Influence in Ratings-Based Recommender Systems: An Algorithm-Independent Approach

Al Mamunur Rashid George Karypis John Riedl*

Abstract
Recommender systems have been shown to help users find items of interest from among a large pool of potentially interesting items. *Influence* is a measure of the effect of a user on the recommendations from a recommender system. Influence is a powerful tool for understanding the workings of a recommender system. Experiments show that users have widely varying degrees of influence in ratings-based recommender systems. Proposed influence measures have been algorithm-specific, which limits their generality and comparability. We propose an algorithm-independent definition of influence that can be applied to any ratings-based recommender system. We show experimentally that influence may be effectively estimated using simple, inexpensive metrics.

1 Introduction

Sociologists have long tried to characterize the influence of a person in a social network of many people [1]. Identifying the influential people can bring twin advantages to those who study group dynamics: (1) The influential people can be directly studied, yielding insight since their choices may be predictive of group choices; or (2) The influential people may be influenced to change the behavior of the group. Many social networks are formed and maintained through informal, qualitative, and unobserved interactions. Capturing data about these interactions is difficult, and the act of capturing those data may change the social interactions themselves.

Collaborative Filtering (CF) recommender systems [2, 3, 4] base their decisions on the opinions of users. In contrast to other social networks, recommender systems capture interactions that are *formal*, *quantitative*, and *observed*. The social network can be analyzed directly through data already captured in the computer system.

Past research has demonstrated that analyzing the social network can provide leverage in influencing the group [5]. The analysis performed in these studies is based on a deep investigation of the characteristics of one particular recommender algorithm, the well-known user-user nearest neighbor algorithm [2]. Careful analysis of this type has many advantages, but one key disadvantage: it is tied closely to the details of the algorithm. In principle, similar techniques could be applied to other algorithms, but doing so would be laborious, and the resulting influence measure only applies to algorithms that work precisely according to the details of the analysis. Since many commercial operators tweak the operation of the recommender in many ways to fit the needs of their business, this analysis may not apply in practice. Further, the resulting measures of influence would be unlikely to be comparable between different algorithms, since they have been produced through very different techniques.

A key goal of the present research is to identify a measure of influence for recommender systems that is applicable to any ratings-based recommender system, independent of the particulars of the algorithm. Such a measure would allow for consistent, black-box analysis of influence.

2 Related Work

2.1 Recommender Systems. Resnick, et al. [2] introduced an automatic collaborative filtering algorithm based on a k-nearest neighbors (kNN) algorithm among users; this algorithm is now called *user-user* CF. The *user-user* algorithm we use in this paper is a version of the original kNN algorithm, tuned to achieve best known performance. Sarwar et al. [4] proposed an alternative kNN CF algorithm based on similarity among items. This variant is often called *item-item* CF. Breese et al. [3] have divided a number of CF algorithms into two classes: memory-based algorithms and model-based algorithms. Over the years many other algorithms were proposed including ones based on SVD, clustering, Bayesian Networks [3]. We focus on the *user-user* and *item-item* algorithms in this paper because they are the most common in existing systems.

2.2 Social Networks and Influence. A Social network is a form of graph delineating relationships and interactions among individuals. Finding the important nodes in such graphs has been an object of interest to sociologists for a long time. One proposed measure for importance is *centrality* [1]. Two examples of "centrality" measures are "degree centrality", which treats high degree nodes as important, and "distance central-

*Department of Computer Science & Engineering, University of Minnesota, Minneapolis, MN-55455, {arashid, karypis, riedl}@cs.umn.edu

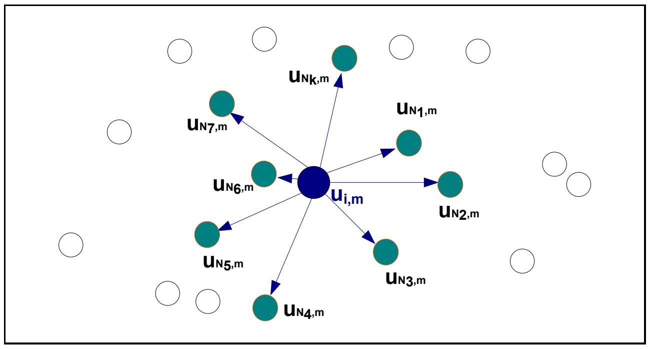

Figure 1: Showing the notion of *in-links* for the k closest neighbors of u_i. Here, prediction is being computed for the (user, item) pair, (u_i, m).

ity", which treats nodes with short paths to many other nodes as important [1] . Kleinberg's HITS [6], and Brin and Page's PageRank [7] algorithms for ordering nodes in a graph of web are based on social network principles.

Domingos et al. [5] have studied the problem of choosing influential users for marketers who wish to attract attention to their products. They show that selecting the right set of users for a marketing campaign can make a big difference. Kempe et al. [8] focus on a collection of models widely studied in social networks, as well as the models in [5], under the categories: Linear Threshold Models, and Independent Cascade Models.

Our research also investigates influence in social networks. Like Domingos et al. we focus on networks in recommender systems. We extend their research to general measures of influence that are independent of the particular recommender algorithm being used.

3 Defining Influential Users in CF Systems

We first discuss the data used in this project, then analyze a popular CF algorithm to understand a possible formation process of influential users, and then try different ways to set the definition.

3.1 The Data. We have used a publicly available dataset from *www.grouplens.org*. The dataset is a fraction of the usage data drawn from MovieLens (*www.movielens.org*), a CF-based online movie recommendation system. It contains 6,040 users, 3,593 movies, and about one million ratings on a 5-star scale. Each user has rated at least 20 movies in the dataset. We have partitioned this data into training and test sets by a random 80%/20% split.

3.2 The *User-User* Algorithm. The most widely cited and arguably the most commonly used CF algorithm in research is a kNN-based algorithm. In this scheme the users' preference data is represented in a $n \times m$ user-item matrix for a system with n users and m items, where the (i, j)-th entry of this matrix stands for the user u_i's rating on item j, or null, depending on whether the user u_i has rated the item j, or not, respectively. The *user-user* algorithm can be thought of working in two stages. In the first stage, similarities between every pair of users are computed and are stored as a model. Although many different formulations are possible for similarity weight calculations, the *GroupLens* [2] proposed mechanism is the Pearson correlation coefficient. Accordingly, the similarity weight between two users, u_i, and u_j is measured by equation 3.1:

$$W_{ij} = \frac{\sum_{k\in I}(R_{ik} - \overline{R}_i)(R_{jk} - \overline{R}_j)}{\sqrt{\sum_{k\in I}(R_{ik} - \overline{R}_i)^2 \sum_{k\in I}(R_{jk} - \overline{R}_j)^2}} \tag{3.1}$$

where I is the set of items rated by both of the users, R_{ik} is user u_i's rating on item k, and $\overline{R}_i$ is the average rating of u_i. Using this similarity metric, the next step, prediction generation, is carried out as follows. Prediction on item a for user u_i is computed by picking k nearest users who have also rated item a, and by applying a weighted average of deviations from the selected users' means:

$$P_{ia} = \overline{R}_i + \frac{\sum_{u=1}^{k}(R_{ua} - \overline{R}_u)W_{iu}}{\sum_{u=1}^{k} W_{iu}} \tag{3.2}$$

3.3 Some Plausible Influence Metrics Based on Prior Work. We can now propose several influence metrics. One type of metric is motivated by *targeted marketing*. Another type of metric exploits connections between users based on similarity.

3.3.1 Expected Lift in Profit: Network Values. This approach, as outlined in [5], is based on the goal of *targeted marketing*. In this scheme, users who can yield the most *expected lift in profit* by making a cascading adoption of a product happen, are considered as influential users. Domingos et al. [5] have applied this idea on a recommendation system dataset based on the *user-user* CF algorithm described in the last section.

The probabilistic model in [5] is based on the *Markov Random Fields*, which requires the neighbors be symmetric; i.e., two users are neighbors to each other if one of them is a neighbor to the other. The authors mention that in a kNN-based CF system, this might not hold. Again, *ELP_Network_Value* is tied to a particular product; more specifically, it is specific to a set of features of the product being marketed. Translating this issue into the RS domain, *ELP_Network_Values* are specific to particular genre vectors. Thus a user's *ELP_Network_Value* will differ for movies with different genre vectors.

3.3.2 Network Structure: Similarity Links. By closely observing the process of neighbor-selection, we notice some network structure that could facilitate in forming a definition for influential users. Figure 1 demonstrates a situation where the system is computing a prediction on item m for user u_i . In order to do so, it selects top k neighbors who also have rated the item m. Now we can imagine directed edges from u_i towards each of the k neighbors.

Equations 3.3 and 3.4 show the updated *authority* and *hub* equations. In order to consider the fact that all the links may not of same weight, we have incorporated a weight term similar to [9] to the basic HITS [6] equations. Here the conditional probability, $p(i|j)$ refers to the degree of user u_j's presence indicating user u_i's presence.

$$a(i) = \sum_{j \to i} p(i|j)h(j)W_{ij} \tag{3.3}$$

$$h(i) = \sum_{i \to j} p(j|i)a(j)W_{ij} \tag{3.4}$$

We can use this modified *authority* to represent influence.

The drawback of this scheme of influence, however, is algorithm dependence: the network structure captured here is very much algorithm-specific; and, for other algorithms, the structure might not be as apparent. In order to derive a definition that is generic enough, yet simple, we use the *Hide-one-User* approach discussed next. The fundamental concept with this approach is figuring out which user causes the largest cumulative change of prediction in the system.

4 Algorithm-Independent Influence

These metrics define influence as the amount of effect a user has over others via the predictions they receive. One way to observe this effect is to exclude a user and measure the net changes in predictions caused by the removal.

The idea: Let U be the set of available users in the system, M_U be the model built with the preference data of this set of users. We call NPD_{u_i} (*Number of Prediction-Differences*) as the number of times the following expression holds true:

$$|P_{ja}(M_U) - P_{ja}(M_{U-\{U_i\}})| \geq \delta, \forall j \neq i$$

Here, $P_{ja}(M_U)$ is the prediction on item a for the user u_j using the model M_U, δ is a threshold that can be tied with the smallest prediction change perceivable to the users via the available user interface. As an example, smallest prediction change a MovieLens user would notice is 0.5 or a half-star. In essence, the expression for NPD_{u_i} says how many times the predictions would change beyond some threshold if we build the model without the user u_i. NPD_{u_i} is the influence level of user u_i. There is a problem with NPD_{u_i}: if the group of users, who get affected by u_i's removal, need predictions on many items, u_i could exhibit possessing a large NPD_{u_i}. To overcome this problem, we propose another version of this definition and call it $NUPD_{u_i}$. $NUPD_{u_i}$ counts the number of unique users whose predictions' got changed by at least the threshold amount as we keep the i-th user out during model-building.

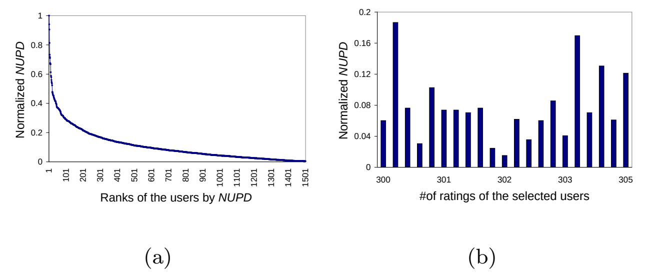

Figure 2: (a) Distribution of influence. (b) *NUPD* values of a group of 20 users who have rated almost the same number of items.

As is evident from the definition of $NUPD_{u_i}$, it is equally applicable to any CF algorithm, provided that we have the historical data to compute it from.

Notice that a straightforward computation of *NUPD*s can become very expensive; if we are to compute *NUPD* online or in a regular basis, we need to find a cheaper way. Section 6 details such an endeavor.

4.1 The Nature of Influence. Figure 2(a) shows normalized *NUPD* values of the top 1500 influential users and highlights the fact that only a handful of the users possess high influence. This is true for both *authority* and *NUPD* measures. The shapes demonstrate the power-law or a Zipf-like distribution. A similar shape is reported in [5] for *ELP_Network_Value*s.

Note that the correlation between *authority* and *NUPD* is 0.96.

5 Building a Predictive Model

As stated before, *NUPD* suffers from a drawback: the computation is quite time consuming. In order to circumvent this limitation, we seek a predictive model that can provide users' influence levels on the fly while maintaining good accuracy.

Although the correlation coefficient between *NUPD* and the number of ratings is 0.75, figure 2(b) shows that the amount of influence can vary widely between users who have rated approximately the same number of movies. This suggests we look for a model that can account for factors not captured by the number of ratings.

In the following section we compile a list of qualitative factors that seem to affect influence levels.

5.1 Qualitative Factors

Number of ratings: This is the most immediate factor one would possibly come along with. If a user rates more items, she has a greater chance to be close to many users. Moreover, such a user can be useful to many users who are looking for recommendations for a wide variety of items.

Degree of agreement with others: This measure attempts to estimate on average how much a user agrees to the average opinion of others: $1/k \sum_{a=1}^{k} |R_{ia} - \overline{R}_a|$. This expression computes the extent to which the user u_i's ratings are swayed from each of the corresponding item's average rating.

Rarity of the rated items: This is a measure very similar to that of the Inverse Document Frequency (IDF), which penalizes frequent items, as they are considered to have little discriminating power: $1/k \sum_{j \in I_{u_i}} 1/freq(j)$; where, I_{u_i} is the set of items that user u_i has rated.

Standard deviation in one's rating: This amounts to the degree a user's ratings deviate from her rating-average. The implication is that a higher standard deviation contributes a greater value through the term, $(R_{ik} - \overline{R}_i)$ in equation 3.1.

Degree of similarity with top neighbors: This is the average similarity weight of the top k neighbors of a user u_i: $1/k \sum_{j=1}^{k} W_{ij}$. This factor can be associated with two opposing implications: users having higher values from this expression might be able to exert more effect to be influential; whereas, a user might be easier to replace if she is very similar to a number of other users.

Aggregated popularity of the rated items: If the sum of the popularities of the rated items is high enough, the user has a greater chance to have overlapped items with many users.

*Aggregated MoviePopularity*Entropy*: Entropy of a movie simply indicates the dispersion of the ratings it received. Multiplying this with the popularity of the movie gives a measure that tries to balance between popularity and variance.

5.2 The Regression Model

We chose to use SVM Regression (SVR) for our modeling. SVMs follow the *Structural Risk Minimization Principle* which seeks to minimize an upper bound on the generalization error rather than the principle used in most of the learning machines: *Empirical Risk Minimization Principle*– minimizing the training error. Hence, SVMs have been showing better generalization in many results. Although most of the practical usages for SVMs used to be in classification problems, SVMs have been extended to solve non-linear regression problems, mostly because of the introduction of the *ε-insensitive loss function* [11]; and the resulting regression method called $\varepsilon - SVR$.

We have tried various kernel functions to perform the non-linear mapping from the input space to the feature space. However, the radial basis function (RBF) produced the best regression result. In order to select the values of the parameters, C and ε, a cross-validation approach was carried out.

We have randomly selected 2416 users (40% of the total) and partitioned them into training and test sets by a 8:2 split. libsvm[10] was used to generate regression models using the following: the seven factors outlined before as predictors (independent variables), an RBF kernel, $\varepsilon - SVR$, and the parameters, C and ε. The model gave a squared correlation coefficient of 0.94. Figure 3 shows the prediction performance by plotting predicted *NUPD*s against the corresponding actual *NUPD*s taken from the test-set. A five-fold cross validation was carried out to ensure the results' validity. Table 1 has the regression results as well as a few statistics of the actual *NUPD* values in the test set, averaged over the five folds.

6 Influence in an Item-based Algorithm

We now turn to how the influence picture looks when using another prediction algorithm in order to see how algorithm-dependent our measures are.

The *item-item* Algorithm. The kNN based CF algorithm proposed in [4] is different in many ways than the user-based algorithm we have addressed so far. The algorithm first builds the model by computing item-item similarities. [4] proposed adjusted cosine measure for estimating the similarity between two items i, and j:

$$s_{i,j} = \frac{\sum_{u \in U} (R_{u,i} - \overline{R}_u)(R_{u,j} - \overline{R}_u)}{\sqrt{\sum_{u \in U} (R_{u,i} - \overline{R}_u)^2 \sum_{u \in U} (R_{u,j} - \overline{R}_u)^2}}$$

Prediction for the (user, item) pair, (u, i) is computed as: $\sum_{all\_similar\_items,N} (s_{i,N} * R_{u,N}) / \sum(|s_{i,N}|)$.

We could not employ *authority* on this algorithm, as it is not quite straightforward to establish di-

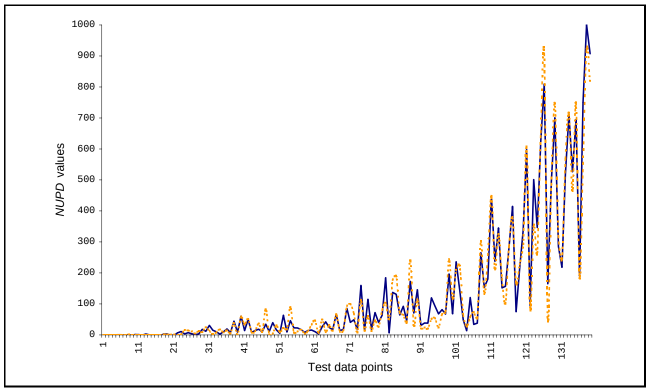

Figure 3: Performance of SVM regression for *NUPD* on *user-user* algorithm. The dotted line shows the actual values; whereas, the continuous line represents the predicted values.

rect edges between users. We could not compute *ELP_Network_Value*s on this algorithm either, since *ELP_Network_Value*s involve the notion of how *neighbors* affect a user, and corresponding probability computations based on this. However, applying *NUPD* by *Hide-one-User* method was easy. We have estimated *NUPD*s for the same set of users we have selected for the user-based approach. Modeling with $\varepsilon - SVR$ gave a very good performance: squared correlation coefficient was 0.989.

7 Conclusion

In this paper, we have continued the investigation into influence in recommenders begun in [5]. We have shown that how many opinions a user expresses is an important component of influence, but not the whole story. We have defined several plausible influence metrics and shown that in general, they correlate strongly.

We believe our proposed metric, *NUPD*, is explainable both to researchers and operators of recommender systems. *NUPD* is also algorithm independent—it applies to any recommender system algorithm that makes predictions. *NUPD* is computationally inefficient. However, we have demonstrated how to build dataset- and algorithm-specific regression models that allow for the rapid, accurate estimation of a user's influence.

Much remains to be done. Research is needed to understand how the *role* of influence changes it. For instance, when influence is used to help retailers sell products it may have very different characteristics than when it is used to encourage community members to contribute opinions. Another rich area of research is in interfaces for communicating influence to community members. The interface is likely to impact both the interpretation of influence and its effectiveness in changing behavior.

References

[1] S. Wasserman, K. Faust, *Social Network Analysis: Methods and Applications*, Cambridge University Press, (1994).

[2] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl, *Grouplens: An open architecture for collaborative filtering of netnews*, in Proceedings of CSCW 1994, ACM SIG Computer Supported Cooperative Work, 1994.

[3] J. S. Breese, D. Heckerman and C. Kadie, *Empirical analysis of predictive algorithms for collaborative filtering*, in Proceedings of the Fourteenth Annual Conference on Uncertainty in AI, July 1998.

[4] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, *Item-based collaborative filtering recommendation algorithms*, in Proceedings of the 10th International World Wide Web Conference (WWW10), Hong Kong, May 2001.

[5] P. Domingos and M. Richardson, *Mining the Network Value of Customers*, Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, 2001. ACM Press, pp. 57–66.

[6] L. Kleinberg. *Authoritative sources in a hyperlinked environment*, Journal of the ACM, 46, 1999.

[7] L. Page, S. Brin, R. Motwani, and T. Winograd. *The PageRank citation ranking: Bringing order to the web*, Technical Report, Stanford University, Stanford, CA. 1998.

[8] D. Kempe, J. Kleinberg, and Tardos, *Maximizing the spread of influence through a social network*, in Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, Washington DC, 2003, pp. 137–146.

[9] K. Wang, and M. Y. T. Su, *Item Selection by "Hub-Authority" Profit Ranking*, in SIGKDD '02, Canada.

[10] C. C. Chang, and C. J. Lin, *LIBSVM : a library for support vector machines*, 2001.

[11] V. N. Vapnik, *The Nature of Statistical Learning Theory*, New York, Springer-Verlag, 1995.

Table 1: Regression results on both CF algorithms

		User-User	*Item-Item*
Regression Performance	MAE	15.26	30.6
	Sq. corr. coeff.	0.94	0.99
	MSE	1036	2252.6
NUPD Statistics in Test Set	Avg.	81.57	405.6
	Min	0	0
	Max	980	2487
	StdDev	123.25	454.6