University of Dayton
eCommons

Computer Science Faculty Publications | Department of Computer Science

2005

The Good, Bad and the Indifferent: Explorations in Recommender System Health

Benjamin J. Keller
Eastern Michigan University

Sun-mi Kim
Eastern Michigan University

N. Srinivas Vemuri
Virginia Polytechnic Institute and State University

Naren Ramakrishnan
Virginia Polytechnic Institute and State University

Saverio Perugini
University of Dayton, sperugini1@udayton.edu

Follow this and additional works at: http://ecommons.udayton.edu/cps_fac_pub


Part of the Databases and Information Systems Commons, Graphics and Human Computer Interfaces Commons, Numerical Analysis and Scientific Computing Commons, OS and Networks Commons, Other Computer Sciences Commons, and the Software Engineering Commons

eCommons Citation

Keller, Benjamin J.; Kim, Sun-mi; Vemuri, N. Srinivas; Ramakrishnan, Naren; and Perugini, Saverio, "The Good, Bad and the Indifferent: Explorations in Recommender System Health" (2005). *Computer Science Faculty Publications.* Paper 31.
http://ecommons.udayton.edu/cps_fac_pub/31

This Article is brought to you for free and open access by the Department of Computer Science at eCommons. It has been accepted for inclusion in Computer Science Faculty Publications by an authorized administrator of eCommons. For more information, please contact frice1@udayton.edu, mschlangen1@udayton.edu.

The Good, Bad and the Indifferent: Explorations in Recommender System Health

Benjamin J. Keller and Sun-mi Kim
Department of Computer Science
Eastern Michigan University, Ypsilanti, MI 48108
bkeller@emich.edu, skim8@emich.edu

N. Srinivas Vemuri and Naren Ramakrishnan
Department of Computer Science
Virginia Tech, Blacksburg, Virginia 24061
nvemuri@vt.edu, naren@cs.vt.edu

Saverio Perugini
Department of Computer Science
University of Dayton, Ohio 45469
saverio@udayton.edu

December 7, 2004

Introduction

Our work is based on the premise that analysis of the connections exploited by a recommender algorithm can provide insight into the algorithm that could be useful to predict its performance in a fielded system. We use the jumping connections model defined by Mirza *et al*. [6], which describes the recommendation process in terms of graphs. Here we discuss our work that has come out of trying to understand algorithm behavior in terms of these graphs. We start by describing a natural extension of the jumping connections model of Mirza *et al*., and then discuss observations that have come from our studies, and the directions in which we are going.

Jumping Connections Revisited

Mirza *et al*. define a model that describes algorithms based on user-similarity, such as the nearest neighbor algorithms described by Herlocker *et al*. [2]. The ratings data correspond to a directed, weighted, bipartite graph called the *rating* graph in which vertices are users and items, and arcs are the ratings. Fig. 1 shows the subgraph of a rating graph involved in computing a nearest neighbor prediction of item a for user p. A *social network* is formed the ratings by using the users as vertices, and using a similarity measure (and possibly filtered by a threshold) to determine the edges. Mirza *et al*. use commonality of ratings to define a *hammock* measure of similarity where a threshold can be used to indicate the minimum number of ratings that must be common. The *recommender* graph is formed by adding the ratings back into the social network, and is the space in which predictions are computed. Fig. 2 shows the recommender graph for the neighborhood in Fig. 1.

Sarwar *et al*. [8] introduce *item-based* nearest neighbor algorithms, which in the graph model is just a dual construction. The item-based analogue to the social network is formed by using the dual similarity relationship between items, which forms an *artifact* network. The artifact network can be extended to a (item-based) recommender graph by adding the ratings as shown in Fig. 3.

Observations

Our work coming out of the experiments reported by Mirza *et al*. has dealt with analysis of the social and artificial networks, and trying to relate graph structure to algorithm performance. There are three key points to our work so far: (1) ignoring ratings is not useful in studying algorithms that employ them, (2) there is some significance of the graph structure to accuracy, but (3) what that influence is, is not yet clear.

Ratings change everything. The experiments described in Mirza *et al*. [6] showed that there is a place for studying recommendation based on commonality of

Copyright is held by the author/owner(s).
Workshop: Beyond Personalization 2005
IUI'05, January 9, 2005, San Diego, California, USA
http://www.cs.umn.edu/Research/GroupLens/beyond2005

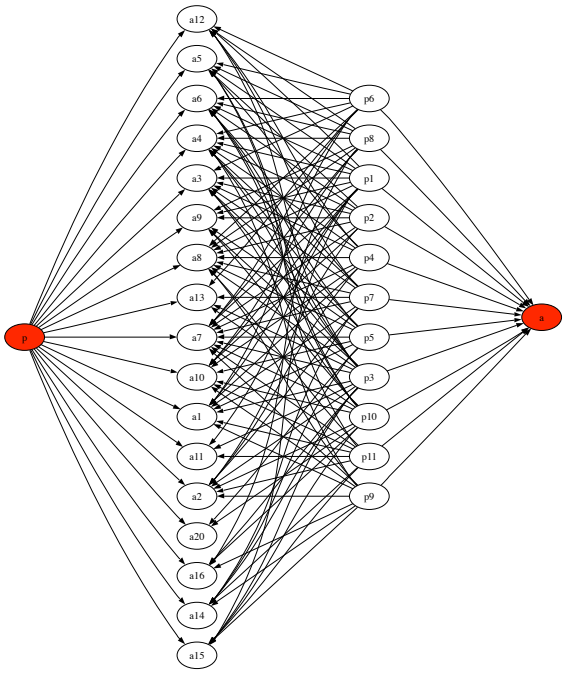


Figure 1: Subgraph of rating graph for prediction of item a for user p.

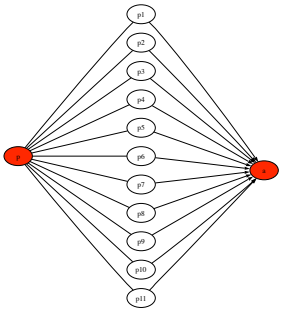


Figure 2: Subgraph of user-based recommender graph for prediction of item a for user p.

ratings. In the case of movies, we know that a few users tend to rate a lot of movies, so in a user-based algorithm, these users play an important role in forming the social network by making it possible for users with relatively few ratings to get recommendations (see the results on the minimum rating constraint in Mirza *et al*. [6]). Looking at commonality therefore allows us to understand how what people rate is important. However, if we look at the properties of the graphs induced by commonality and those induced by similarity measures based on the ratings, we see that the ratings change everything.

Just to illustrate the point, consider the plots of degree correlation for the social network based on commonality and the social network based on the Pearson r correlation shown in Fig. 4 and Fig. 5. Degree correlation measures the similarity of the degrees of adjacent vertices [7]. We should first caution that the x-axis in these plots are different, and so the comparison is slightly dangerous (which is part of the point). Both plots show the effects of filtering the edges of the graph by increasing a threshold (minimum items in common, and minimum correlation). The figure shows that the edges in the commonality-based social network are only between vertices of dissimilar degree, which suggests that users who rate many movies are serving as hubs for users who have not rated many movies. The plot for the Pearson similarity network, on the other-hand shows a phase-shift in the connectedness of the graph that indicates most connections between users of dissimilar degree have low correlations.

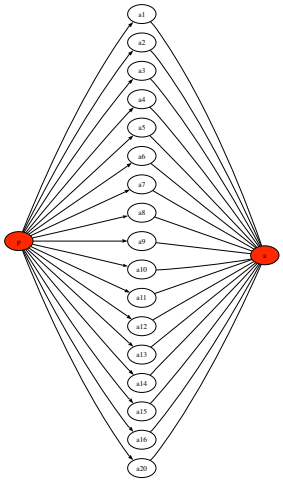


Figure 3: Subgraph of and item-based recommender graph for prediction of item a for user p.

Neighborhood structure does affect predictive accuracy. A question that had been posed to us in several settings was whether we could say something interesting about predictive accuracy through the graph structure. We first attempted a "jacknife" study using the 100,000 rating MovieLens data set, where for each user-rating pair we cut out a user's rating of an item and then predicted the rating. The results were inconclusive, so we took a different approach, which led to Srinivas Vemuri's thesis [9].

The approach in this case was to introduce a structural filter on the neighborhood and then measure the affect in terms of predictive accuracy. The filters applied were basically the requirement that two neighbors of the user are only kept if they are neighbors of each other — thus

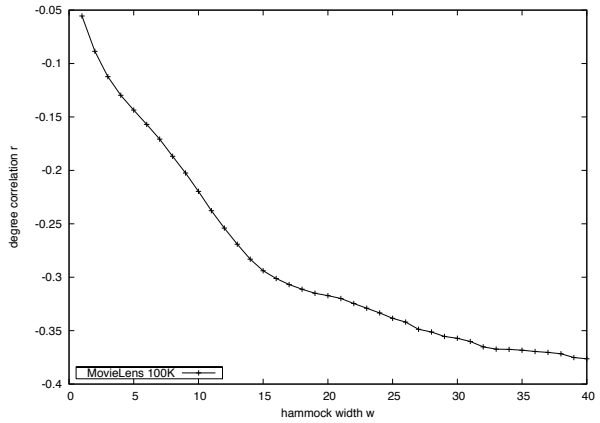


Figure 4: Plots of degree correlations for hammock social network (based on commonality) of MovieLens 100,000 rating data set.

forming a triangle. Vemuri was able to demonstrate an improvement in predictive accuracy (see Fig. 6), but at the price of loss of coverage. However, he also defined an approach that reweights the neighbors based on their involvement in triangles that produces similar results without the loss of coverage.

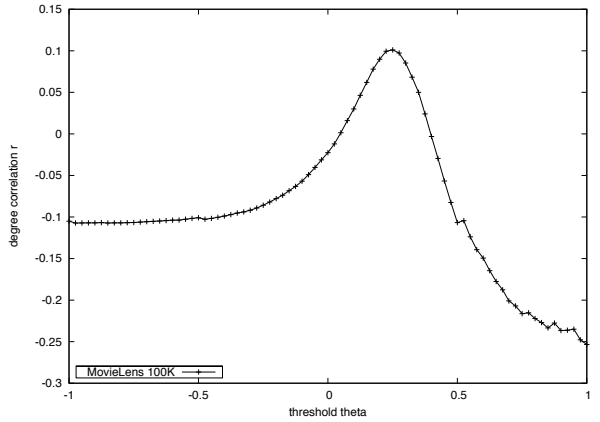


Figure 5: Plots of degree correlations for Pearson social network of MovieLens 100,000 rating data set.

Good neighborhoods don't always have good structure. The use of the triangular filter is based on the assumption that having better connected neighbors necessarily leads to better predictions, or, at least, eliminates the bad ones. However, further analysis of the results of applying the filters shows that the filters are somewhat indiscriminate, and make some predictions better, some worse, some impossible, and have no affect on the majority of predictions. Fig. 7 shows a typical configuration (although smaller than most) of a neighborhood affected by the triangle filters. In some cases, the loss

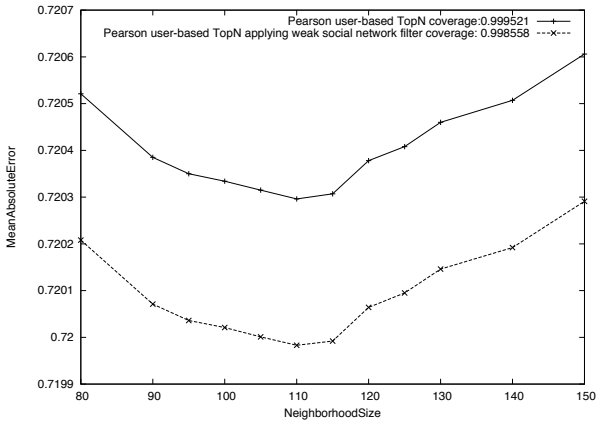
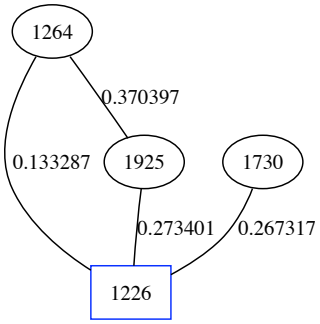


Figure 6: MAE versus neighborhood size for top-N user-based Pearson algorithm with and without triangles for 1 million rating MovieLens data set.

of the neighbor not involved in a triangle improves accuracy and in others makes it worse. The reason that the overall predictive accuracy improves is that the number of bad predictions lost exceeds the number of good predictions made worse or lost. However, the number of predictions changed is small (1% or less), unless a high threshold is used to define the triangles.



Prediction- 4.69065 Rating- 4

Figure 7: Typical neighborhood for a prediction affected by triangle filters.

Directions

The following describes work that is ongoing following the observations described above.

Recommendation Metrics

In considering the outcome of Vemuri's work on filtering neighborhoods, another question arose concerning whether the improvement in mean absolute error was significant. In particular, the issue of whether users would notice the minor improvement was raised. Of course, the problem with predictive accuracy as a metric of recommendation is that it has little to do with the way in which recommendations are typically presented. A user is presented a top-N list of items in decreasing order by prediction, and, in this setting, an error in prediction is only significant if it is noticeable by the user (either before or after the fact). Therefore, we are looking at recommendation metrics in terms of observability of errors by a user.

In looking at recommendation list metrics, we assume that we are measuring error over a test set of items that the user has rated (or ranked). Therefore, we can form a list of the user's ratings for these items, ordered by the prediction of the algorithm for each item. For instance, if the user rated five items 5, 5, 4, 3, 1, 1 and the algorithm predicted that they would be rated 3, 4, 4, 5, 2, 3, then we would consider the list 3, 5, 4, 5, 1, 1. A metric then measures the cost of sorting the list so that the ratings are ordered properly. The simplest metric is the count of the number of inversions, which is the cost of performing a bubblesort on the list. This is not a novel approach, since inversions are the basis of Kendall's tau, however, it matches the intuition behind comparing top-N lists. (Fagin *et al*. have shown Kendall's tau [4] is equivalent in a meaningful way to other reasonable choices by which top-N lists could be compared [1].) Vemuri [9] has also suggested counting the number of inversions between sorted runs of ratings as an alternative.

The problem of comparing comparing recommendation lists is much more complex than comparing the order, because the lists are presented in pages, and a user will also only view a prefix of the recommendation list [5, 3] Therefore, we might also consider factors such as the number of items presented per page, the total number of items (or pages) viewed by the user, and the user's tolerance for errors. In its simplest form, user tolerance can be modeled as an equivalence between rating values, since this would hide short swaps. However, a user's tolerance for longer swaps might be affected by whether they cross a page, or whether they include or exclude an item from the prefix of the list. It is not clear the extent to which these might be factors that are important to consider in the metric, and we are working on defining a user study that might help understand what a user might be able to observe (or care about).

Local Health

All of this work has led us in the direction of studying the "health" of a recommender system, and to begin with the local health of the system. We consider the *local* health of a recommender system as any property of the system that could affect the user's perception of the system, and *observability* by the user is a key property. Our goal is to define the user observable properties of recommender systems, and to characterize the underlying properties of the algorithm and data that lead to pathologies observable by the user. We concentrate on user observable properties of the recommendation list, including list accuracy, list stability, and variability of new items. This is the topic of Sun-mi Kim's research.

We have started by exploring the issue of what makes a good neighborhood from the standpoint of recommendation list error. To do this, we have identified users who have sufficient diversity in their use of ratings (an entropy value of 2 or more) and have either good or bad inversion rates, and have been studying their neighborhoods. We are starting with the obvious pathologies of the neighborhoods that lead to errors, and hope to find graph properties that we can use as measures of neighborhood quality.

For the other properties, we are following a similar approach to find alternative graph-based metrics that are descriptive. As an example, for novelty, we can define *bridge length* metrics based on the number of ratings that are required to add certain items to the recommendation list by bridging to a new neighbor who has rated the items. In some sense this is like *potential* coverage; coverage being a measure of how many of the total items can be recommended to the user [3].

Conclusion

Overall, our work is part of a larger agenda to be able to characterize the healthy properties of recommender systems. We believe that the graph models provide a useful framework for this study by focusing attention on the connections that are used in the computations. Both the work of Mirza [6] and Vemuri [9] already support this contention. (We should acknowledge that the term *recommender system health* came to us from Joe Konstan.)

1. R. Fagin, R. Kumar, and D. Sivakumar. Compar-

ing top k lists. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms in Artificial Intelligence*, pages 28–36, 2003.

2. 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.
3. J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM TOIS*, 22(1):5–53, 2004.
4. M. Kendall and J. D. Gibbons. *Rank Correlation Methods*. Edward Arnold, London, 1990.
5. S. K. Lam and J. Riedl. Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on World Wide Web*, pages 393–402. ACM Press, 2004.
6. B. J. Mirza, B. J. Keller, and N. Ramakrishnan. Studying recommendation algorithms by graph analysis. *Journal of Intelligent Information Systems*, 20(2):131–160, March 2003.
7. M. Newman. Assortative Mixing in Networks. *Phys. Rev. Lett.*, Vol. 89(20):208701, 2002.
8. B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-Based Collaborative Filtering Recommendation Algorithms. In *WWW'10, Proceedings of the Tenth International World Wide Web Conference*, pages 285–295, 2001.
9. N. S. Vemuri. Linking accuracy of recommendation algorithms to objective measures. Master's thesis, Eastern Michigan University, 2004.