Supporting Social Recommendations with Activity-Balanced Clustering

F. Maxwell Harper, Shilad Sen, Dan Frankowski
GroupLens Research
University of Minnesota
Minneapolis, MN 55455, USA
{harper, ssen, dfrankow}@cs.umn.edu

ABSTRACT
In support of social interaction and information sharing, online communities commonly provide interfaces for users to form or interact with groups. For example, a user of the social music recommendation site last.fm might join the "First Wave Punk" group to discuss his or her favorite band (The Clash) and listen to playlists generated by fellow fans. Clustering techniques provide the potential to automatically discover groups of users who appear to share interests. We explore this idea by describing algorithms for clustering users of an online community and automatically describing the resulting user groups. We designed these techniques for use in an online recommendation system with no pre-existing group functionality, which led us to develop an "activity-balanced clustering" algorithm that considers both user activity and user interests in forming clusters.

Categories and Subject Descriptors
H.5.3 [**Information Interfaces and Presentation**]: Group and Organizational Interfaces – *Web-based Interaction*; I.5.3 [**Pattern Recognition**]: Clustering – *Algorithms*

General Terms
Algorithms, Design, Human Factors

Keywords
Activity-balanced clustering, user group summarization.

1. INTRODUCTION
Online communities offer new opportunities for social interaction and information sharing. Interfaces that support activities such as posting, tagging, and rating allow community members to share information as well as to create and sustain relationships. Online communities take many forms, from discussion forums to social bookmarking sites to recommender systems.

Preece [8] advocates that community designers should think both in terms of interface *usability* and in terms of community *sociability*. Usable interfaces allow members to discover and use system features effectively, while sociable systems encourage user interactions and the formation of a community purpose. In this paper, we consider algorithms in support of essentially sociable features – user groups and social recommendations – and examine techniques for making these features more understandable for users.

User group features are typically intended to facilitate information exchange and social interaction. For example, Flickr, an online photo sharing community, allows members to organize and join public and private groups in order to share and discuss photos among friends or others with shared interests. Large groups may form around common interests (e.g. the "Hardcore Street Photography" group has 12,000 members), and coexist with small groups that cater to more personal relationships (e.g. the "Friends of Hatfield Forest" group has 4 members).

Group interfaces effectively support the notion of social recommendations, where members trade recommendations through social features rather than through an algorithmic process such as collaborative filtering. Last.fm, an online community about music, allows members to publish information about the songs they listen to; the group features allow members to find others with similar interests. Through browsing the 45 member "Yanni" group, one might find new friends or discover new artists to listen to by browsing other Yanni lovers' playlists. StumbleUpon, a social recommender for web pages, also supports groups for interest areas. Members might join the "Cooking Websites" group to find recommendations for great cooking resources from other Stumblers who like to cook.

In this research, we explore a new paradigm of user group features by looking at algorithms for automatically forming and describing user groups. By automatically assigning users to a group, all users have access to group-based social recommendations from their first login, whether or not they know others in the system and irrespective of their commitment to the system. Also, automatic grouping has the capability of forming groups based on shared traits or preferences, which might improve the accuracy or believability of social recommendations.

To be specific, we introduced "movie groups" into MovieLens (http://movielens.org), an online movie recommender system. MovieLens was originally built to allow members to rate movies and receive algorithmically generated movie recommendations [9]; it has over time evolved into an online community that supports discussion [4] and tagging [10]. In this paper, we

RecSys'07, October 19–20, 2007, Minneapolis, Minnesota, USA.

describe the methods of "activity-balanced clustering" and automatic "user group summarization" we developed to algorithmically form and describe the movie groups

2. BACKGROUND

Clustering algorithms partition sets of data into groups based on measurements of similarity between constituent elements [6]. In subsequent sections, we describe methods for clustering MovieLens users based on their movie rating profiles. To compute the similarity between two users, we calculate the adjusted cosine similarity between vectors of movie ratings; this is standard practice in recommender systems research [9].

Based on our design of the movie groups feature, we wish to generate a fixed number of clusters, with each cluster containing approximately the same number of users. Distributing data equally among clusters is known as "balanced clustering" [1]. Most "standard" clustering algorithms such as k-means may produce clusters of widely differing sizes.

Our design is premised on the belief that movie groups are more interesting and useful if users can understand the characteristics that define and differentiate the groups. There are several approaches to the problem of describing clusters; most relevant to our research is the practice of choosing a cluster's "representative" items. For example, researchers have summarized clusters of text documents by choosing representative words. Intuitive approaches for summarizing text may lead to choosing bland or obscure summary words [7], problems that we revisit in this paper in a new context.

Both algorithms that we present in this research draw on prior work on stable matching algorithms. Stable matching is a process for matching elements of two sets such that there does not exist any pair of elements that would both prefer to be matched together over their current match. We use a variant of the Gale-Shapley matching algorithm [3], which begins with each element defining a ranked list of preferences for elements in the other set, and continues with a series of iterations where elements from one set "propose" to elements in the other set.

3. ACTIVITY-BALANCED CLUSTERING

Our task is to create user groups; our approach is algorithmic clustering. Prerequisite to clustering is picking some dimension along which to define user similarity, data that might be found by looking at profile data (explicitly given) or usage log data (implicitly observed). In MovieLens, we have chosen to group users based on their movie rating profiles, since these data offer us the greatest depth and coverage of any data available.

Recall that we are designing for sociability. In support of this goal, we wish to create groups where each group contains about the same number of active users, because groups without activity cannot interact and share social recommendations. We have chosen to define an "active user" as one who has logged in during each of the past three months. At the time of our study, there were 1,394 such active users – 25% of the total number of users to visit MovieLens during those three months. These active users substantially affect visible contributions to MovieLens, accounting for 84% of forum posts during this time period. However, they only accounted for 22% of movie ratings. Thus, in order to balance the level of shared ratings information available for display, we added an additional requirement that all groups contain approximately the same total number of members.

3.1 Clustering Procedure

Given our requirement for cluster balance, we do not believe standard data clustering techniques are sufficient. Thus, we developed a new two stage approach to clustering users (see Algorithm 1). Stage 1 uses balanced agglomerative clustering to build high-quality balanced clusters of hundreds of active users while Stage 2 uses stable matching, a computationally efficient technique, for assigning thousands of less active users in a balanced fashion. This two stage approach is related to and inspired by prior work in balanced clustering algorithms [1]. Let U be the set of all users, A be the set of recently active users, and U-A be the set of all other users. We desire k clusters as output.

Stage 1: A balanced agglomerative clustering algorithm

```
1. Consider each user in A to be a cluster of
size one.  These clusters are not "done".
2. Until there are k "done" clusters:
  2.1. Merge the two most similar clusters
  that are not "done". Call this cluster c.
  2.2. If |c| >= |A|/k, remove the user in c
  least similar to c's centroid until |c| ==
  k. Declare cluster c "done".
```

Stage 2: A stable matching algorithm

```
1. Consider each user in U-A to be
"unmatched".
2. For each cluster output by Stage 1, rank
all users in U-A according to adjusted cosine
similarity between the user and the cluster
centroids
3. While there exists an "unmatched" user,
for each cluster c:
  3.1. Until cluster c finds a user to add, c
  proposes to the highest ranked user u that
  it has not yet proposed to.
    3.1.1. u accepts the proposal if it is
    "unmatched" or if it prefers c to the
    cluster it is currently matched with.
    3.1.2. If u accepts, consider u to be
    "matched", remove u from any previously
    matched clusters, and add u to c.
```

Algorithm 1. Activity-Balanced Clustering

3.2 Evaluation

In this section, we provide a preliminary evaluation of the performance of activity-balanced clustering, using (1) standard clustering metrics, and (2) a comparison with standard k-means clustering on several key indicators. We collected data by running both the activity-balanced clustering procedure described above and standard k-means on the dataset of MovieLens users who had logged in during the past year (n=18,760).

Two useful metrics for evaluating the output of a clustering algorithm are *compactness* and *separation* [5]. We strive for low compactness scores and high separation scores, since a smaller compactness value represents a cluster with less variance, and a higher separation value represents clusters that are more distinct. Table 1 shows the result of our analysis; activity-balanced clustering outperforms k-means for both metrics in this analysis.

Table 1. Cluster compactness and separation for activity-balanced and k-means clustering on MovieLens users.

	Compactness	Separation
Activity-balanced	0.93	0.75
k-means	0.99	0.41

Cluster quality is not the whole story. We are also interested in the balancing performance of activity-balanced clustering as compared with an unbalanced technique such as k-means. Table 2 compares the activity-balanced clustering algorithm with k-means in terms of the standard deviation of the number of users per cluster and the number of active users per cluster. Activity-balanced clustering produces far more uniform clusters than k-means, both in terms of the total number of users per cluster, and in terms of the number of active users. K-means resulted with one very large cluster (with 74% of all the users, and 84% of the active users) and nine very small clusters, reflected in the larger standard deviation. Clearly, k-means would not be appropriate given our requirements – contrary to our goal of sociability, k-means might lead to one very successful group, and nine very quiet groups. In comparison, the clusters generated by activity-balanced clustering ranged between 9-11% of users each, and between 6-13% of active users each.

Table 2. Standard deviation of the number of users or active users per cluster across three clustering runs.

	Std Dev # Users	Std Dev # Active Users
Activity-balanced	2.87	22.00
k-means	4243.01	334.19

4. USER GROUP SUMMARIZATION

As a result of running activity-balanced clustering, we had ten "movie groups" each consisting of thousands of users. When users log in and discover their new group, they are bound to ask why they were assigned to one group and not another. Preece's call for considering usability tells us that we should present the groups so that members can determine why they have been grouped, and what other users in their group might be like. We also wish for our groups to appear to be distinct from one another – having ten similar-looking groups might be confusing to users.

Previous work has described clusters by picking "representative" entities for display (e.g., [7]). Were we to take this approach in MovieLens, we might pick a few users to display, or we might construct "meta-users" that represent cluster centroids. Since all users express themselves through rating movies, we summarize clusters by displaying their representative movies.

However, picking representative movies presents several interesting challenges related to what has been called the "banana problem" [2]. Because bananas are such a commonly purchased item in (United States) grocery stores, a recommendation system that doesn't know any better might always recommend bananas, irrespective of context or state. The authors of this paper might also label this the "Shawshank Redemption problem", where all clusters of users show a strong (average) preference for a 1994 movie called *The Shawshank Redemption*. Of the ten movie groups currently live in MovieLens, eight have *The Shawshank Redemption* as one of their two highest rated movies. Because we wish for our users to be able to distinguish between clusters, we should not describe a cluster by simply listing its highest rated movies.

Another approach for choosing representative movies for a cluster is to pick movies that a cluster likes more than other clusters. However, this approach also does not lead to unique lists of representative movies for each cluster, and tends to pick "average" movies as representative, which does not have face-validity to users. For example, many users will vocally complain if they are placed in a group that is represented by the movie *The Return of the Texas Chainsaw Massacre*. For good reason.

4.1 Summarization Procedure

Our cluster summarization algorithm, based on stable matching, is capable of picking well-liked, distinct movies for each cluster. Let C be the set of k clusters, and M be the set of j movies.

```
1. Consider each movie in M to be "unmatched".
2. For each cluster in C, rank all movies in M
according to the cluster's average rating for
each movie.  Linearly devalue average movie
ratings with fewer than 50 votes.
3. While there exists an "unmatched" movie, for
each cluster c in C:
  3.1. Until cluster c finds a movie to add, c
  proposes to the highest ranked movie m that
  it has not yet proposed to.
    3.1.1. m accepts the proposal if it is
    "unmatched" or if it prefers c to the
    cluster it is currently matched with.
    3.1.2. If m accepts, consider m to be
    "matched", remove it from any previously
    matched clusters, and add to c.
  3.2. Remember the order in which movies are
  picked.  The nth movie picked is the nth most
  representative movie for c.
```

Algorithm 2. User Group Summarization

Intuitively, we might trust 100 ratings with an average of 4.5 stars more than a single 5 star rating. To model this, we linearly devalue similarity scores for movies that fewer than 50 co-ratings (see Algorithm 2, line 2). Other systems might choose different smoothing techniques for devaluing items with little data.

4.2 Evaluation

To evaluate the types of movies our algorithm returns, we compare it against several other plausible algorithms:

- POP - ("popularly rated") Each cluster is summarized by the movies its users have rated most often
- HR20 - ("high ratings 20") Each cluster is summarized by the movies its users have given the highest average rating. Only consider movies that have been rated by 20 or more of the cluster's users.
- DM20 - ("differing means 20") Each cluster is summarized by the movies its users have rated the highest compared with users in other clusters. Only consider movies that have been rated by 20 or more of the cluster's users.

In Table 3 we compare these three algorithms with our own algorithm (MATCH) on a number of dimensions. First, we consider "popRank" (popularity rank), a proxy for how commonly recognizable a movie is. The most often rated movie in MovieLens, Pulp Fiction, has popRank=1 out of the 9,500 movies in the database. Second, we consider "avgRating" (average rating), which can range between 0.5 stars and 5 stars. Finally, we consider "uniqueness", which we define to be the number of unique movies that are shown in the ten clusters as the "five most

representative movies". The highest possible uniqueness score is 50 (no clusters have overlap), and the lowest possible score is 5 (all clusters are summarized by the same 5 movies). In Table 4 we show the top two representative movies from one user group (for illustrative purposes only). MATCH returns movies with fewer ratings and a lower average rating than the other algorithms, but does the best job of providing unique descriptions for each cluster.

Table 3. A comparison of our user group summarization technique (MATCH) vs. other techniques in the chosen movies' avg. popularity, avg. user rating, and uniqueness.

	MATCH	POP	HR20	DM20
popRank	1754.0	53.5	801.6	3821.1
avgRating	3.9	4.1	4.2	3.2
uniqueness	50	11	34	48

Table 4. Top-two representative movies for one movie group.

MATCH	Das Boot (1981), Blade Runner (1982)
POP	Pulp Fiction (1994), The Matrix (1999)
HR20	The Godfather (1972), The Shawshank Redemption (1994)
DM20	Return of the Texas Chainsaw Massacre (1994), Cursed (2005)

5. DISCUSSION

In this paper, we have examined the problems of automatically grouping users and describing the resulting groups. Two methods result, one for activity-balanced clustering, and the other for user group summarization by way of choosing representative movies.

Using these groupings, we have modified MovieLens to give each movie group a home page, where information about activity, recent ratings, recent posts, etc. is shown. Members may browse these pages to explore the MovieLens zeitgeist. This research is part of a larger study where we are examining different forms of attachment in online communities. We save deep analysis of user behavior for future work. Based on user reactions posted in the discussion forums (we have received over 90 forum posts about the groups), there are several lessons to be learned:

- Some users were astounded by how well they were grouped, others by how badly – they appear to base their judgment largely on how well they like the five representative movies we show on the group home page.
- Users wish to know how they were assigned to their group. Although we told users that they were grouped according to movie preferences, many wished for more detail.
- Many users agree that they should not be grouped until they have reached a certain level of contribution (say, 100 movie ratings). Before then, users might be placed in a newbie-only group. This would encourage more accurate grouping, which users appear to value.
- Users were curious to learn about others in their group, and wished that we had built even more support for social recommendations than we did.

Most importantly to nurturing the sociability of MovieLens, all groups have remained active since the launch of the groups (the standard deviation of the number of 3-month and 1-year active members across clusters is currently 25.6 and 76.6, respectively). Activity-balanced clustering has thus-far led to a sustainable set of user groups.

Time will tell if automatic user grouping is a useful feature in encouraging sociability in an online community. In future work, we will examine this question deeply, and evaluate this work from the perspective of a community designer. We also hope to follow this work with a more rigorous investigation of the principles of clustering and describing users in online communities.

6. ACKNOWLEDGMENTS

Our thanks to Sara Drenner for her work implementing the social recommendations software in MovieLens, to Loren Terveen and John Riedl for early discussions about clustering, and to Yuqing Ren, Robert Kraut, Sara Kiesler, and Joe Konstan for their contributions to the broader scope of this research. This work is funded by the National Science Foundation, grant IIS 03-24851.

7. REFERENCES

[1] Banerjee, A., Ghosh, J. Scalable Clustering with Balancing Constraints. *Data Mining and Knowledge Discovery*, 13(3), 2006.

[2] Burke, R. Integrating Knowledge-Based and Collaborative Filtering Recommender Systems. *Workshop on Artificial Intelligence for Electronic Commerce*, 1999.

[3] Gale, D., Shapley, L. College Admissions and the Stability of Marriage. *American Mathematical Monthly*, 69(1), 1962.

[4] Harper, F., Frankowski, D., Drenner, S., Ren, Y., Kiesler, S., Terveen, L., Kraut, R., Riedl, J. Talk Amongst Yourselves: Inviting Users To Participate In Online Conversations. *IUI*, 2007.

[5] He, J., Tan, A., Tan, C., Sung, S. On Quantitative Evaluation of Clustering Systems. In *Information Retrieval and Clustering*, Kluwer Academic Publishers, 2002.

[6] Jain, A., Murty, M., Flynn, P. Data Clustering: A Review. *ACM Computing Surveys*, 31(3), 1999.

[7] Popescul, A., Ungar, L. Automatic Labeling of Document Clusters, *Unpublished Manuscript*, Available at http://citeseer.nj.nec.com/popescul00automatic.html, 2000.

[8] Preece, J. *Online Communities: Designing Usability, Supporting Sociability*. John Wiley & Sons, 2000.

[9] Sarwar, B., Karypis, G., Konstan, J., Riedl, J. Item-Based Collaborative Filtering Recommendation Algorithms. *WWW*, 2001.

[10] Sen, S., Lam, S., Cosley, D., Rashid, A., Frankowski, D., Osterhouse, J., Harper, F., Riedl, J. tagging, community, vocabulary, evolution. *CSCW*, 2006.