Push-Poll Recommender System: Supporting Word of Mouth

Andrew Webster and Julita Vassileva

Department of Computer Science
University of Saskatchewan, Saskatoon SK, S7N 5C9, Canada
asw292@mail.usask.ca, jiv@cs.usask.ca

Abstract. Recommender systems produce social networks as a side effect of predicting what users will like. However, the potential for these social networks to aid in recommending items is largely ignored. We propose a recommender system that works directly with these networks to distribute and recommend items: the informal exchange of information (word of mouth communication) is supported rather than replaced. The paper describes the push-poll approach and evaluates its performance at predicting user ratings for movies against a collaborative filtering algorithm. Overall, the push-poll approach performs significantly better while being computationally efficient and suitable for dynamic domains (e.g. recommending items from RSS feeds).

1 Introduction

The main advantages of recommender systems are that recommendations are supplied *on demand* and are made from *a massive item collection*. For example, Amazon.com can be queried in the middle of the night for book/music recommendations from a list of millions [1]. However, recommending an item and giving a clear explanation of why it was recommended is not forthcoming in these distinctively "black box" systems [2]. We see the lack of believable explanations as a symptom of a wider limitation: the control and execution of a distributed process by a central authority. Recent recommender systems tend to be placed between people as authoritative intermediaries. It appears to the user that she is engaged in a dialog with the system–not her peers–about what to view next, although in fact the system may be associating her with other like-minded users in order to predict items of interest. These user-to-user associations, or *connections*, are left embedded within the system, and their full potential for improving recommendations is largely ignored.

Early recommender systems were said to simulate the informal, verbal exchange of information known as *word of mouth* communication [3]. Revisiting this conceptualization, we propose a recommender system that directly exploits user-to-user connections as the primary method to distribute and recommend information items: word of mouth processes are supported rather than replaced. Our approach models the implicit social networks that normally develop around

C. Conati, K. McCoy, and G. Paliouras (Eds.): UM 2007, LNAI 4511, pp. 288–297, 2007.

shared topics of interests, or *channels*. Recommendation is achieved by two interrelated processes: push and poll. *Push* seeds an item into the social network associated with the channel the item is most appropriate for. The item then spreads according to *diffusion of innovation* models [4]. *Poll* queries adjacent users whether the item should be *activated* (i.e. recommended) for the current user, given a certain *activation threshold*. Feedback from users reshapes the network, affecting the spread and activation of subsequent items.

The next section reviews related work on social networks and how information flows in them. The proposed approach, push-poll, is described in Section 4. In Section 5, its performance is evaluated against a common collaborative filtering algorithm (which is reviewed in Section 3). We conclude in Section 6 that there are immediate advantages from directly working with social networks.

2 Related Work

Users' activity within a recommender system has been acknowledged as "inducing an implicit social network and [influences] the connectivities in this network" [5]. *Social networks* [6] are conceptualized as graphs that represent people and the relationships between them as nodes and edges, respectively. Edges, or *connections*, traditionally denote the existence of social relationships (e.g. friendship) but can also indicate more general relationships, e.g. users who have shared interests. Thus, recommender systems produce social networks, i.e. user-to-user connections, *as a consequence* of predicting user-to-item connections.

It has been emphasized that the recommendation process naturally involves bringing people together and how these connections are determined is a significant, but neglected, aspect of recommender systems research [7]. User modeling, either direct (e.g. using explicit input like item ratings) or indirect (e.g. data mining e-mail logs), and the computed similarity between user models was seen as the primary means to obtain social networks that are exploitable by the recommendation process either through structural analysis or by embedding additional information into connections between users. For an example of the former approach, recommender systems in general were evaluated in light of the network structure created between users under certain conditions [5]. One condition that was analyzed was the minimum number of shared items users must rate in order to all be connected together (identifying this number would help strike a balance between ensuring good recommendations and not alienating users with too much work). For an example of the latter approach, explicit indication of trust between users was collected, embedded into the computed social network, and used to generate improved movie recommendations [8].

The study of information propagation through social networks is another related area of research. The spread and adoption of *social innovations* within real-world communities [9] is of particular relevance as ensuing models can be applied to online environments. For example, a model for the spread of discussion topics in *weblogs*, or blogs, is presented in [10] and the identification of a

minimal set of people whose adoption of a new product would maximize the spread of that product through the given social network is described in [11].

3 Collaborative Filtering

We begin with a brief overview of the collaborative filtering (CF) algorithm used to evaluate the performance of push-poll in Section 5. CF operates on the *user-item matrix*, R, where entry $r_{c,s}$ indicates the rating score user $c \in \{c_1, c_2, \ldots, c_m\}$ has given item $s \in \{s_1, s_2, \ldots, s_n\}$. Each row represents all ratings a particular user has made, and each column represents all ratings a particular item has collected. Often, rating scores follow a numerical scale (e.g. 1 to 5 stars) and are explicit, but they also may be inferred from item purchases and other implicit user actions [12]. The ultimate goal is to predict the score of empty cells for the *active user*, the user currently requesting recommendations.

CF algorithms are divided into two categories: *memory-based* and *model-based*. We focus on a memory-based algorithm because of its wide use and satisfactory prediction accuracy. For a review of CF, we refer to [13].

3.1 Memory-Based Algorithm

Memory-based CF algorithms rely on exploiting gaps within the user-item matrix. The intuition is that users who have similar preferences will generally rate items in a similar manner. Therefore, if the active user c has not rated item s, but the recommender system can find similar or correlated users (i.e. *neighbours*) who have, then a rating score can be predicted using (1).

$$r_{c,s} = \bar{r}_c + k \sum_{\acute{c} \in \hat{C}} \mathrm{sim}(\mathrm{c}, \acute{\mathrm{c}}) \times (\mathrm{r}_{\acute{c},\mathrm{s}} - \bar{\mathrm{r}}_{\acute{c}}) \tag{1}$$

$$\mathrm{sim}(\mathrm{c}, \acute{\mathrm{c}}) = \frac{\sum_{\mathrm{s} \in \mathrm{S}_{c\acute{c}}} (\mathrm{r}_{\mathrm{c},\mathrm{s}} - \bar{\mathrm{r}}_{\mathrm{c}})(\mathrm{r}_{\acute{c},\mathrm{s}} - \bar{\mathrm{r}}_{\acute{c}})}{\sqrt{\sum_{\mathrm{s} \in \mathrm{S}_{c\acute{c}}} (\mathrm{r}_{\mathrm{c},\mathrm{s}} - \bar{\mathrm{r}}_{\mathrm{c}})^2 \sum_{\mathrm{s} \in \mathrm{S}_{c\acute{c}}} (\mathrm{r}_{\acute{c},\mathrm{s}} - \bar{\mathrm{r}}_{\acute{c}})^2}} \tag{2}$$

$\hat{C}$ is the set of neighbours for the active user and implies there are some number of rated items in common between the active user and each neighbour. Users tend to use ratings scales differently. For example, on a 1 to 5 rating scale, the active user may seldom rate 1 or 5 while a neighbour only rates 1 and 5. Therefore, the average rating of the active user and current neighbour ($\bar{r}_c$ and $\bar{r}_{\acute{c}}$, respectively) are used to smooth out this inconsistency.

The Pearson coefficient (2) correlates the degree of similarity $\mathrm{sim}(\mathrm{c}, \acute{\mathrm{c}})$ between two users where $S_{c\acute{c}}$ is the set of items both users have rated in common. The degree of similarity ranges from -1 (perfect negative correlation) to +1 (perfect positive correlation). The similarity value is then used by equation (1) as the impact weight each neighbour has in determining the final predicted value (typically the N most similar neighbours are used). Thus, a neighbour with a

similarity value 1 will have a large influence in moving the predicted score towards her (relative) rating. Finally, k is a normalizing factor and is the inverse summation of the absolute similarity values.

An advantage of CF is that nothing needs to be known about the items, for items that are difficult to analyze (e.g. video) this is clearly beneficial. However, CF performs poorly under sparse rating information, especially for new items and users, and does not scale well [14].

4 Push-Poll Approach

Push-poll is envisioned to operate in a massive, highly dynamic environment, and a potential application is recommending items from *Rich Site Summary* (RSS) *feeds* [15]. RSS is a popular method to publish content to the Web and is often used by blogs and news services to alert subscribers to new entries. RSS items follow well-known XML formats and usually include a headline, a short description, and a URL to the full item of interest. The breadth of topics and overwhelming number of feeds presents an exciting challenge for a recommender system that must manage many new and diverse items per day.

Our objective is to treat recommendation as an intuitive process that results from user interaction and follows how information propagates by word of mouth in the real world. We achieve this by modeling (inferring, representing explicitly and maintaining) social networks centred on specific interests and "shepherding" relevant items through these networks.

4.1 Interest-Based Channels

A dedicated process looks for new items by routinely cycling through a list of RSS feeds (determined by developers and/or users). Once detected, new items are analyzed and separated into channels according to their content. Thus, push-poll is a *hybrid recommender system* [16] as it combines CF with *content-based analysis*.

We suggest that RSS items require only a trivial amount of content-based analysis in order to be classified. So-called *collaborative tagging systems* [17] demonstrate successful item classification by having users provide manual classification through a set of freely-chosen keywords, or *tags*, rather than relying on automated analysis or domain experts. The reoccurrence of certain tags points to a consensus regarding the item's content. Term extraction of the RSS item's headline and description would enable a rough guess as to what channel(s) the item initially "fits" into–a channel is simply represented as a unique set of tags. The overlap between the item's tags and the channel's tags defines the potential fit of the item. Later, tagging by users would overrule the system's tag set and trigger a re-examination, possibly causing the item to be introduced into other channels.

The organization of channels inherits the flexibility of collaborative tagging and allows users to freely define their interests and the scope of their interests.

For example, consider a user interested in foreign policy. She may create a channel with a general interpretation using the tag set {*foreign*, *policy*}. Or, she may desire a narrower focus and conjoin {*American*, *middleeast*} to the previous set.

4.2 Push (Diffusion)

After a new item has been matched to a channel, it is *seeded* into the corresponding network of users who subscribe to that channel (i.e., one user may create a public channel that is shared with others). For now, we assume this network already exists. Users are represented as nodes and a directed edge between nodes describes that one user *influences* another with a specific strength. Influence strength is the edge weight (ranging from -1 to +1) between a pair of user nodes and is related to the similarity between the users (Section 3.1). It determines how items will propagate through the social networks as explained by the *Independent Cascade* model [18] that captures the probability a person will chose to *adopt* an item depending on how many of her social contacts have already adopted it (note, the item could be a new hairstyle, gadget, etc.).

We use the Independent Cascade model to spread items across the network but modify the terminology to illustrate that users have no voluntary control over whether they adopt an item or not. Instead of "adopting" an item, a user is *infected* with it, and infection is a condition where the item becomes a candidate for activation (Section 4.3). For each new RSS item, some users are targeted to be initial *seeds*, i.e. the nodes that are automatically infected. We suggest some criteria for determining a potential seed: the user provides quick feedback (e.g. rates often) and acts as an authority (i.e., exerts strong, direct influence on many users). However, we leave seed determination as future work.

At the start of the push process, all seed nodes try to infect their "contacts", or neighbour nodes (i.e. the nodes at the end of outgoing edges), with the item. A seed node u infects a neighbour node v with probability $p_{u,v}$–the absolute value of the influence strength from node u to v. Infected nodes have a single attempt that will either succeed or fail at infecting a neighbour node. Success or failure is independent of all previous attempts to infect the node in question. Note, this assumption is relaxed in the *General Cascade* model [11]. After the seed nodes cannot induce any new infections, all newly infected nodes try to infect their neighbours, and this breadth-first cycle repeats until no new infections are possible. Ultimately, depending on their direct/indirect connections to seed users, some users in the network will be infected while others will not.

4.3 Poll (Activation)

If a user is infected with an item, the item is left in the user's respective channel queue. Poll is the process that ultimately activates (i.e. recommends) these queued items, and it is based on the *Threshold Model of Collective Behaviour* [19]. The model is similar to Independent Cascade and describes that node v has an intrinsic threshold level $\theta_{v,s} \in [0, 1]$ for adopting item s and a set of contacts

I that have already adopted. For each node u in I, there is an associated weight $t_{u,v}$ that describes how much "influence" u exerts on v.

Node v will adopt s if (3) holds true, i.e., the influence exerted by v's contacts is greater than v's internal resistance to adopting s. In many models, θ is randomly chosen from a distribution (uniform) to capture various levels of willingness. In our case, I is the set of infected neighbours and θ is computed as a confidence level based on some type of content analysis (e.g. comparing the item's tags against previously liked tag sets). If the system is confident that the item is relevant (e.g. $\theta < 0.25$), then the item is automatically activated. Otherwise, the active user's infected neighbours are polled using (4).

$$\sum_{u \in I} t_{u,v} \geq \theta_{v,s} \tag{3}$$

$$k \sum_{u \in I} t_{u,v} \times r_{u,s} \geq \theta_{v,s} \tag{4}$$

Equation (4) is similar to the CF prediction (1) except rating scale smoothing has been dropped and influence strengths between nodes are used instead of similarity values that are produced by (2) which is an expensive operation. The rating value $r_{u,s} \in [-1, 1]$ captures explicit feedback on the extremes (that u did or did not like the item), and implicit feedback lies on medium values following Nichols' implicit rating strength order [20]. Note, the normalizing factor k allows incoming influence strengths to sum to values greater than 1.

Determination of θ and polling is only performed when needed, i.e. when the user is active and is requesting content for the specific channel(s). While computationally efficient, there is a definite timing issue to this approach as users activating an item early in its lifetime will find infected neighbours have not yet provided feedback. One workaround would be to automatically activate the item for seed nodes, assuming these users will most likely see the item first. Otherwise, an item that failed to be activated could be saved back in the queue for a later activation attempt.

4.4 Network Feedback

Once feedback from a user for an item is recorded, her connections (influence strengths from neighbours who have also provided feedback) are updated. Feedback can be implicit (e.g. following the link of an item to the full story) or explicit (e.g. tagging an item). Note, if feedback is explicitly positive, then a "re-push" could be triggered using the active user as the new seed node. Users in agreement will see their influence strengths move to either positive or negative unity while users with low/noisy agreement will have their connections dropped. Network readjustment will ultimately affect the subsequent spread and activation of later items. In our implementation, a simple pay-off scheme is used to adjust influence strengths. However, more advanced learning algorithms could be used instead.

The question of how these networks are first formed is dependent on the types of modeling and data collection employed by the recommender system. As noted in Section 2, there are a number of means to infer social networks.

5 Evaluation

We compare the performance of a basic implementation of push-poll to the CF algorithm reviewed in Section 3.1 using a simulation. Our goals are to show that the underlying concept of social networks is feasible and to gain insight into its advantages/disadvantages.

We used the well-known *100K MovieLens* data set which contains 100,000 ratings (on a scale of 1 to 5) by 943 users for 1682 movies [21]. Each user is guaranteed to have rated a minimum of 20 movies. Data was captured during a 7 month period from September 1997 to April 1998.

The metric, mean absolute error (MAE), is used to compare performance.

$$\mathrm{MAE} = \frac{\sum_{c=1}^{N} |r_i - \acute{r}_i|}{N} \tag{5}$$

N is the total number of predictions attempted, r_i is the actual rating given by the user, and $\acute{r}_i$ is the predicted rating. Over- and under-estimation of r_i by $\acute{r}_i$ is treated the same by taking the absolute value of the difference between the two. A lower score means more accurate predictions.

Our hypothesis is that push-poll will perform as good as or better than CF at predicting ratings. In a general system, we anticipate the number of users sharing any given channel will be small. Therefore, we wish to investigate how push-poll performs in small vs. large user groups. We also hypothesize that push-poll will do better in narrow content scopes (e.g. American foreign policy vs. foreign policy) with a small group of highly interested users as connections with strong influence strengths are more likely to develop in such situations.

5.1 Simulation

We chose to classify movies by genre due to the lack of additional information in the data set like plot summaries. A general (G) and a narrow (N) genre classification were selected to represent channels a user could "subscribe" to: {*adventure*} with 135 matching *target movies* and {*science-fiction action adventure*} with 27 target movies, respectively. A target movie is a movie that's genre(s) fits those of the genre channel (note, N is a subset of G). For example, The Princess Bride (*action, adventure, children's, romance*) would be a target for G but not N.

A minimum threshold of target movies must be rated by a user before she is selected as a *target user* for the corresponding channel. If this threshold is set low then a large group of target users ($\sim$ 200) are captured, and they are conceptualized as "subscribing" to the channel with low interest (LI). If the threshold is relatively high then a small group of target users ($\sim$ 25) subscribe with high interest (HI). We assume this threshold is a proxy for interest level as users should have stronger opinions/interest within genres they rate more often.

A number of target movies were then randomly chosen as *test movies* which comprise the trial set of movies that ratings are predicted for. The training set is then comprised of the remaining target movies. Training and trial set sizes were split to capture the situations where rating information is abundant (80/20) or sparse (20/80). Altogether, there were 8 simulation configurations (2 genre channels x 2 interest levels x 2 training/trial combinations) with 5 random test movie sets run 5 times apiece for each configuration. The MAE for each of the 25 runs was averaged and reported in the next section.

For push-poll, seed nodes were randomly selected from target users for each trial movie. System parameters for push-poll were optimally set depending on the interest level: the number of seed nodes was set to ensure the majority of target users were infected ($\sim$ 10 target users for both LI and HI), and the number of infected neighbours polled was 10% of target users for LI and 20% for HI. Push-poll requires initial influence strengths between users to work with. The similarity values (2) calculated from the ratings matrix with non-target users, non-target movies, and trial movies removed were used as initial influence strengths. CF was allowed to use rating information from non-target movies in addition to what push-poll used–significantly more information.

Because an actual rating is being predicted, (1) was used by push-poll with influence strengths substituted for similarity values (activation thresholds were not determined). Each test movie had all its predictions for target users who had rated it performed sequentially (by the rating timestamp) before the next test movie was seeded. Influence strength was updated if it was determined that a pair of users had rated the same test movie. CF performed predictions in the order of the rating timestamp, regardless of the test movie. Finally, after a lapse of 24 hours, CF was allowed to update the similarity values between users using any new rating information introduced between lapses (these re-calculations took the bulk of the simulation time).

5.2 Results

Overall, push-poll significantly outperformed the CF algorithm's MAE score by an average of 1.93% ($p<0.001$). This is an encouraging result, considering the extra rating information used by CF.

The results for LI configurations are presented in the firsthalf of Figure 1. On average, push-poll consistently and significantly outperformed CF by 2.58% in these settings, and both algorithms performed better with the large training set (the 80/20 configuration). However, this intuitive expectation is reversed for CF in HI configurations: we believe correlations for a small group of users are noisy when considering all rating information (target movies and non-target movies) and prediction accuracy is largely dependent on the selected test movies (those that have lots of ratings versus few). Yet, push-poll's behaviour remains consistent for the training/trial splits but experiences increased variance in its scoring. We hypothesize that at the time of rating a user may find only a few neighbours with low influence strength who have already rated. A complete

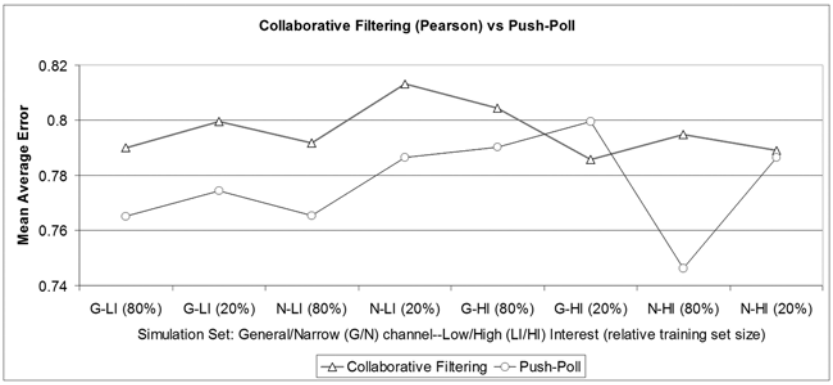

Fig. 1. Mean average error (MAE) results for simulation sets

implementation would leave that movie in the queue, waiting for feedback from stronger connections. However, further testing of this hypothesis is required.

According to our hypothesis, for small user groups (HI), we see push-poll performed better in the narrow genre channel versus the general one (an average of 1.1%). Its best performance (.7462) was in the configuration where a small group of users showed a high level of interest for specific content (with a large training set), indicating that careful selection and development of strong connections between like-minded users for a specific interest will lead to improved prediction accuracy.

6 Conclusions

We presented the design of a push-poll recommender system that supports word of mouth processes by spreading/activating information items through social networks centred on shared interests. A basic implementation of our algorithm significantly outperformed a common CF algorithm. There are a number of advantages to this approach: (1) recommendation is treated as a process and not as an outcome of pre-arranged rules, giving users some intuition over how their interactions affect which items are recommended to them, (2) new items are introduced with a minimum of content analysis; and, (3) the underlying algorithm is computationally efficient since a Pearson coefficient is not computed (we just look up the influence values on demand and do straightforward updates on any feedback). Future work includes developing push-poll into a working system that recommends items of general interest from RSS feeds.

References

1. Linden, G.: Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing 7(1), 76–80 (2003)
2. Herlocker, J.L., Konstan, J., Riedl, J.: Explaining Collaborative Filtering Recommendations. In: Proceedings of the 2000 ACM Conference on Computer Supported Collaborative Work, pp. 241–250. ACM Press, New York (2000)

3. Shardanand, U., Maes, P.: Social Information Filtering: Algorithms for Automating Word of Mouth. In: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'95), pp. 210–217. ACM Press, New York (1995)
4. Rogers, E.: Diffusion of Innovations, 5th edn. Free Press, New York (2003)
5. Mirza, B.J., Keller, B.J., Ramakrishnan, N.: Studying Recommendation Algorithms by Graph Analysis. Journal of Intelligent Information Systems 20(2), 131–160 (2003)
6. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications. Cambridge University Press, New York (1994)
7. Perugini, S., Gonçalves, M., Fox, E.: Recommender Systems Research: A Connection-Centric Survey. Journal of Intelligent Information Systems 23(2), 107–143 (2004)
8. Golbeck, J.: Generating Predictive Movie Recommendations from Trust in Social Networks. In: Proceedings of the 4th International Conference on Trust Management, Springer, Heidelberg (2006)
9. Valente, T.W.: Network Models and Methods for Studying the Diffusion of Innovations. In: Carrington, P.J., Scott, J., Wasserman, S. (eds.) Models and Methods in Social Network Analysis, New York, pp. 98–116. Cambridge University Press, Cambridge (2005)
10. Gruhl, D., Guha, R., Liben-Nowell, D., Tomkins, A.: Information Diffusion through Blogspace. In: Proceedings of the 13th International Conference on World Wide Web, pp. 491–501. ACM Press, New York (2004)
11. Kempe, D., Kleinberg, J.M., Tardos, É.: Maximizing the Spread of Influence through a Social Network. In: Proceedings KDD, pp. 137–146. ACM Press, New York (2003)
12. Schafer, B., Konstan, J., Riedl, J.: Recommender Systems in E-commerce. In: Proceedings of the 1st ACM Conf. on E-Commerce, pp. 158–166. ACM Press, New York (1999)
13. Adomavicius, G., Tuzhilin, A.: Toward the Next Generation of Recommender Systems: a Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data. Engineering 17(6), 734–749 (2005)
14. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Analysis of Recommendation Algorithms for E-Commerce. In: Proceedings of the 2nd ACM Conference on E-Commerce, pp. 158–167. ACM Press, New York (2000)
15. King, A.: The Evolution of RSS (2003) [online] [Accessed 5 November 2006] Available from: `http://www.webreference.com/authoring/languages/xml/rss/1/`
16. Burke, R.: Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction 12(4), 331–370 (2002)
17. Golder, S.A., Huberman, B.A.: Usage Patterns of Collaborative Tagging Systems. Journal of Information Science 32(2), 198–205 (2006)
18. Goldenberg, J., Libai, B., Muller, E.: Talk of the Network: A Complex Systems Look at the Underlying Process of Word-of-Mouth. Marketing Letters 12(3), 209–221 (2001)
19. Granovetter, M.: Threshold Models of Collective Behavior. The American Journal of Sociology 83(6), 1420–1443 (1978)
20. Nichols, D.M.: Implicit Rating and Filtering. In: Proceeding of the 5th DELOS Workshop on Filtering & Collaborative Filtering, Budapest, Hungary, pp. 31–36 (1997)
21. GroupLens Research Project: (2003) GroupLens Home Page [Accessed 5 November 2006] [online] Available from: `http://www.grouplens.org`