Providing Recommendations in an Open Collaboration System

Fillia Makedon[1,2], Sheng Zhang[1], Zhengyi Le[1], James Ford[1,2],
and Euripides Loukis[3]
[1]Department of Computer Science, Dartmouth College
[2]Department of Computer Science and Engineering, University of Texas Arlington
[3]Department of Computer Science, University of the Aegean, Greece

Abstract

Open Collaboration (OC) is a tool being developed to support a variety of electronic collaboration needs. In OC, group and role information is propagated in a peer-to-peer fashion, and peers can share data resources with any peer who is a member of an appropriate group or role. In this paper, we introduce our current work on incorporating a recommendation component in OC. The objective of this component is to help users find the most reliable, valuable, important, and interesting information quickly and easily. Four implemented recommendation algorithms (User-based, Item-based, Singular Value Decomposition based, and Non-negative Matrix Factorization based algorithms) in our recommendation component are discussed.

Keywords: Open collaboration, recommender systems, collaborative filtering

1. Introduction

An Open Environment (OE) implies that a stranger may need to join a group collaboration where the entities are diverse and autonomous. It is important that an OE provide secure data sharing, access to data (or other resources), storage of data, and transmission of data. Thus, an effective OE will enable secure collaboration mechanisms that permit (a) on demand formation of collaboration groups, (b) the ability for qualified strangers to join a collaboration group, (c) the ability to operate in a totally distributed setting without a central administration, and (d) guarantees of privacy and security control by the users of the collaboration system.

Our system, which is called "OC" for "Open Collaboration", is an open source resource sharing and collaboration system. Our approach is based on existing Automated Trust Negotiation (ATN) methods using Peer-to-Peer (P2P) solutions and uses configurable profiles for groups and individuals to enable privacy and security control. ATN provides the tools need to help a stranger join existing collaborations without human intervention using digital certificates and associated policies. The use of P2P protocols makes a centralized server unnecessary, and allows any node to be both a resource consumer and a resource provider. OC also applies Role-Based Access Control (RBAC) on shared files, which allows for a more flexible and

scalable access authorization solution than traditional Access Control List (ACL) mechanisms.

When there are many resources in an open collaboration environment, it becomes difficult for users to find those that are really valuable and interesting to them. In order to ameliorate such information overload, we introduce a recommendation component in OC. The recommendation component allows users to leave feedback (ratings) on the data resources they have downloaded and used, and to share their ratings with each other. Based on these ratings, the recommendation component predicts which items have the highest likelihood of being useful and valuable to particular users.

Four recommendation algorithms are implemented in our recommendation component. A user-based algorithm (1) and an item-based algorithm (2) make recommendations by exploiting information from those users and items (respectively) that are similar to the current user and the current item. A Singular Value Decomposition (SVD) based algorithm (3) computes a low-dimensional linear model from all observed ratings and uses the computed model for recommendations. Finally, a Non-negative Matrix Factorization (NMF) algorithm (4) also computes a linear model, but enforcing a non-negativity constraint to allow more explainability.

A framework for evaluating the above four collaborative filtering algorithms of the recommendation component has been synthesized. It combines approaches and elements from research on both recommender systems evaluation and technology acceptance models.

2. Open Collaboration (OC)

In an open environment, any interested user may ask to join a collaboration. The traditional approach to joining a collaboration is to let a system administrator review a registration form and any required qualification credentials of the interested user and then create a new account for him or her (or reject the application). This human-interactive one-way authentication is not suitable for dynamic and large-scale applications. If the applicants have questions about the group, more human intervention and delay may be introduced.

OC uses the concept of automated trust negotiation to avoid this administrative overhead. ATN works as follows: an applicant sends a request to join a group to a group recruiter (which may actually be an agent, *i.e.*, a computer program running autonomously and without human input), and the group recruiter sends join requirements back to an applicant. The join requirements may include some attribute-based credentials (*e.g.*, Age $\geq$ 18) and possibly other credentials, such as the electronic equivalent of identity or membership cards. After the applicant receives the requirements, they check their local policy pool to see where any required credentials

are considered sensitive. If sensitive, they will have a release policy that protects this credential. In this case, the applicant sends back a counter-request indicating the requirements for releasing this credential. If the recruiter can satisfy the counter-request (which may involve further derivative request), the applicant will send the requested credentials. Once the recruiter receives and verifies those credentials, the applicant is issued a credential indicating that they are a member of the group. The entire procedure can be executed automatically.

In a specific OE application, a collaborative group may require that some services, *e.g.*, public file sharing, should be open to everybody while other services, *e.g.*, sensitive file sharing, should be open only to a qualified subset of users. In our OC system, we use role based trust management to control data access. When a user joins a group, they are automatically assigned a role, typically as a guest or junior member. If they want to obtain additional roles, they must repeat the application procedure again specifically for a desired role in order to obtain a role certificate specific to that role. Different services may be protected by different policies, some of which ask the requester to present specific role certificates. When required, a user uses their role certificate to request these services, *e.g.*, the downloading of certain files.

Using a peer-to-peer approach removes the need for a centralized server. A "pure" P2P network does not have the notion of clients or servers, but only equal peer nodes that simultaneously function as both "clients" and "servers" with respect to the other nodes on the network. This model of network arrangement differs from the client-server model, where client communication is usually to and from a central server. Since in many situations in this domain each participant could be both a data provider and a data consumer (*i.e.*, be both a server and a client), P2P meets the need of open environment collaborations very well. P2P networks are also more efficient for data sharing and avoid the single point of failure problem since data are distributed among the peer nodes (if desired, with some level of redundancy).

3. Recommendation Component

Recommendation algorithms are usually classified according to how recommendations are made into the following two categories. **Content-based recommendations**: the user will be recommended items judged to be similar to the ones the user preferred in the past. **Collaborative filtering recommendations**: the user will be recommended items that people judged to have similar tastes and preferences liked in the past. Our recommendation component mainly focuses on collaborative filtering (CF) recommendations because it is the most popular category.

CF algorithms can be further divided into two categories: **memory-based algorithms** [Herlocker et al. 1999, Resnick et al. 1994, Sarwar et al. 2001] and **model-based algorithms** [Canny 2002, Hofmann 2004, Sarwar et Al. 2000, Srebro et al. 2003, Zhang et al. 2005, Zhang et al. 2006]. Memory-based algorithms compute a

prediction by combining ratings of selected users or items that are judged to be relevant. Model-based algorithms use all available ratings to learn a model, which can then be used to predict the rating of any given item by any given user.

The objective of our recommendation component is to help users to find the most reliable, valuable, important, and interesting information quickly and easily. In its most common formulation, the recommendation problem is reduced to the problem of predicting ratings for the items that a user has not seen before. Once we can estimate a user's ratings for all unrated items, we can recommend the items predicted to receive the highest ratings. Figure 1 gives a simple recommendation scenario composed of 8 users and 6 items. User preferences on items are expressed using discrete numerical values from 1 to 5, where 5 represents the fact that a user likes the corresponding item very much. In order to make recommendations to user Alice, we compute predicted ratings for Alice on those items that she has not rated yet and then recommend items that have the highest predicted ratings.

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
Alice	5		3			**?**
User1			4		4	
User2		3			2	1
User3	4			1		
User4	3			3		2
User5		3				1
User6	5	3				
User7				4		2

Figure 1. *A simple recommender system scenario. In order to recommend items to user Alice, predictions to those missing entries corresponding to the row of Alice are computed first, and then items are recommended to Alice based on their predictions.*

4. Recommendation Component

In this section, we introduce the four collaborative filtering algorithms that are used for providing users with recommendations in OC.

4.1 User-based Algorithm

The user-based CF algorithm [Herlocker et al. 1999, Resnick et al. 1994] first computes the correlations between users using a mean-adjusted Pearson correlation, and then combines a weighted average of the *k* nearest neighbors' ratings to produce a prediction. More precisely, a predicted rating $P_{i,j}$ for user *i* on item *j* is computed by

$$P_{i,j} = \overline{A_i} + \frac{\sum_{u=1}^{k} w_{i,u} \cdot (A_{u,j} - \overline{A_u})}{\sum_{u=1}^{k} |w_{i,u}|},$$

where $A_{u,j}$ is user *u*'s rating on item *j*, $\overline{A_i}$ is user *i*'s average rating, and $w_{i,u}$ is the correlation between users *i* and *u*.

Several different similarity weighting have been used to compute $w_{i,u}$. The most common weighting measure used is the *Pearson correlation coefficient*. Pearson correlation measures the degree to which a linear relationship exists between two variables. It is derived from a linear regression model that relies on a set of assumptions regarding the data, namely that the relationship must be linear, and the errors must be independent and have a probability distribution with mean 0 and constant variance for every setting of the independent variable [Herlocker et al. 1999, McClave et al. 1988]. If the Pearson correlation coefficient is used, we have

$$w_{i,u} = \frac{\sum_{l=1}^{h} (A_{i,j_l} - \overline{A_i})(A_{u,j_l} - \overline{A_u})}{h\sigma_i\sigma_u},$$

in which *h* is the number of common items that both user *i* and *u* have rated, and j_l is the index of the *l*th common item. Both rating averages ($\overline{A_i}$, $\overline{A_u}$) and rating standard deviations (σ_i, σ_u) are computed based on those common items only.

4.2 Item-based Algorithm

The item-based algorithm [Sarwar et al. 2001] is analogous to the previous one, but computes and uses similarities between items rather than users. The formula used to compute a prediction is

$$P_{i,j} = \frac{\sum_{v=1}^{k} s_{v,j} \cdot A_{i,v}}{\sum_{v=1}^{k} |s_{v,j}|}.$$

Here, $s_{v,j}$ is the similarity between items *v* and *j*. Our implementation uses an *adjusted cosine correlation* to compute similarities. That is,

$$s_{v,j} = \frac{\sum_{u \in U} (A_{u,v} - \overline{A_u})(A_{u,j} - \overline{A_u})}{\sqrt{\sum_{u \in U} (A_{u,j} - \overline{A_u})^2} \sqrt{\sum_{u \in U} (A_{u,v} - \overline{A_u})^2}},$$

where *U* denotes the set of all users.

4.3 Singular Value Decomposition based Algorithm

Singular Value Decomposition (SVD) was first introduced into recommendation systems in [Billsus et al. 1998] and [Sarwar et al. 2000]. The underlying assumption of applying SVD to a rating matrix is that observed ratings $A_{i,j}$ are combinations of ratings from a low-dimensional linear model (denoted as *X*) and Gaussian noise (with zero mean). That is,

$$A_{i,j} = X_{i,j} + Z_{i,j}, \text{ with } Z_{i,j} i.i.d. \sim N(0, \sigma^2). \quad \textbf{(1)}$$

Since the rating matrix in the real world is incomplete and sparse, Srebro and Jaakkola [Srebro et al. 2003] proposed an Expectation-Maximization (EM) algorithm to maximize the log-likelihood of all observed ratings A^o, that is $\log \Pr(A^o \mid X)$. In this paper, we use this SVD-based algorithm in which the EM algorithm is incorporated. The details of the derivation of this EM algorithm can be found in [Srebro et al. 2003, Zhang et al. 2005]; an overview is given below.

In the Expectation step of the *t*th iteration, a filled-in matrix $A^{(t)}$ is formed where unobserved entries $A_{i,j}^{(t)}$ are equal to the corresponding values of the computed linear model in the previous iteration ($X_{i,j}^{(t-1)}$) and observed entries are unchanged from *A*. That is,

$$A_{i,j}^{(t)} = \begin{cases} A_{i,j} & \text{if } A_{i,j} \text{ is rated} \\ A_{i,j}^{(t-1)} & \text{otherwise} \end{cases}.$$

In the following Maximization step, we perform SVD on this filled-in matrix $A^{(t)}$ to get $A^{(t)} = USV^T$. The updated linear model $X^{(t)}$ is computed as $X^{(t)} = U_k S_k V_k^T$, where U_k, S_k, and V_k are matrices composed of the top *k* left singular vectors, singular values, and right singular vectors, respectively.

The above EM procedure is ensured to converge, which means that the log-likelihood of all observed ratings given the current model estimate is always nondecreasing. After the EM procedure finishes, a prediction is computed as the corresponding entry in the final computed model, *i.e.*, $P_{i,j} = X_{i,j}$.

4.4 Non-negative Matrix Factorization based Algorithm

We proposed a Non-negative Matrix Factorization (NMF) based algorithm in [Zhang et al. 2006] to compute a low-dimensional linear model with a non-negativity

constraint. The enforced non-negativity constraint ensures that each user's rating profile is an additive linear combination of *k* canonical coordinates, and each coordinate is in the range of the normal rating space. Therefore, each coordinate can be regarded as a representative rating profile from a user community or interest group, and each user's ratings can be modeled as an additive mixture of rating profiles from user communities or interest groups. A user community can be thought of as an expression of a particular statistical pattern in the opinions of users, and typically has some kind of real world interpretation.

For ease of our expression, we now transpose a rating matrix to an *n* items-by-*m* users matrix. The linear model *X* in Eq. (1) is enforced as a product of two non-negative matrices *U* (*n*-by-*k*) and *V* (*k*-by-*m*), *i.e.*, $X = UV$. If original ratings can take negative values, we can simply shift all ratings into a non-negative range by subtracting the minimum of the original range (and finally shift the obtained linear model back to the original range).

In order to compute the model *X* (a pair of *U* and *V*) from all observed ratings A^o, one approach is to use an EM procedure (similar to the one in Sec. 4.3). The Expectation step in each iteration remains the same; while in the Maximization step, the updated model is computed using the following updating formulas:

$$U_{i,j}^{(t+1)} = U_{i,j}^{(t)} \frac{(AV^T)_{i,j}}{(UVV^T)_{i,j}}, \quad V_{i,j}^{(t+1)} = V_{i,j}^{(t)} \frac{(U^T A)_{i,j}}{(U^T UV)_{i,j}}.$$

The second approach to compute the model is to directly apply the Lagrange multiplier on objective function $\log \Pr(A^o \mid X)$, which results in the following updating formulas for Weighted Non-negative Matrix Factorization (WNMF):

$$U_{i,j}^{(t+1)} = U_{i,j}^{(t)} \frac{((W * A)V^T)_{i,j}}{((W * (UV))V^T)_{i,j}}, \quad V_{i,j}^{(t+1)} = V_{i,j}^{(t)} \frac{(U^T (W * A))_{i,j}}{(U^T (W * (UV)))_{i,j}}.$$

In the above, *W* is a weight matrix in which $W_{i,j}$ is equal to one if $A_{i,j}$ is the observed rating and zero otherwise; * denotes element-wise multiplication.

As we pointed in [Zhang et al. 2006], the EM procedure converges well empirically and is less susceptible to the initial starting conditions than WNMF, but the latter is much more computationally efficient. Taking into account the advantages of both algorithms, a hybrid approach was also introduced in [Zhang et al. 2006]; here, the EM procedure is performed first on the rating matrix for several iterations to obtain a preliminary linear model (a pair of *U* and *V*), and then this pair of *U* and *V* are taken as the initial values of the WNMF approach. Compared with a randomized model, the preliminary model obtained after several iterations of the EM procedure is more

likely to be accurate, so that the WNMF approach is more likely to obtain a good local (or global) optimum. In short, such an approach not only has a high likelihood of obtaining accurate predictions, but reduces the computational cost as well.

5. An Evaluation Framework

For the evaluation of the recommendation component, an evaluation framework has been synthesized. It combines approaches and elements from the research that has been conducted on the evaluation of recommender systems and also draws from the research on technology acceptance models.

The following are the most widely used evaluation metrics in recommender systems.

- Predictive accuracy metrics (*e.g.*, Mean Absolute Error (MAE)) measure how close a recommender system's predicted ratings are to true user ratings.
- Classification accuracy metrics (*e.g.*, Precision and Recall) measure the frequency with which a recommender system makes correct or incorrect decisions about whether an item is relevant (interesting) to a user.
- Receiver Operating Curve (ROC) plots the percentage of relevant items selected and shown versus the percentage on non-relevant items selected and shown. The area underneath this curve is known as the ROC area.
- The correlation between a recommender system's predicted ratings and corresponding true user ratings.

The approaches and methods developed in the extensive research that has been conducted on the acceptance of new technologies can also be useful for the formulation of a recommender system evaluation framework. The most widely utilized model in this area is the Technology Acceptance Model (TAM) [Davis 1986]. According to this model the individual acceptance of a technology is influenced mainly by two basic determinant factors: the Perceived Usefulness (the degree to which a person believes that using a particular system would enhance his or her job performance) and the Perceived Ease of Use (the degree to which a person believes that using a particular system will be free of effort).

By combining approaches and elements from these two areas, we have devised a framework for evaluating the above four collaborative filtering algorithms of the recommendation component. It includes two layers of factors that are to be assessed (*e.g.*, through a questionnaire on a Lickert scale from 1 to 7) by a number of individuals having used these four algorithms for some time (initially free use for doing any search each user wants, followed by execution of a number of predefined scenarios). The first layer includes nine factors measuring subjective user perceptions concerning: (a) how difficult it was to learn using the system; (b) how difficult it was to enter the ratings required by the system; (c) how structured, clear and understandable was the interface of the system; (d) to what extent the items

recommended by the system were relevant and useful; (e) to what extent the system proposed useless non-relevant items; (f) to what extent the system is believed to have recommended all the relevant and useful items available; (g) the perceived novelty (new, not known before, and non-obvious) of the recommended items; (h) how much the user trusts the system (*i.e.*, believes that the recommendations provided by the system are correct); and (i) to what extent the system reduces the time and the effort required for finding relevant and useful items.

The second layer includes three factors measuring the overall experience of the user: (a) overall satisfaction of the user, (b) strength of intention of the user to use it again, and (c) strength of intention of the user to recommend it to colleagues.

Additionally for each algorithm the following objective measures are calculated: the Mean Absolute Error, the Precision, the Recall, and the correlation between recommender system's predicted ratings and corresponding true user ratings.

Data collected using the above evaluation framework are processed in the following ways.

- For each of the above factors of the first layer and the second layer, the basic descriptive statistics (*e.g.*, average, standard deviation) is calculated.
- Correlations between corresponding subjective and objective measures are calculated (*e.g.*, the correlation between factor (d) and Precision, between factor (f) and Recall).
- Relations between each of the factors of the second level and each of the factors of the first layer are examined. For this purpose initially the corresponding correlations is calculated, and then a structural equation model is estimated (see [Tinsley et al. 2000]; all the paths between each of the factors of the second level and each of the factors of the first layer are estimated). In this way we can examine which of the factors of the first layer has a larger effect on the overall experiences of the users, and in general we will gain a better understanding of the user value generation mechanism.

The application of the evaluation framework will be carried out in a future study.

6. Summary

In this paper, we describe our work in building a recommendation component in an Open Collaboration system to help users find valuable information they are potentially interested in. Our recommendation component has four implemented recommendation algorithms. The user-based algorithm computes predictions by exploiting information from those users similar to the current user, and the item-based algorithm operates analogously for items. The SVD-based algorithm and the NMF-based algorithm compute a low-dimensional linear model from all observed ratings

and uses this model for predictions. We present an evaluation framework for making an assessment of the usefulness of these (or other) algorithms in a particular application or setting, taking into account both objective and subjective evaluative factors.

Acknowledgements

This material is based in part upon work supported by the National Science Foundation under award number IDM 0308229.

References

Billsus D., Pazzani M. J. (1998), *Learning collaborative information filter*, in Proc. of the 15th Int. Conf. on Machine Learning, pp. 46-54.

Canny J. (2002), *Collaborative filtering with privacy via factor analysis*, in Proc. of the 25th ACM SIGIR, pp. 238-245.

Davis F. D. (1986), *A technology acceptance model for empirically testing new end-user information systems: theory and results*, PhD thesis, Sloan School of Management, Massachusetts Institute of Technology.

Herlocker J. L, Konstan J. A., Borchers A., Riedl J. (1999), *An algorithmic framework for performing collaborative filtering*, in Proc. of the 22nd ACM SIGIR, pp. 230-237, 1999.

Hofmann T. (2004), *Latent semantic models for collaborative filtering*, ACM Tran. on Information Systems, vol. 22(1), pp. 89-115.

McClave J. T., Dietrich F. H. II. (1988), *Statistics*, Dellen Publishing Company.

Resnick P., Iacovou N., Suchak M. Bergstrom P., Riedl J. (1994), *GroupLens: An open architecture for collaborative filtering of netnews*, in Proc. of the ACM Conf. on Computer Supported Cooperative Work, pp. 175-186.

Sarwar B. M., Karypis G., Konstan J. A., Riedl J. (2001), *Item-based collaborative filtering recommendation algorithms*, in Proc. of the 10th WWW, pp. 285-295.

Sarwar B. M., Karypis G., Konstan J. A., Riedl J. (2000), *Application of dimensionality reduction in recommender systems-a case study*, in Proc. of the ACM WebKDD Workshop.

Srebro N., Jaakkola T. (2003), *Weighted low-rank approximation*, in Proc. of the 20th Int. Conf. on Machine Learning, pp. 720-727.

Tinsley H., Brown S., editors. (2000), *Handbook of applied multivariate statistics and mathematical modeling*, Academic Press.

Zhang S., Wang W., Ford J., Makedon F. (2006), *Learning from incomplete ratings using non-negative matrix factorization*, in Proc. of the SIAM-Data Ming Conf.

Zhang S., Wang W., Ford J., Makedon F., Pearlman J. (2005), *Using singular value decomposition approximation for collaborative filtering*, in Proc. of the 7th IEEE Conf. on E-commerce, pp. 257-264.