detailed measure of similarity for users in Quadrant 1 when generating item predictions: whereas, when producing top- N recommendations for users in Quadrants 2 and 4 an item-based method is being tested to prune candidate item sets of unlikely TP s.

REFERENCES

- [1] Resnick, P., Varian, H.: Recommender Systems. *CACM*. 40(3), 56--58 (1997)
- [2] Goldberg, D., Nichols, D., Oki, B. M., Terry, D.: Using Collaborative Filtering to Weave an Information Tapestry. *Commun. ACM* 35(12), 61--70 (1992)
- [3] Breese, J., Heckerman, D., Kadie, C.: Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In: *UAI-98*, 43--52 (1998)
- [4] Hayes, C., Massa, P., Avesani, P., Cunningham, P.: An On-line Evaluation Framework for Recommender Systems. In: *Workshop on Personalization and Recommendation in E-Commerce* (2002)
- [5] Melville, P., Mooney, R. J., and Nagarajan, R. Content-boosted collaborative filtering for improved recommendations. In *Eighteenth National Conference on Artificial Intelligence* R. Dechter, M. Kearns, and R. Sutton, Eds. American Association for Artificial Intelligence, Menlo Park, CA, 187-192. (2002)
- [6] Anand, S.S., Mobasher, B.: Intelligent Techniques for Web Personalization. In: Mobasher, B., Anand, S.S. (eds.) *LNCS*, vol. 3169, pp. 1--37. Springer-Verlag, Berlin, Germany (2005)
- [7] McLaughlin, M.R., Herlocker, J.L.: A Collaborative Filtering Algorithm and Evaluation Metric that Accurately Model the User Experience. In: *SIGIR '04*. ACM, New York, 329--336 (2004)
- [8] Symeonidis, P., Nanopoulos, A., Papadopoulos, A., Manolopoulos, Y.: Collaborative Filtering Process in a Whole New Light. In: *IDEAS'06*. IEEE Computer Society, Washington, DC, 29--36. (2006)
- [9] Redpath, J., Glass, D. H., McClean, S., Chen, L.: Collaborative Filtering: The Aim of Recommender Systems and the Significance of User Ratings. In Gurrin et al. (Eds): *ECIR 2010*, *LNCS* 5993, pp. 394--406, Springer-Verlag, Berlin, Heidelberg (2010)
- [10] Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based Collaborative Filtering Recommendation Algorithms. In: *WWW'01*. ACM, New York, 285--295. (2001)
- [11] Yildirim, H. and Krishnamoorthy, M. S.: A random walk method for alleviating the sparsity problem in collaborative filtering. In: *RecSys '08*. ACM, New York, NY, 131-138. (2008)
- [12] Herlocker, J., Konstan, J., Terveen, L., Riedl, J.: Evaluating Collaborative Filtering Recommender Systems. *ACM TOIS*. 22(1), 5--53 (2004)
- [13] MovieLens Dataset, 2006, <http://www.grouplens.org/>
- [14] Konstan, J.A., Riedl, J.: Research Resources for Recommender Systems. In: *CHI'99*. (1999)
- [15] Gunawardana, a., Shani, G.: A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *Journal of Machine Learning Research*. Vol. 10, 2935--2962. (2009)
- [16] Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Analysis of Recommendation Algorithms for E-commerce. In: *EC '00*. ACM, New York, 158--167. (2000a)
- [17] Hernandezdelolmo, F., Gaudioso, E.: Evaluation of recommender systems: A new approach. *Expert Systems with Applications* 35(3):790-804 (2008)
- [18] Maltz, D and Ehrlich, K.: Pointing the way: Active Collaborative Filtering. *Human Factors in Computing Systems: Proceedings of CHI'95*, May 1995, pp. 202--209
- [19] Aggarwal, C. C., Wolf, J. L., Wu, K., and Yu, P. S.: Horting hatches an egg: a new graph-theoretic approach to collaborative filtering. In: *ACM SIGKDD'99*. ACM, New York, NY, 201-212. (1999)
- [20] Ungar, L., Foster, D.: Clustering methods for Collaborative Filtering. In: *AAAI Workshop on Recommendation Systems*. California: AAAI. (1998)
- [21] Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Application of Dimensionality Reduction in Recommender System - A Case Study. In *Proceedings of the ACM WebKDD Workshop*. Boston, MA, USA, 82--90. (2000b)
- [22] Huang, Z., Zeng, D., Chen, H.: A Comparison of Collaborative-Filtering Recommendation Algorithms for E-commerce. *IEEE IS*, 22(5), 68-78 (2007)
- [23] McNee, S. M., Riedl, J., Konstan, J. A.: Being Accurate is Not Enough: How Accuracy Metrics have hurt Recommender Systems, In *CHI '06*. ACM, New York, 1097-1101 (2006)
- [24] Ziegler, C., McNee, S. M., Konstan, J. A., Lausen, G.: Improving recommendation lists through topic diversification. In: *WWW '05*. ACM, New York, 22-32 (2005)