Bulletin of the Technical Committee on

Data Engineering

June 2008 Vol. 31 No. 2

IEEE Computer Society

Letters

Special Issue on Recommendation and Search in Social Systems

Conference and Journal Notices

The TC on Data Engineering
Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems. The TC on Data Engineering web page is
http://tab.computer.org/tcde/index.html.

The Data Engineering Bulletin
The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

The Data Engineering Bulletin web site is at
http://tab.computer.org/tcde/bull_about.html.

Letter from the Editor-in-Chief

Changing Editors

Editors for the Data Engineering Bulletin serve two year appointments and are responsible for two issues over that time on topics that they select with a bit of coordination from me. This means that with some regularity (on a two year cycle) I have the task, which is actually a pleasure, of thanking outgoing editors for working hard in producing the special issues that capture the current state of the art in exciting areas of current interest in our field. The outgoing editors during this cycle are Natasha Ailamaki, Jayant Haritsa, Nick Koudas, and Dan Suciu. The success of the Bulletin depends absolutely on having great editors like these. Over this past two years, these editors have brought you issues on data quality, data provenance, self managing systems, web scale systems, multi-lingual systems, mobile systems, and applications in social sciences. Thank you Natasha, Jayant, Nick, and Dan for serving the Bulletin so well during your terms as editor.

Selecting editors is the most important task that I have. I have been very lucky to have been able over the years to find outstanding editors who have produced wonderful issues of the Bulletin. My luck seems to be holding, and I can proudly announce the appointment of new editors for the next two years. The editors are Sihem Amer-Yahia of Yahoo!, Beng Chin Ooi of the National University of Singapore, Jianwen Su of UC Santa Barbara, and Vassilis Tsotras of UC Riverside. The new editors all have outstanding research reputations and great professional visibility. I welcome them to the Bulletin and look forward to working with them over the next two years to continue producing the fine special topic issues that make the Bulletin the unique publication that it is.

The Current Issue

There are an increasing number of sites on the web involving social networking and user provided content. A real challenge in this environment is to not only find the sites of interest but to gauge how useful the vast amount of content on these sites may be. It is clearly impossible to personally review it all. Even doing a search frequently produces an overwhelming number of ”answers”, the number being large enough to make the results at times of limited use. Recommendation functionality is increasingly provided by these sites to improve the prospects of users finding the information that they want or connecting to others with similar interests, etc. Recommendation systems is the topic of the current issue.

Sihem Amer-Yahia has succeeded in enticing authors from some of the leading vendors as well as leading researchers in this new space to contribute articles showcasing their efforts in recommendation systems. The current issue shows some of the diverse approaches to research, implementation and deployment of such systems. It continues what I regard as the unique character of the Bulletin in tapping both reseach and industrial work to give a clearer and more complete picture of the field. I want to thank Sihem for doing a fine job on this issue and can ”recommend” the issue to you, our readers, in what is my first contribution to the field– i.e. recursive recommendation!

David Lomet
Microsoft Corporation

Letter from the Special Issue Editor

Social systems are becoming the preferred destinations to share content (whether they are generated by the user as in Flickr and YouTube, or by other means as in the case of del.icio.us), express opinions (in the form of tagging, rating, and/or reviewing), and build connection with other users (whether they are real-life friends or merely people with similar interests). Finding interesting and relevant content on those sites, however, has become increasingly difficult due to the enormous amount of high quality content available. There are three main channels for finding content in those sites: *browsing*, *searching*, and *being served with recommendations*. Search requires to be revisited in a context where the opinion of other users matters. Recommendation has been receiving growing attention lately. It is therefore not surprising that more and more sites have begun to adopt recommendation as one of the core mechanisms with which they present the user with content. The ability to understand how search and recommendation interact, in particular, in the presence of social ties, is crucial for the survival of those user-driven sites.

This issue is a call to the database community to learn about recommender systems and incorporate social aspects in database research. Social systems constitute a great opportunity for socially-inspired research and a great source of data. This issue is a good start towards the understanding of social databases. It first presents two papers which contain an overview of recommendations strategies and state-of-the-art solutions for robustness, an important quality management issue. The third paper describes a scalable and efficient recommendation infrastructure already in use at Yahoo! The "social" aspect becomes more prominent with the fourth contribution which reports on a user study of the interaction between search and recommendation, followed by an IR-inspired approach for socially-aware search, and finally, a proposal for a social SQL.

I would like to thank the authors who graciously volunteered their time and effort in putting together this special issue.

Sihem Amer-Yahia
Yahoo! Research
New York City, USA

A Survey of Collaborative Recommendation and the Robustness of Model-Based Algorithms*

J.J. Sandvig and Bamshad Mobasher and Robin Burke
DePaul University
School of Computer Science, Telecommunications
and Information Systems
{jsandvig,mobasher,rburke}@cs.depaul.edu

Abstract

The open nature of collaborative recommender systems allows attackers who inject biased profile data to have a significant impact on the recommendations produced. Standard memory-based collaborative filtering algorithms, such as k-nearest neighbor, are quite vulnerable to profile injection attacks. Previous work has shown that some model-based techniques are more robust than standard k-nn. Model abstraction can inhibit certain aspects of an attack, providing an algorithmic approach to minimizing attack effectiveness. In this paper, we examine the robustness of several recommendation algorithms that use different model-based techniques: user clustering, feature reduction, and association rules. In particular, we consider techniques based on k-means and probabilistic latent semantic analysis (pLSA) that compare the profile of an active user to aggregate user clusters, rather than the original profiles. We then consider a recommendation algorithm that uses principal component analysis (PCA) to calculate the similarity between user profiles based on reduced dimensions. Finally, we consider a recommendation algorithm based on the data mining technique of association rule mining using the Apriori algorithm. Our results show that all techniques offer large improvements in stability and robustness compared to standard k-nearest neighbor. In particular, the Apriori algorithm performs extremely well against low-knowledge attacks, but at a cost of reduced coverage, and the PCA algorithm performs extremely well against focused attacks. Furthermore, our results show that all techniques can achieve comparable recommendation accuracy to standard k-nn.

1 Introduction

A widely accepted approach to user-based collaborative filtering is the k-nearest neighbor algorithm. However, memory-based algorithms such as k-nn do not scale well to commercial recommender systems. Model-based algorithms are widely accepted as a way to alleviate the scaling problem presented by memory-based algorithms in data-intensive commercial recommender systems. Building a model of the dataset allows off-line processing for the most rigorous similarity calculations. In some cases, this is at the cost of lower recommendation accuracy [1].

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*This work was supported in part by the National Science Foundation Cyber Trust program under Grant IIS-0430303.

A positive side effect of a model-based approach is that it may provide improved robustness against attacks. An adaptive system dependent on anonymous, unauthenticated user profiles is subject to manipulation. The standard collaborative filtering algorithm builds a recommendation for a target user by combining the stored preferences of peers with similar interests. If a malicious user injects the profile database with a number of fictitious identities, they may be considered peers to a genuine user and bias the recommendation. We call such attacks *profile injection attacks* (also known as *shilling* [2]).

Recent research has shown that surprisingly modest attacks are sufficient to manipulate the most common CF algorithms [3, 2, 4, 5]. Profile injection attacks degrade the objectivity and accuracy of a recommender system over time, causing frustration for its users and potentially leading to high user defection. However, a model-based approach is an abstraction of detailed user profiles. We hypothesize that this abstraction minimizes the influence of an attack, because attack profiles are not directly used in recommendation.

In our study, we have focused on the robustness of user clustering, feature reduction, and association rules. We first consider techniques based on k-means clustering and probabilistic latent semantic analysis (pLSA) that compare the profile of an active user to aggregate user clusters, rather than the original profiles. Probabilistic latent semantic analysis is used infer hidden relationships among groups of users, which are then used to form "fuzzy" clusters. Each user has a degree of association with every cluster, allowing particularly authoritative users to exercise greater influence on recommendation.

We then consider a recommendation algorithm that uses principal component analysis (PCA) to calculate the similarity between user profiles based on reduced dimensions. Principal component analysis tries to extract a set of uncorrelated factors from a given set of multicolinear variables. By keeping only those principal components that explain the greatest amount of variance in the data, we effectively reduce the number of features that must be used for a similarity calculation.

Finally, we consider a recommendation algorithm based on the data mining technique of association rule mining using the Apriori algorithm. Association rule mining is a technique common in data mining that attempts to discover patterns of products that are purchased together. These relationships can be used for myriad purposes, including marketing, inventory management, etc. We have adapted the Apriori algorithm [6] to collaborative filtering in an attempt to discover patterns of items that have common ratings.

The primary contribution of this paper is to demonstrate that model-based algorithms provide an algorithmic approach to robust recommendation. Our results show that all techniques offer large improvements in stability and robustness compared to standard k-nearest neighbor. In particular, the Apriori and PCA algorithms performs extremely well against low-knowledge attacks, but in the case of Apriori at a cost of reduced coverage, and the k-means and pLSA algorithms perform extremely well against focused attacks. Furthermore, our results show that all techniques can achieve comparable recommendation accuracy to standard k-nn.

2 Recommendation Algorithms

In general, user-based collaborative filtering algorithms attempt to discover a neighborhood of user profiles that are similar to a target user. A rating value is then predicted for all missing items in the target user's profile, based on ratings given to the item within the neighborhood. We begin with background information on the standard memory-based k-nn. We then present several recommendation algorithms based on model-based techniques of user clustering (k-means and pLSA), feature reduction (PCA), and association rules (Apriori).

2.1 k-Nearest Neighbor

The standard k-nearest neighbor algorithm is widely used and reasonably accurate [7]. Similarity between the target user, u, and a neighbor, v, is computed using Pearson's correlation coefficient:

$$sim_{u,v} = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u) * (r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2} * \sqrt{\sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}} \tag{1}$$

where $r_{u,i}$ and $r_{v,i}$ are the ratings of some item i for u and v, respectively; and $\bar{r}_u$ and $\bar{r}_v$ are the average of the ratings of u and v over I, respectively.

After similarities are calculated, the k most similar users that have rated the target item are selected as the neighborhood. This implies a target user may have a different neighborhood for each target item. It is also common to filter neighbors with similarity below a specified threshold. This prevents predictions being based on very distant or negative correlations. After identifying a neighborhood, we compute the prediction for a target item i and target user u as follows:

$$pred_{u,i} = \bar{r}_u + \frac{\sum_{v \in V} sim_{u,v}(r_{v,i} - \bar{r}_v)}{\sum_{v \in V} |sim_{u,v}|} \tag{2}$$

where V is the set of k similar neighbors that have rated i; $r_{v,i}$ is the rating of i for neighbor v; $\bar{r}_u$ and $\bar{r}_v$ are the average ratings over all rated items for u and v, respectively; and $sim_{u,v}$ is the Pearson correlation between u and v. The formula computes the degree of preference for all neighbors, weighted by their similarity, and then adds this to the target user's average rating.

2.2 k-Means Clustering

A standard model-based collaborative filtering algorithm uses k-means to cluster similar users. Given a set of user profiles, the space can be partitioned into k groups of users that are close to each other based on a measure of similarity. The discovered user clusters are then applied to the user-based neighborhood formation task, rather than individual profiles.

To make a recommendation for a target user u and target item i, we select a neighborhood of user clusters that have a rating for i and whose aggregate profile v_k is most similar to u. This neighborhood represents the set of user segments that the target user is most likely to be a member, based on a measure of similarity. For this task, we use Pearson's correlation coefficient. We can now make a prediction for item i as described in the previous section, where the neighborhood V is the set of user cluster aggregate profiles most similar to the target user.

2.3 Probabilistic Latent Semantic Analysis

Probabilistic latent semantic analysis (pLSA) models [8] provide a probabilistic approach for characterizing latent or hidden semantic associations among co-occurring objects. We have applied pLSA to the creation of user clusters in the context of collaborative filtering [9].

Given a set of n users, $U = \{u_1, u_2, \cdots, u_n\}$, and a set of m items, $I = \{i_1, i_2, \cdots, i_m\}$ the pLSA model associates an unobserved factor variable $Z = \{z_1, z_2, \cdots, z_l\}$ with observations in the rating data. For a target user u and a target item i, the following joint probability can be defined:

$$P(u,i) = \sum_{k=1}^{l} Pr(z_k) \cdot Pr(u|z_k) \cdot Pr(i|z_k) \tag{3}$$

In order to explain a set of ratings (U, I), we need to estimate the parameters $Pr(z_k)$, $Pr(u|z_k)$, and $Pr(i|z_k)$, while maximizing the likelihood of the rating data $L(U, I) = \sum_{u \in U} \sum_{i \in I} r_{u,i} \cdot \log Pr(u, i)$ where $r_{u,i}$ is the rating of user u for item i. The Expectation-Maximization (EM) algorithm is used to perform maximum likelihood parameter estimation, based on initial values of $Pr(z_k)$, $Pr(u|z_k)$, and $Pr(i|z_k)$. Iterating the expectation and maximization steps monotonically increases the total likelihood of the observed data $L(U, I)$, until a local optimum is reached.

We now identify clusters of users that have similar underlying interests. For each latent variable z_k, we create a user cluster C_k and select all users having probability $Pr(u|z_k)$ exceeding a certain threshold μ. If a user does not exceed the threshold for any latent variable, it is associated with the user cluster of highest probability. Thus, every user profile will be associated with at least one user cluster, but may be associated with multiple clusters. This allows authoritative users to have broader influence over predictions, without adversely affecting coverage in sparse rating data. A recommendation is made for a target user u and target item i in a similar manner to k-means clustering.

2.4 Principal Component Analysis

Principal component analysis is a dimensionality reduction technique that tries to extract a set of uncorrelated factors from a given set of multicolinear variables. Each factor represents a latent pattern that is explained by the degree of correlation to the explicit variables. Factor analysis in general assumes there is an underlying structure to the explicit variables. In a recommendation context, the factors may represent fine-grained groupings of items. For example, movies may have implicit groupings such as style and genre.

PCA identifies the orthogonal axes of variance within a dataset, where the first axis represents the largest variance in the data, the second axis represents the second largest variance, and so on. It is based on a theorem of linear algebra stating that for any real symmetric matrix A, there exists a unitary matrix Λ such that $\Sigma = \Lambda^T A \Lambda$ and Σ is diagonal.

A solution is found using the eigenvectors of A, where the columns of Λ are the eigenvectors ordered in decreasing eigenvalues. Then, $\Sigma_{ii} = \lambda_i$ is the i^{th} largest eigenvalue of A. The principal components are calculated using the covariance matrix of the user data U with respect to items, such that $A = \frac{1}{n-1} U^T U$. Prior to calculating the covariance matrix, it is important to adjust the matrix U such that each item vector is zero-mean. For each u_i in U, modify the user vector such that $u'_i = u_i - m$ where $m = \frac{1}{n} \sum_{i=0}^{n} u_i$ is the vector of item means.

A caveat to this approach is the potential effect of missing data. Collaborative filtering datasets are notoriously sparse, but PCA requires a dense covariance matrix to calculate eigenvectors. We have resolved this issue with an elegant solution. Before adjusting for item means and calculating the covariance matrix, we subtract each user's mean rating from the user vector, where the mean is calculated by ignoring the missing ratings. The idea is that different users may have different "baselines" around which their ratings are distributed. We then set all missing values to 0 under the assumption that a user has no preference for an item that has not been rated.

In order to reduce the number of dimensions in the feature vector Λ, we simply keep the eigenvectors with the largest eigenvalues and discard the rest. There are several ways to choose the number of eigenvectors to keep for PCA, but in our experiments we have found the percentage of variance criteria to yield the most accurate results. We keep the number of eigenvectors such that the total cumulative percentage of variance surpasses some threshold, μ.

To calculate a prediction for a target item i and target user u, we modify the standard k-nn algorithm, such that Equation 1 is calculated with respect to the reduced dimension vectors of user u and neighbor v. Each reduced dimension vector is calculated as $u' = \Lambda'^T (u - m)$, where Λ' is the reduced dimension feature vector; m is the vector of item means; and u is the target user or neighbor vector, mean adjusted according to that user's mean.

2.5 Association Rule Mining

Association rule mining is a common technique for performing market basket analysis. The intent is to capture relationships among items based on patterns of co-occurrence across transactions. We have applied association rules to the context of collaborative filtering [10]. Considering each user profile as a transaction, it is possible to use the Apriori algorithm [6] and generate association rules for groups of commonly liked items.

Given a set of user profiles U and a set of item sets $I = \{I_1, I_2, \ldots, I_k\}$, the *support* of an item set $I_i \in I$ is defined as $\sigma(I_i) = |\{u \in U : I_i \subseteq u\}| \,/\, |U|$. Item sets that satisfy a minimum support threshold are used to generate association rules. These groups of items are referred to as *frequent item sets*. An association rule r is an expression of the form $X \implies Y(\sigma_r, \alpha_r)$, where X and Y are item sets, σ_r is the support of $X \cup Y$, and α_r is the *confidence* for the rule r given by $\sigma(X \cup Y)/\sigma(X)$. In addition, association rules that do not satisfy a minimum *lift* threshold are pruned, where lift is defined as $\alpha_r/\sigma(Y)$.

Before performing association rule mining on a collaborative filtering dataset, it is necessary to discretize the rating values of each user profile. We first subtract each user's average rating from the ratings in their profile to obtain a zero-mean profile. Next, we give a discrete category of "like" or "dislike" to each rated item in the profile if it's rating value is $>$ or $\leq$ zero, respectively. In classic market basket analysis, it is assumed that a customer will not purchase an item they do not like. Hence, a transaction always contains implicit positive ratings. However, when dealing with explicit rating data, certain items may be disliked. A collaborative recommender must take such preference into account or risk recommending an item that is rated often, but disliked by consensus.

To make a recommendation for a target user profile u, we create a set of candidate items C such that an association rule r exists of the form $X \subseteq u \implies i \in C$ where i is an unrated item in the profile u. In practice, it is not necessary to search every possible association rule given u. It is sufficient to find all frequent item sets $X \subseteq u$ and base recommendations on the next larger frequent itemsets $Y \supset X$ where Y contains some item i that is unrated in u. The candidate set C is then sorted according to confidence scores and the top N items are returned as a recommendation.

A caveat to this approach is the possibility of conflicting recommendations in the candidate set C. For example, one association rule may add item i to the candidate set with a "like" label, whereas another rule may add the same item with a "dislike" label. There is no ideal solution, but we have chosen to assume that there are opposing forces for the recommendation of the item. In our implementation, we subtract the confidence value of the "dislike" label from the confidence value of the "like" label.

3 Profile Injection Attacks

A collaborative recommender database consists of many user profiles, each with assigned ratings to a number of products that represent the user's preferences. A malicious user may insert multiple profiles under false identities designed to bias the recommendation of a particular item for some economic advantage. This may be in the form of an increased number of recommendations for the attacker's product, or fewer recommendations for a competitor's product.

3.1 An Example

Consider an example recommender system that identifies interesting books for a user. The representation of a user profile is a set of product / rating pairs. A rating for a particular book can be in the range 1-5, where 5 is the highest possible rating. Alice, having built a profile from previous visits, returns to the system for new recommendations. Figure 1 shows Alice's profile along with that of seven genuine users.

An attacker, Eve, has inserted three profiles (Attack1-3) into the system to mount an attack promoting the target item, Item6. Each attack profile gives high ratings to Eve's book, labeled Item6. If the attack profiles are constructed such that they are similar to Alice's profile, then Alice will be recommended Eve's book. Even

	Item1	Item2	Item3	Item4	Item5	Item6	Correlation with Alice
Alice	5	2	3	3		?	
User1	2		4		4	1	-1.00
User2	3	1	3		1	2	0.76
User3	4	2	3	1		1	0.72
User4	3	3	2	1	3	1	0.21
User5		3		1	2		-1.00
User6	4	3		3	3	2	0.94
User7		5		1	5	1	-1.00
Attack1	**5**		**3**		**2**	**5**	1.00
Attack2	**5**	**1**	**4**		**2**	**5**	0.89
Attack3	**5**	**2**	**2**	**2**		**5**	0.93
Correlation with Item6	0.85	-0.55	0.00	0.48	-0.59		

Figure 1: an example attack on Item6

without direct knowledge of Alice's profile, similar attack profiles may be constructed from average or expected ratings across all users.

Disregarding Eve's attack profiles for a moment, we can compute Alice's predicted preference for Item6. Assuming 1-nearest neighbor, Alice will not be recommended Item6. The most highly correlated user to Alice is User6, who clearly does not like Item6. Therefore, Alice is expected to dislike Item6.

After Eve's attack, however, Alice receives a very different recommendation. As a result of including the attack profiles, Alice is most highly correlated to Attack1. In this case, the system predicts Alice will like Item6 because it is rated highly by Attack1. She is given a recommendation for Item6, although it is not the ideal suggestion. Clearly, this can have a profound effect on the effectiveness of a recommender system. Alice may find the suggestion inappropriate, or worse; she may take the system's advice, buy the book, and then be disappointed by the delivered product.

3.2 Attack Types

A variety of attack types have been studied for their effectiveness against different recommendation algorithms [4, 5]. An *attack type* is an approach to constructing attack profiles, based on knowledge about the recommender system, its rating database, its products, and/or its users. In a push attack, the target item is generally given the maximum allowed rating. The set of *filler items* represents a group of selected items in the database that are assigned ratings within the attack profile. Attack types can be characterized according to the manner in which they choose filler items, and the way that specific ratings are assigned. In this paper, we focus on three attack types that have been shown to be very effective against standard user-based collaborative filtering recommenders.

The random attack is a basic attack type that assigns random ratings to filler items, distributed around the global rating mean [2, 4]. The attack is very simple to implement, but has limited effectiveness.

The average attack attempts to mimic general user preferences in the system by drawing its ratings from the rating distribution associated with each filler item [2, 4]. An average attack is much more effective than a random attack; however, it requires greater knowledge about the system's rating distribution. In practice, the additional knowledge cost is minimal. An average attack can be quite successful with a small filler item set, whereas a random attack usually must have a rating for every item in the database in order to be effective.

An attacker may be interested primarily in a particular set of users – likely buyers of a product. A segment attack attempts to target a specific group of users who may already be predisposed toward the target item [4]. For example, an attacker that wishes to push a fantasy book might want the product recommended to users expressing interest in *Harry Potter* and *Lord of the Rings*. A typical segment attack profile consists of a number of selected items that are likely to be favored by the targeted user segment, in addition to the random filler items. Selected items are expected to be highly rated within the targeted user segment and are assigned the maximum rating value along with the target item.

4 Experimental Evaluation

To evaluate the robustness of model-based techniques, we compare the results of push attacks using different parameters. In each case, we report the relative improvement over the k-nearest neighbor approaches.

4.1 Dataset

In our experiments, we have used the publicly-available Movie-Lens 100K dataset[1]. This dataset consists of 100,000 ratings on 1682 movies by 943 users. All ratings are integer values between one and five, where one is the lowest (disliked) and five is the highest (liked). Our data includes all users who have rated at least 20 movies.

To conduct attack experiments, the full dataset is split into training and test sets. Generally, the test set contains a sample of 50 user profiles that mirror the overall distribution of users in terms of number of movies seen and ratings provided. The remaining user profiles are designated as the training set. All attack profiles are built from the training set, in isolation from the test set.

The set of attacked items consists of 50 movies whose ratings distribution matches the overall ratings distribution of all movies. Each movie is attacked as a separate test, and the results are aggregated. In each case, a number of attack profiles are generated and inserted into the training set, and any existing rating for the attacked movie in the test set is temporarily removed.

For every profile injection attack, we track *attack size* and *filler size*. Attack size is the number of injected attack profiles, and is measured as a percentage of the pre-attack training set. There are approximately 1000 users in the database, so an attack size of 1% corresponds to about 10 attack profiles added to the system. Filler size is the number of filler ratings given to a specific attack profile, and is measured as a percentage of the total number of movies. There are approximately 1700 movies in the database, so a filler size of 10% corresponds to about 170 filler ratings in each attack profile. The results reported below represent averages over all combinations of test users and attacked movies.

4.2 Evaluation Metrics

There has been considerable research on the accuracy and performance of recommender systems [11]. We use the mean absolute error (MAE) accuracy metric, a statistical measure for comparing predicted values to actual user ratings.We define coverage as the percentage of items in the database for which an algorithm is able to make a prediction.

However, our overall goal is to measure the effectiveness of an attack; the "win" for the attacker. In the experiments reported below, we measure hit ratio - the average likelihood that a top n recommender will recommend a pushed item, compared to all other items.

Hit ratio measures the effectiveness of an attack on a pushed item compared to other items. Let R_u be the set of top n recommendations for user u. For each push attack on item i, the value of a recommendation hit for user u denoted by H_{ui}, can be evaluated as 1 if $i \in R_u$; otherwise H_{ui} is evaluated to 0. We define hit ratio as the number of hits across all users in the test set divided by the number of users in the test set. The hit ratio for a pushed item i over all users in a set can be computed as $\sum H_{ui}/|U|$. Average hit ratio is calculated as the sum of the hit ratio for each push attack on item i across all pushed items divided by the number of pushed items.

Hit ratio is useful for evaluating the pragmatic effect of a push attack on recommendation. Typically, a user is only interested in the top 20 to 50 recommendations. An attack on an item that significantly raises the hit

[1] http://www.cs.umn.edu/research/GroupLens/data/

ratio, regardless of prediction shift, can be considered effective. Indeed, an attack that causes a pushed item to be recommended 80% of the time has achieved the desired outcome for the attacker.

4.3 Accuracy Analysis

We first compare the accuracy of k-nn versus the model-based algorithms. To monitor accuracy, and to assist in tuning the recommendation algorithms, we use MAE. In all cases, 10-fold cross-validation is performed on the entire dataset and no attack profiles are injected.

In neighborhood formation, we achieved optimal results using $k = 20$ users for the neighborhood size of the k-nn algorithm. For the model-based algorithms, we obtained the most favorable results using $k = 10$ user segments for the neighborhood size. In all cases, we filter out neighbors with a similarity score less than 0.1. For pLSA, we observed an optimum threshold of $\mu = 0.035$. We obtained the best results for PCA by extracting the factors that explain at least 60% of total variance. The average number of extracted principal components was approximately 100.

Table 1 displays the results from one of five test runs performed. The difference in accuracy between the standard and PCA approaches is not statistically significant. This is a very promising result, as the scalability of model-based algorithms often come at the cost of lower recommendation accuracy [1]. For example, k-means and pLSA show small decreases in accuracy compared to standard k-nn.

Determining a suitable evaluation metric for the Apriori recommender was challenging because it is based on a fundamentally different approach. The k-nn algorithm predicts a rating value for each target item and ranks all items based on this score. The association rule algorithm produces a ranked list, such that the recommendation score is the confidence that a target user will like the recommended item. It is not possible to make a prediction of the rating value from the association rule recommendation list. However, the association rule recommender does make a more general prediction; it predicts a binary "like" or "dislike" classification for a recommended item if the confidence value is positive or negative, respectively.

For brevity, we do not include the derived metric, but a detailed description can be found in [10]. Our results showed the difference in accuracy between the association rule recommender and k-nn to be statistically insignificant. But because Apriori selects recommendations from only among those item sets that have met the support threshold, it will by necessity have lower coverage than the other model-based algorithms. There will be some items that do not appear and about which the algorithm cannot make any predictions. This problem may occur in a k-nn algorithm as well, since there may be no peer users who have rated a given item. However, this is a relatively rare occurrence. The coverage of the k-nn algorithm is near 100%, while Apriori is consistently around 47%.

The Apriori algorithm would therefore lend itself best to scenarios in which a list of top recommended items is sought, rather than a rating prediction scenario in which the recommender must be able to estimate a rating for any given item. The selectivity of the algorithm may be one reason to expect it will be relatively robust - it will not make recommendations without evidence that meets the minimum support threshold.

4.4 Robustness Analysis

To evaluate the robustness of model-based algorithms, we compare the results of push attacks on collaborative recommendation algorithms using k-nearest neighbor, k-means clustering, pLSA, PCA, and Apriori techniques.

Table 1: Accuracy

	k-nn	k-means	plsa	pca
mae	0.7367	0.7586	0.7467	0.7327

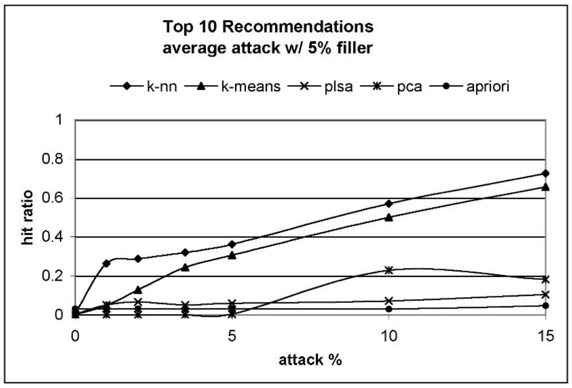


Figure 2: Average attack hit ratio at 5% filler size

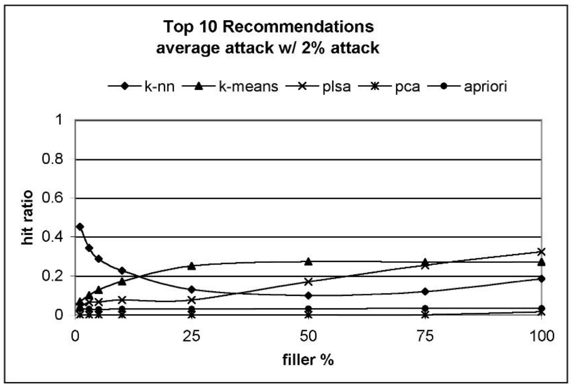


Figure 3: Average attack hit ratio at 2% attack size

We report the results for average and segment attacks, and exclude results for random attack, because average attack is more effective and exhibits similar robustness trends.

4.4.1 Average Attack

Figure 2 presents hit ratio results for top 10 recommendations at different attack sizes, using a 5% filler. With the exception of k-means, the model-based techniques show notable improvement in stability over k-nn. Apriori and pLSA, in particular, have superior performance at all attack sizes, and PCA performs extremely well at small attack sizes of 5% or less. Under a 15% attack, an attacked movie is in a user's top 10 recommended list nearly 80% of the time for k-nn and k-means. However, the attacked movie only shows up in a user's top 10 recommendations slightly greater than 5% of the time for Apriori or pLSA and less that 20% of the time for PCA.

Robustness of the Apriori algorithm may be partially due to lower coverage. However, this does not account for the flat trend of hit ratio with respect to attack size. At a 5% attack, we observed only 26% coverage of the attacked item. But at a 10% attack, we observed 50% coverage, and at 15% attack, we observed a full 100% coverage of the attacked item.

It is precisely the manner in which an average attack chooses filler item ratings that causes the combination of multiple attack profiles to short-circuit the attack. Recall that filler item ratings in an average attack are distributed around their mean rating. When an average attack profile is discretized, there is equal probability that a filler item will be categorized as "like" or "dislike". Therefore, multiple attack profiles will show little more than chance probability of having common itemsets. The lack of mutual reinforcement between filler items prevents the support of itemsets containing the attacked item from surpassing the threshold.

To evaluate the sensitivity of filler size, we have tested a full range of filler items. The 100% filler is included as a benchmark for the potential influence of an attack. However, it is not likely to be practical from an attacker's point of view. Collaborative filtering rating databases are often extremely sparse, so attack profiles that have rated every product are quite conspicuous.Of particular interest are smaller filler sizes. An attack that performs well with few filler items is less likely to be detected. Thus, an attacker will have a better chance of actually impacting a system's recommendation, even if the performance of the attack is not optimal.

Figure 3 depicts hit ratio for top 10 recommendations at the full range of filler sizes with a 2% attack size. Surprisingly, as filler size is increased, hit ratio for standard k-nn goes down. This is because an attack profile with many filler items has greater probability of being dissimilar to the active user. On the contrary, hit ratio for k-means and pLSA tend to rise with larger filler sizes. Eventually, both algorithms are surpassed by k-nn and actually perform worse with respect to robustness.

However, the PCA and Apriori algorithms hold steady at large filler sizes and are essentially unaffected. As

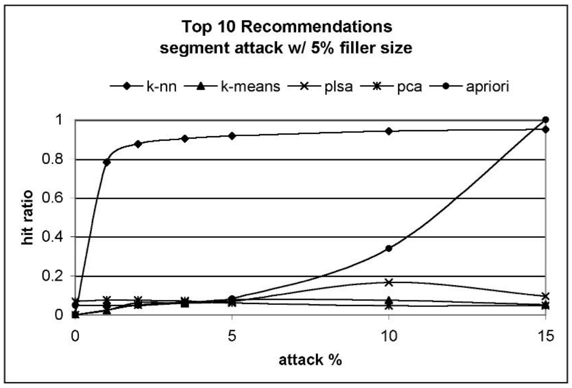


Figure 4: Segment attack hit ratio at 5% filler size

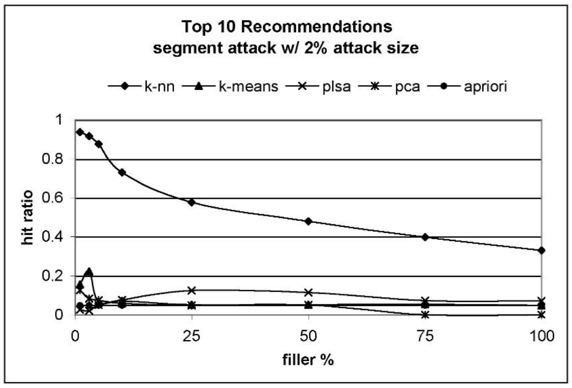


Figure 5: Segment attack hit ratio at 2% attack size

with attack size, the reason that filler size does not affect the robustness of Apriori is because adding more filler items does not change the probability that multiple attack profiles will have common itemsets. The fact that a profile's ratings are discretized to categories of "like" and "dislike" means that an attack profile with 100% filler size will cover exactly half of the total features used in generating frequent itemsets. Therefore, it is very unlikely that multiple attack profiles will result in mutual reinforcement.

4.4.2 Segment Attack

The segment attack is designed to have particular impact on likely buyers, or "in-segment" users. These users have shown a disposition towards items with particular characteristics, such as movies within a particular genre. For our experiments, we selected popular horror movies (Alien, Psycho, The Shining, Jaws, and The Birds) and identified users who had rated all of them as 4 or 5. This is an ideal target market to promote other horror movies, and so we measure the impact of the attack on recommendations made to the in-segment users.

Figure 4 depicts hit ratio for top 10 recommendations at different attack sizes, using a 5% filler. Clearly, the attack is extremely effective against the k-nn algorithm. A meager 1% attack shows a hit ratio of nearly 80%. By contrast, a segment attack has little effect on k-means, pLSA, and PCA.

The Apriori algorithm appears to have the same robustness as the other model-based algorithms at small attack sizes. However, beyond a 5% attack, Apriori performs quite poorly with respect to robustness. Hit ratio reaches 100% at a 15% attack. The cause of such dramatic effect is precise targeting of selected items by the attacker. This is the opposing force to the phenomena witnessed against an average attack. A segment attack profile consists of multiple selected items, in addition to the target item, where the maximum rating is assigned. Clearly, all such items will always be categorized as "like". Therefore, the mutual reinforcement of common item sets is a given, and a user that likes any permutation of the selected items receives the attacked item as a recommendation with high confidence.

Although the performance of Apriori is not ideal against a segment attack, certain scenarios may minimize the performance degradation in practice. In particular, a recommender system with a very large number of users is somewhat buffered from attack. The algorithm is quite robust through a 5% attack, and is comparable to both k-means, pLSA, and PCA. The robustness of Apriori is not drastically reduced until attack size is 10% or greater. Certainly it is feasible for an attacker to inject the necessary number of profiles into a recommender with a small number of users, but it may not be economical for a commercial recommender such as Amazon.com, with millions of users.

5 Conclusion

The standard user-based collaborative filtering algorithm has been shown quite vulnerable to profile injection attacks. An attacker is able to bias recommendation by building a number of profiles associated with fictitious identities. In this paper, we have demonstrated the relative robustness and stability of model-based algorithms over the memory-based approach.

References

[1] M. O'Conner and J. Herlocker, "Clustering items for collaborative filtering," in *Proceedings of the ACM SIGIR Workshop on Recommender Systems*, Berkeley, CA, August 1999.

[2] S. Lam and J. Riedl, "Shilling recommender systems for fun and profit," in *Proceedings of the 13th International WWW Conference*, New York, May 2004.

[3] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre, "Collaborative recommendation: A robustness analysis," *ACM Transactions on Internet Technology*, vol. 4, no. 4, pp. 344–377, 2004.

[4] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, "Towards trustworthy recommender systems: An analysis of attack models and algorithm robustness," *ACM Transactions on Internet Technology*, vol. 7, no. 4, 2007.

[5] B. Mobasher, R. Burke, R. Bhaumik, and J. Sandvig, "Attacks and remedies in collaborative recommendation," *IEEE Intelligent Systems*, vol. 22, no. 3, pp. 56–63, May/June 2007.

[6] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*, Santiago, Chile, September 1994.

[7] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR'99)*, Berkeley, CA, August 1999.

[8] T. Hofmann, "Probabilistic latent semantic analysis," in *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, Stockholm, Sweden, July 1999.

[9] B. Mobasher, R. Burke, and J. Sandvig, "Model-based collaborative filtering as a defense against profile injection attacks," in *Proceedings of the 21st National Conference on Artificial Intelligence*. AAAI, July 2006, pp. 1388–1393.

[10] J. J. Sandvig, B. Mobasher, and R. Burke, "Robustness of collaborative recommendation based on association rule mining," in *Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys'07)*, October 2007, pp. 105–111.

[11] J.Herlocker, J. Konstan, L. G. Tervin, and J. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 5–53, 2004.

A Survey of Attack-Resistant Collaborative Filtering Algorithms

Bhaskar Mehta Thomas Hofmann
Google Inc.
Brandschenkestrasse 110
Zurich, Switzerland - 8004
{bmehta,thofmann}@google.com

Abstract

With the increasing popularity of recommender systems in commercial services, the quality of recommendations has increasingly become an important to study, much like the quality of search results from search engines. While some users faithfully express their true opinion, many provide noisy or incorrect ratings which can be detrimental to the quality of the generated recommendations. The presence of noise can violate modeling assumptions and may thus result in unstable estimates or predictions. Even worse, malicious users can deliberately insert attack profiles in an attempt to bias the recommender system to their benefit. This is a particularly important issue, and it is necessary for systems to provide guarantees on the robustness of recommendations to ensure continued user trust. While previous research has attempted to study the robustness of various existing Collaborative Filtering (CF) approaches, the explicit design of robust recommender systems remains a challenging problem. Approaches such as Intelligent Neighborhood Selection, Association Rules and Robust Matrix Factorization are generally known to produce unsatisfactory results. In this paper, we review previous approaches to robust collaborative filtering; we also describe promising recent approaches that exploit a Singular Value Decomposition (SVD) and are both accurate as well as highly stable to shilling.

1 Introduction

Collaborative filtering technology is being widely used on the web as an approach to information filtering and recommendation by commercial service providers like *Amazon* and *Yahoo!*. For multimedia data like music and video, where pure content-based recommendations perform poorly, collaborative filtering is the most viable and effective solution, and is heavily used by providers like *YouTube* and *Yahoo!* Launchcast. For malicious attackers, or a group interested in popularizing their product, there is an incentive in biasing the collaborative filtering technology to their advantage. Such attacks have been refered to as *shilling* attacks, and attackers as *shillers*. Since user profiles of shillers looks very similar to an authetic user, it is a difficult task to correctly identify shilling attacks. Early algorithms exploited signatures of attack profiles and were moderately accurate. In particular, by looking at individual users and mostly ignoring the combined effect of such malicious users, these detection algorithms suffred from low accuracy in detecting shilling profiles. Recent approaches based on SVD (cf. [8]) have proved to be much more accurate, exploiting the *group effect*, i.e. eliminating groups

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

of attackers which seem to work together. However, employing such detection approaches as a preprocessing step are computationally expensive, and essentially not an online process. The next logical step is to build in detection into the recommendation algorithm itself; this work provides a survey of such robust collaborative filtering algorithms proposed in the past and until recently.

1.1 Shilling attacks on Collaborative Filtering

Collaborative Filtering systems are essentially social systems which base their recommendation on the judgment of a large number of people. Like other social systems, they are also vulnerable to manipulation by malicious social elements. As an example, a loosely organized group managed to trick the Amazon recommender into correlate the book *Six Steps to a Spiritual Life* (written by the evangelist Pat Robertson) with a book for gay men[1].

A lot of web-enabled systems provide free access to users via a simple registration process. This can be exploited by attackers to create multiple identities for the *same* system and insert ratings in a manner that affect the robustness of a system or algorithm, as has been studied in recent work [6]. *Shilling attacks* add a few user profiles which need to be identified and protected against. Shilling attacks can be classified into two basic categories: inserting malicious profiles which rate a particular item highly are called *push* attacks, while inserting malicious profiles aimed at downgrading the popularity of an item are called *nuke* attacks. Various attack strategies were then invented; these include [3]:

1. *Random attacks*, where a subset of items is rated randomly around the overall mean vote.
2. *Average attacks*, where a subset of items is rated randomly around the mean vote of every item
3. *Bandwagon attacks*, where a subset of items is rated randomly around the overall mean, and some popular items are rated with the maximum vote.

Random and Bandwagon attacks are low-knowledge attacks requiring information only about some popular items and overall vote statistics. Average attacks require more information and have been shown to be near optimal [7] in impact. They have also been observedly difficult to detect [15].

The strength of shilling attacks is specified using two metrics: *filler size* and *attack size*. Filler size is the set of items which are voted for in the attacker profile, usually measured in %. Attack size refers to the number of shilling profiles inserted into user data. The impact of the attacks is measured by the increase in the number of users to whom an attacked item is recommended. Generally, average attacks are strogner than random or bandwagon attacks.

Metrics for Collaborative Filtering and Shilling The task of evaluating predictions in collaborative filtering is easily described as the measurement of the *deviation* from observed values; accuracy of a CF algorithm is measured over some held-out data from the training dataset. The effect of an attack is measured by the deviation in predited ratings before and after attack profiles have been added. *Prediction Shift* measures the average change in prediction of the attacked item (before and after attack) of a CF algorithm. This metric is also sensitive to the strength of an attack, with stronger attacks causing a larger prediction shift.

Hit Ratio measures the effect of attack profiles on top-k recommendations. Since the end effect of a recommender system is a list of items recommended to a particular user, this metric captures the fraction of users affected by shilling attacks. Let $H_{u,i} = 1$ if an item i is a top-k recommendation to user u, and $H_{u,i} = 0$ otherwise. Hit ratio is a fraction between 0–1; when expressed as a percentage (%), it is defined as follows:

$$\mathcal{H} = \frac{100}{N} \times \sum_{u} \Delta H_{u,i}, \quad \text{s.t. } N = \text{\# users} \tag{4}$$

[1]Story at `http://news.com.com/2100-1023-976435.html`.

Finally, *Precision and recall* are standard IR measures which are also applicable widely; various detection aaproaches report precision and recall for various siyes of attacks. Clearly, high precision is an indication of an accurate method.

2 Robust Collaborative Filtering using Intelligent Neighbor Selection

The earliest algorithms for collaborative filtering relied on neighborhood formation based on user similarity; these are known as k-nearest neighbor (kNN) CF algorithms. These algorithms remain extremely popular due to their simplicity and intuitiveness. This family of algorithms use a weighted reconstruction of the votes of users similar to the current user as a prediction for the rating for a previously unrated item. Various improvements have been made to the basic mechanism of predicting votes using Pearson's correlation, but they mostly comply to the following scheme: assume the user database consists of a set of votes $v_{i,j}$ corresponding to the vote for user i on item j. The predicted vote for an active user for item j, $p_{a,j}$ is a weighted sum of the votes of other users:

$$p_{a,j} = \overline{v}_a + \kappa \sum_{i=1}^{n} w(a,i)(v_{i,j} - \overline{v}_i), \quad \text{where } w(a,i) \text{ is a similarity function} \tag{5}$$

As explained in Section 1.1, attacks in recommender systems can be achieved by the addition of malicious profiles. By special construction, these profiles can be made to look extremely similar to influential users. The impact is that such malicious users can be wrongly identified as *neighbors* for normal users, thus influencing the results of the recommendation algorithm. O'Mahony et al.[12] first discussed *neighbourhood filtering*, where the key idea is to prevent suspicious users from influencing other users. The strategy for selecting useful neighbors takes into account the *reputation* of all users participating in the recommendation process. The reputation measurement strategy is adapted from literature and requires the computation of a reputation score for a particular user who is providing ratings on an item y_i. The first step is calculating the reputation of users who form a neighborhood for other users that have rated y_i; the second step is to filter the neighborhood and remove any possible malicious ratings. The observation here is that if there is indeed an attack, there would be a marked difference between real users and malicious ones, thus leading to two clusters; thus clustering is performed to detect if such a pattern is observed for a particular item. Given that an attack may be trying to promote an item, or demote it, its essential to know the *direction* of the shift for this filtering strategy to work. The approach employed by [12] overcomes this by thresholding the difference in the mean values of the two clusters; if this is above a threshold (evaluated empirically), an attacked is supposed to have occured. To detect which cluster contains the attack users, the one with the lower standard deviation is chosen. The authors provide experimental evidence that this strategy has successful outcome.

The approach suffers from some drawbacks: the detection of the wrong cluster of users can result in filtering-out of genuine users, and thus predicting baised estimates. Further, real life attacker might employ strategies which ensure more deviation in their votes, thus fooling the filtering process. Also, there might be both push and nuke attackers for the same item, which the algorithm is not designed to handle. Thirdly, the running time of the algorithm will be much higher than what is required for large-scale systems. Fourthly, given that neighborhood selection methods are thresholded, it is possible that the number of attackers is so high that the selected neighbours of a user may all be malicious; thus the outlined approach might fail in the face of large and continuous attacks.

3 Robust Collaborative filtering using Existing Approaches: Association Rules and PLSA

3.1 Robustness of Association Rules

A popular approach for frequent-pattern mining is the use of association rules. This technique has been used for market basket analysis, finding interesting patterns of what users buy together, and have been found useful as prediction models too [1]. Applying this technique to user ratings, [13] suggest how a robust method for collaborative filtering could be devised. The application of the above approach has been demonstrated to provide significant robustness to user recommendations. As compared to kNN, k-means clustering and PLSA, the outlined algorithm has a very low *hit-ratio*, meaning that a small fraction of users are recommended an attacked item. For attack sizes below 15%, the hit-ratio of Association rule-based CF is below 0.1, while for k-NN it ranges from 0.8 to 1.0 (meaning all users are recommended an attacked item). For more details, we refer the interested reader to reported numbers from [13](Fig 2).

Clearly, the hit-ratio for association rules is very low, thus implying high robustness. This however comes at the cost of accuracy; Sandvig et al. report that the coverage of this algorithm is below 0.5, which means that the algorithm cannot make any predictions for over half the items in the recommender system. These are typically items which are not very frequently rated. This makes the above approach suitable only in cases where top-n recommendations are required, rather than a complete set of predictions.

3.2 Robust Collaborative filtering using PLSA

Probabilistic Latent Semantic Analysis (PLSA) is a well known approach for text analysis and indexing used to discover hidden relationships between data; it has also been extended to collaborative filtering [5]. PLSA enables the learning of a compact probabilistic model which captures the hidden dependencies amongst users and items. While accuracy has been a well known advantage of PLSA, recent studies have also concluded that PLSA is a very robust CF algorithm, and is highly stable in the face of shilling attacks. [11] indicates that the prediction shift for PLSA is much lower than similarity based approaches; [7] investigated the reasons for PLSA's robustness over many experiments and observed the model to understand the mechanisms. The intuition is that PLSA leads to clusters of users (and items) which are used to compute predictions, rather than directly computing neighbors. However this intuition is challenged by experimental results using a k-means clustering algorithm in the same work. Clearly, shilling profiles deceive clustering algorithms due to their high similarity with normal users.

[7] also outlines a detection approach for shilling attacks exploiting the high stabilty of PLSA. The main idea is that PLSA is a mixture model where membership to a distribution is not constrained; a data point can belong (probabilistically) to many distributions. However some clusters maybe *tighter* thn others: [7] shows that using the average Mahalanobis distance to indetify tight clustes leads o the detection of attack profiles with reasonably high accuracy.

Robust PLSA using shilling detection We suggest using the following strategy to further robustify PLSA: we eliminate the tightest clusters, as identified by the above tightness measure. We now renormalize the probability distribution of the remaining clusters ($p(z|u)$) so that they sum up to 1. One can even attempt to eliminate the suspicious users, randomly perturb the parameters and rerun the last few steps of training. Initial results of this version have maintained the prediction accuracy, while reducing the prediction shift in a statistically significant manner. A more thorough investigation of this idea is under progress.

4 Robust Collaborative Filtering using SVD

SVD stands for Singular Value Decomposition; it is a method of factorizing a matrix into two orthonormal matrices and a diagonal matrix. SVD has become an important linear algebra procedure over the last 2 decades due to its extensive application in Information Retrieval and Data mining. It has been used for Latent Semantic Analysis and Collaborative Filtering with much success. Since SVD is fundamental to the algorithms discussed in this paper, we explore SVD in detail. Further, we briefly explain the Robust Matrix Factorization algorithm described in [9] which is also based on SVD and is robust variant of SVD. Finally, we explain our proposed VarSelect SVD variant as a robust CF solution.

4.1 Singular Value Decomposition (SVD)

SVD is a more general form of Eigen value decomposition (EVD), applicable to rectangular matrices. SVD factorizes a rectangular $n \times m$ matrix $\mathbf{D}$ as $\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$ where $\mathbf{U}, \mathbf{V}$ are unitary normal matrices and $\mathbf{\Sigma}$ is a diagonal matrix of size $\mathrm{rank}(\mathbf{D}) \leq \min(m, n)$, where $\mathrm{rank}(\mathbf{D})$ is the rank of the matrix $\mathbf{D}$. Moreover, the entries on the diagonal of $\mathbf{\Sigma}$ are in non-increasing order such that $\sigma_i \geq \sigma_j$ for all $i < j$. Note that we may chose to set all singular values $\sigma_i = 0$, $i > k$ for some $k \leq \mathrm{rank}(D)$ (say $k = 10$), leading to a low-rank approximation $\mathbf{D}_k$ of the matrix $\mathbf{D}$ ($\mathbf{D}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^{\mathrm{T}}$).

SVD for Collaborative Filtering: Applications of SVD to Collaborative Filtering assume the representation of user-item ratings by such a $n \times m$ matrix $\mathbf{D}$. Typically, user–item matrices are very sparse ($\leq 5\%$ non-zero entries). Initial applications of SVD to CF (c.f. [14]) compensated for sparsity by replacing the missing values by overall mean. This approach, though more successful than previous CF approaches, is highly biased towards the used means. In addition, the lack of sparsity implies a larger computational problem to solve. In the last decade, there has been significant research on SVD for large and sparse matrices e.g. *PROPACK* and *SVDPACK*. However, these approaches do not treat missing values in a principled fashion, either treating them as zeros, or doing mean imputation. A recent algorithm by Gorrell [4] proposed a new approach to computing SVD for virtually unbounded matrices. This method is based on the Generalized Hebbian Algorithm and calculates SVD by iterating through only observed values. The method has been found to be highly accurate for CF and scales easily to the NetFlix dataset with 100 million votes.

4.2 Robust Matrix Factorization

Robust regression problems have been studied in a linear setting where observables Y and inputs X are known and Y is assumed to be noisy. Robust Matrix Factorization (RMF) is algorithm which performs a robust SVD for CF using an alternating fitting scheme [9]. The core idea is the use of bounded cost-functions, which limit the effect of outliers. There is an entire body of work on such bounded functions which are effective against noise; these functions are called Maximum Likelihood estimators or *M-estimators*.

Armed with a robust estimator, we would like the perform the following Matrix factorization: assume we want to find the rank–1 factors $\mathbf{G}, \mathbf{H}$ as for data $\mathbf{D}$. such that

$$\underset{\mathbf{G},\mathbf{H}}{\mathrm{argmin}} \sum_{D_{ij} \neq 0} \rho(D_{ij} - G_i \cdot H_j) \text{ s.t. } \rho(r) = \begin{cases} |r| \leq \gamma & \frac{1}{2\gamma} r^2 \, , \\ |r| > \gamma & |r| - \frac{\gamma}{2} \end{cases}$$

where ρ is an M-estimator called the Huber M-estimator (see [9]. The above optimization can be solved using *Iteratively Re-weighted Least Squares* and is described by [9]. Experiments show that Robust Matrix factorization algorithm performs well in the face of moderate attacks. Clearly, the effect of shilling is low at small attack sizes, as the majority opinion is given more importance. However, once the number of votes by shillers are more than actual users, RMF starts treating the shillers' view as the majority opinion. Mehta et al. also show that RMF is more tolerant to shilling and model deviations than SVD and pLSA: importantly, the prediction accuracy of RMF is higher than any other method; this trend continues even in the face of attacks.However for larger attacks, RMF is clearly inadequate at a robust CF algorithm. In the next section, we show how the RMF and SVD frameworks can be further robustified to yield our desired robust CF algorithm.

4.3 VarSelect SVD for Collaborative Filtering

VarSelect [8] is a variable selection algorithm based on PCA for detecting attack profiles. Shilling profiles tend to be highly correlated, which is a result of the colluded nature of shilling attacks. It is known that for multivariate data, highly correlated variables add very little information, and thus are eliminated by dimensionality reduction methods. VarSelect uses Principal Component Analysis to find which users add least information, and produces a ranking of users in order of utility. Experiments have shown that shillers are found with high precision at the top of these rankings.

VarSelect SVD: We first describe the broad framework for our proposed algorithm. SVD and PCA are closely related since PCA can be achieved via SVD. In essence, PCA seeks to reduce the dimensionality of the data by finding a few orthogonal linear combinations (called the *Principal Components*) of the original variables with the largest variance. A principal component is a linear combination of the variables and there are as many PCs as the number of the original variables. Principally, PCA is equivalent to performing an eigen decomposition of the *covariance* matrix of the original data. Since we want to combine VarSelect with Collaborative Filtering, SVD provides the required framework. The algorithm supports two phases: detection (followed by removal of profiles/votes), and recommendation/model building. For efficiency, it is required that these two phases can share computational steps. Since the detection may not be perfect, no user profiles should be completely deleted and even suspected attackers should be able to receive recommendations. Further, the entire procedure should be unsupervised, i.e. no further input should be required after the detection phase has been performed (e.g. thresholding how many shillers are there in the system).

An important observation we make here is that calculating the covariance matrix is unnecessary; we can compute the SVD of $\mathbf{X}$ to get the loading matrix $\mathbf{U}$ (which contains the Principal components). This saves a significant computational effort as Eigen-decomposition of large covariance matrices is very expensive. Note that PCA requires $\mathbf{X}$ to be zero-mean. This can be expliotied by the VarSelect procedure which has been shown to require only the first 3–5 Principal components suffice to detect attack profiles reliably. Thus a complete SVD is not required: instead, partial eigen-decomposition can be performed. Such routines are available as *svds* and *eigs* in MATLAB and Octave, and use the Arnoldi method.

Finding suspected Attack profiles: PCA can find a set of variables which are highly correlated, a fact exploited in the design of Varselect. Varselect essentially performs Variable Selection using a selection criteria called *Normalized Loading Combination*. There are several other selection procedures discussed in literature ([2, 10] provides a good overview of these criteria). [10] reports that the simplest strategy of averaging loading coefficients (*LC*) performs the best. We choose the following heuristic: normalize the scores so that they sum to 1, and then choose all user with scores below $1/n$ for n users. We observe also that 50% recall is the lowest observed; thus we suggest that for attacks of upto 10%, flagging top-20% should suffice. These selected users are known as *flagged* users.

Computing Recommendations The recommendation model is finally based on SVD as well. In essense, we perform SVD on the data matrix treating flagged users in a special manner. To simplify the prediction model, we absorb the eigenvalues into the left and right factors, as in the GHA-based SVD method. As previously, the data matrix is factorized into a factors $\mathbf{G}$ and $\mathbf{H}$, such that the Frobenius norm of the remainder is minimized:

$$\underset{\mathbf{G},\mathbf{H}}{\operatorname{argmin}} ||\mathbf{D} - \mathbf{GH}||_F, \tag{6}$$

In the context of Collaborative Filtering, note that the left matrix $\mathbf{G}$ is user specific, i.e. each user has a corresponding row encoding their hidden preferences. Similarly, the right matrix $\mathbf{H}$ contains a column for each item. The solution to the above optimization requires iterating through all user votes and performing Hebbian updates. Every vote potential influences both $\mathbf{G}$ and $\mathbf{H}$ during the training phase.

Our modification in presence of *flagged* users is to only update the left vectors and not the right vectors. In other words, the contributions of suspicious users towards the prediction model is zero, while the model can still predict the votes for flagged users. For normal users, we update both left and right vectors as with SVD-GHA. This elegant s solution comes with a very small computational cost of checking if a given user is flagged as suspicious. Note also that the model can be initialized with values learnt from the partial SVD performed for PCA. We note that this results in faster convergence for the already computed dimensions. Additionally, we use a regularization parameter κ (set to 0.01); this step has been found to provide better model fitting and faster convergence.

One issue with the above algorithm is that coverage is low for high number of suspected users r. It is possible that some items are voted on mostly by flagged users, hence enough information may not be known. Therefore, even interested users may not be recommended that item. To improve coverage, we ignore only the extreme votes of the flagged users (i.e. maximum vote 5/5 and minimum 1/5); middle votes can be still used to train the right vectors. This removal significantly weakens potential bandwagon attacks as well as average attacks, since the largest deviations in prediction occur due to extreme votes.

5 Discussion

In the previous sections, we have described several state-of-the-art algorithms for robust collaborative filtering. Clearly, there has been more work in the detection of shilling attackers, rather than modelling shillers as a type of noise. In our opinion, a robust recommender algorithm should have the following characteristics:

1. Highly stable to low number of attack profiles inserted on the attacked item (i.e. low prediction shift)
2. Moderately–Very stable against medium sized attacks ($< 5\%$ attackers)
3. Low average effect on prediction accuracy (mean average error) on non-attacked items.
4. Very high stability to the addition of random noise.
5. No loss of accuracy if no attackers are present in a dataset.
6. Ability to partially trust users, i.e. to be able to generate recommendations for suspicious users.
7. Scalability to handle hundreds of thousands of users with a few thousand possible attackers.
8. Ability to handle multiple simultaneous attacks on different items.
9. Ability to generalize to attack models not encountered in training.
10. Non-requirement for processing the entire dataset again when new profiles are added.
11. Being as parameter-free as possible: the algorithm should be able to figure out various thresholds based on the data without requiring human intervention.

Experimental results published previously show that most algorithms surveyed by us do not conform to a majority of the items in the above checklist. Almost all algorithms we studied have a high degree of stability against medium sized attacks. Some of the algorithms are able to handle low attacks, and detect them reliably. Surprisingly, most of the algorithms meet point 3 and 4 as well: random noise has low effect on both neighbor selection based methods and model-based methods. The performance of these algorithms also does not suffer on non attacked items; the only exception might be in methods where there is an explicit step for detecting an item under attack, and this step gives false positives.

On the issue of running these algorithms on untainted data without any attack profiles, most researchers have not reported the performance of their approaches. However, all algorithms that do some user filtering (e.g. removing suspicious users) do suffer from loss of accuracy. An interesting result in this context has been reported in [9]: in this work, some part of the user ratings was removed. The removed data was a random fraction of the extreme votes (say the lowest, and the highest numerical votes), usually to the tune of 20% of a user's votes; for users less than 10 votes. Various algorithms (e.g. kNN, PLSA, SVD and RMF) were run on the remaining 80% data; the results were that all algorithms gained significantly more stability to shi;lling attacks, to the tune of 20% reduction in prediction shift. The RMF and SVD algorithms had even higher stability; interestingly all algorithms under test did not suffer from a large decrease in predictive accuracy. However, the overall performance of SVD and PLSA based algorithms was much better than kNN.

With respect to partial trust, all algorithms which explicitly remove suspicious user profiles will degrade user experience for suspected process. Since the detection of malicious users profiles is not perfect, false positives will arise from time to time. An ideal algorithm should thus be able to accept that attackers maybe in the dataset, and model this explicitly. The main idea is that all users should be able to receive recommendations, and possibly the user interface should not change at all. The algorithm may internally discard some user data for training, or some ratings and making this transparent to users. Algorithms which do this provide high stability as well: VarSelect SVD and Association Rule mining are examples of this. Intelligent neighbor filtering can do this as well; it has been suggested to weight the contribution of a user's neighbors by the degree of trust (which can be easily expressed as a fraction between 0–1) with good success.

Scalability is a general issue for good algorithm design; efficient algorithms are available for several methods used for recommendation and also for detection. With increasingly cheap hardware and cloud computing becoming common, computational challenges are slowly fading away. However, responsiveness of systems to sudden attacks is crucial. Several approaches seem to run in batches, making it difficult to detect attacks online. To a certain extent, the training part of the recommendation models is also offline, thus restricting the extent of the impact. However, the overall scalability of detection is an important aspect: several approaches have expensive detection algorithms which cannot reuse the previously trained models, or incrementally train their models. Approaches like Varselect SVD which are online in nature are highly scalable as a result.

In the real world, several interest groups would try to boost their products at the same time. Thus one can expect more than one type of attack going on at the same time. Almost all published work on detection however consider only one type of attack at a time for their experimental. It is clearly possible that several of these might actually be effective against multiple attacks; for example the detection approaches proposed by [3] use supervised classification trained on generated examples of several types of attacks. These classifiers are then run on each profile, thus possibly detecting more than one types of attack at a time. Similarly, VarSelect detection and VarSelect SVD were demonstrated to be effective against uncoordinated attacks. While no such results have been discussed by [13], it is likely that this approach will be stable to multiple attacks as there wont be much co-occurrence data for the attacked items to make attack votes significant for the association rule learner (recall that less than 50% of items are actually considered by this approach when creating recommendations.)

The continued menace of email spam shows that given enough incentive, spammers can innovate and create new types of attack campaigns, While various attack models for shilling have been identified, it is ovbious that there are several strategies for new attacks that spammers can come up with. Thus it is imperative for robust CF algorothms to be able to generalize to new types of attacks. Supervised learning methods can clearly fail in this regards; for unsupervised methods to suceed, it is important to understand the intent of the attackers. [7] showed that if the intent is to maximize the prediction shift for a large number of users, the resulting attack model is the average attack. [8] also discusses how the *group effect* is a result of spammers trying to maximize their impact. When obfuscation strategies are employed to add stealth to attack profiles, the strength of attacks goes down. Exploiting the group effect is one mechanism for general attack detection.

One practical aspect we noticed is that several algorithms have too many parameters that need to be manually set. In a real world setting, exploring these parameters manually may not be possible or efficient. It would be better if the algorithm can search its own parameters, either using heuristics, or some principled mechanism (like cross-validation). VarSelect SVD is an example of such an algorithm; while the parameter search is not optimal, there is a broad range of stable values for the parameter r which leads to good stabilty and high maintainability.

References

[1] R. Agrawal, T. Imieliński, and A. Swami, *Mining association rules between sets of items in large databases*, ACM SIGMOD Record **22** (1993), no. 2, 207–216.

[2] Noriah M. Al-Kandari and Ian T. Jolliffe, *Variable selection and interpretation in correlation principal components*, Environmetrics **16** (2005), no. 6, 659–672.

[3] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, *Classification features for attack detection in collaborative recommender systems*, ACM Press New York, NY, USA, 2006, pp. 542–547.

[4] Genevieve Gorrell, *Generalized hebbian algorithm for incremental singular value decomposition in natural language processing.*, EACL, 2006.

[5] Thomas Hofmann, *Latent semantic models for collaborative filtering.*, ACM Trans. Inf. Syst. **22** (2004), no. 1, 89–115.

[6] Shyong K. Lam and John Riedl, *Shilling recommender systems for fun and profit*, WWW '04: Proceedings of the 13th international conference on World Wide Web (New York, NY, USA), ACM Press, 2004, pp. 393–402.

[7] Bhaskar Mehta, *Unsupervised shilling detection for collaborative filtering*, AAAI, 2007, pp. 1402–1407.

[8] Bhaskar Mehta, Thomas Hofmann, and Peter Fankhauser, *Lies and propaganda: detecting spam users in collaborative filtering*, IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces (New York, NY, USA), ACM Press, 2007, pp. 14–21.

[9] Bhaskar Mehta, Thomas Hofmann, and Wolfgang Nejdl, *Robust Collaborative Filtering*, In Proceedings of the 1st ACM Conference on Recommender Systems (Joseph A. Konstan, John Riedl, and Barry Smyth, eds.), ACM Press, October 2007, pp. 49–56.

[10] Bhaskar Mehta and Wolfgang Nejdl, *Attack-resistant Collaborative Filtering*, To appear: In Proceedings of the 31st ACM SIGIR Conference, ACM Press, 2008.

[11] Bamshad Mobasher, Robin D. Burke, and Jeff J. Sandvig, *Model-based collaborative filtering as a defense against profile injection attacks.*, AAAI, 2006.

[12] Michael P. O'Mahony, Neil J. Hurley, and Guenole C. M. Silvestre, *An evaluation of neighbourhood formation on the performance of collaborative filtering*, Artificial Intelligence Review - Special Issue **21** (2004), no. 3-4, 215–228.

[13] J. J. Sandvig, Bamshad Mobasher, and Robin Burke, *Robustness of collaborative recommendation based on association rule mining*, RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems (New York, NY, USA), ACM, 2007, pp. 105–112.

[14] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, *Application of dimensionality reduction in recommender systems–a case study*, 2000.

[15] Sheng Zhang, Yi Ouyang, James Ford, and Fillia Makedon, *Analysis of a low-dimensional linear model under recommendation attacks*, SIGIR, 2006, pp. 517–524.

Vibes: A Platform-Centric Approach to Building Recommender Systems

Biswadeep Nag
Monetisation and Targeting Group
Strategic Data Solutions
Yahoo! Inc
biswadeepnag@yahoo.com

Abstract

Recommender systems have gained a lot of popularity as effective means of drawing repeat business, improving the navigability of web sites and generally in helping users and customers quickly locate items that are likely to be of interest. The rich literature of recommendation algorithms presents both opportunities and challenges. Clearly there are a wide variety of algorithmic tools available, but there are only a few that are suited for application to a broad variety of problem domains and even fewer that can scalably deal with very large data sets. In this paper we describe the architecture of the Vibes platform that is used to power recommendations across a wide range of Yahoo! properties including Shopping, Travel, Autos, Real Estate and Small Business.

The design principles of Vibes stress flexibility, re-usability, repeatability and scalability. The system can be broadly divided into the modeling component ("the brains"), the data processing component ("the torso") and the serving component ("the arms"). Vibes can accommodate a number of techniques including affinity based, attribute similarity based and collaborative filtering based models. The data processing component enables the aggregation of data from users' browse and purchase history logs after any required filtering and joining with other data sources such as categorizer outputs and unitized search terms. We are currently working on moving the modeling and data processing components to the Hadoop grid computing platform to enable Vibes to take advantage of even larger data sets. Finally the serving infrastructure uses REST based web services APIs to provide quick and easy integration with other Yahoo! properties. The whole Vibes platform is designed to make it easy to extend and deploy new recommendation models (in most cases without having to write any custom code). We illustrate this point by using a case study of how Vibes was used to build recommendation systems for Yahoo! Shopping.

1 Introduction

Research into recommendation systems goes back more than a decade with several important classes of algorithms proposed[1]. Lately they have achieved quite a bit of commercial success as well[5, 4], culminating in 2007 with the award of the first Netflix prize[7]. Though recommender systems clearly add value to the commercial proposition and user experience of a web site, they suffer from the drawback of being somewhat fragile and

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

sensitive with regard to the matchup between the data and the algorithms.In other words, recommender systems usually need a fair bit of tweaking to work well in a particular application setting. Added to this are the problems in gathering enough data in a common format about user behavior, item metadata and also in presenting the resulting recommendations in a form that can be easily integrated into target web pages.

The Vibes recommendation platform was built as a scalable and generic solution for Yahoo!'s recommendation needs. The platform has the capability of housing a variety of recommendation models along with all the machinery needed (data collection, data processing, model building, recommendation serving and reporting) to deploy recommendation modules for internal customers.

2 A Typical Vibes Use Case

Unlike the well-known problem of trying to construct a *(user, item)* $\rightarrow$ *rating* function given a set of numerical user ratings, Vibes is usually deployed in the following cases:

1. Provide an inter-item similarity or relatedness function: *(item1, item2)* $\rightarrow$ *similarity* $\in$ [0, 1]. There are a couple of ways to do this as described in Section 4.

2. Provide a user-to-item recommendation function *(user, item)* $\rightarrow$ *recommended* $\in \{0, 1\}$. The output of this function is a boolean which decides whether to suggest *item* to *user* or not.

As an example of case 1, take a look at a product detail page in Yahoo! Shopping, for example an Apple iPod (`http://shopping.yahoo.com/p:Apple%20iPod%20touch%208GB%20MP3%20Player:1994935518`). Vibes item-to-item recommendations are shown in the section titled: **Yahoo! Shoppers Who Viewed This Item Also Viewed:**. All the data we need to supply these recommendations can be obtained from a web log of all the product pages viewed by visitors to Y! Shopping. If enough users visit a common set of items within, say a 90-day time period, those items can be the basis of generating recommendation rules.

Providing recommendations for case 2 is harder, primarily because of the sparsity of the data. Getting users to explicitly rate an item can present a barrier, but we can collect implicit binary rating data either from users' browse or buy history. It is fairly straight-forward to generate recommendations for users who *have* a history in Y! Shopping, but it is more challenging to cater to users who land-up directly from another search engine for example. The model then has to augmented with other behavioral data from the Yahoo! network (such as a user's search history) if available. This of course may raise some privacy issues in case we tap into a user's declared age, gender or any other personally identifiably information. Vibes customers may decide on a case-by-case basis to explicitly ask permission from users before showing recommendations that rely on their behavioral history.

Various Vibes customers have reported substantial benefits from adding a recommendation capability to their web site. For example Yahoo! Shopping has recorded a 16% increase in revenue after deploying Vibes. Similarly Yahoo! Small Business has measured a 30% increase in per-order revenue over manually generated cross-sell rules. The big advantage in Yahoo! is that it is possible to leverage users' network-wide behavior, including terms typed into general search to segment users into clusters and further personalize the recommendations. Currently we are in the process of developing such a model.

3 Platform Requirements

Yahoo! is in a unique position as one of the most popular destinations on the internet. Not only does Yahoo! have one of the largest user bases, a number of Yahoo! properties (such as Autos, Games, Shopping, Sports etc.) are ranked in the top 2 web sites of their respective categories. Yahoo! also has a major advantage in the fact that its users spend a significant amount of time on the web site, accumulating a large number of views and clicks, which translate into significant user behavioral history. These User Link Tracking (ULT) data ultimately find its way into data warehouses from which vertical-specific aggregated data can be queried. To achieve deployment of scale of the Vibes recommendation system across a broad range of Yahoo! properties, we started with the following set of requirements:

Loose coupling The recommender system stands apart from its customers. Changes in the recommendation platform, its algorithms and its infrastructure should be transparent to the consumers of the recommendations. This means that Vibes would run on a different set of systems running possibly different operating systems and language runtimes. A standard way of doing this is by using Web Services, in particular using REST principles.

Configurability It should be possible to easily tailor recommendations for each customer in terms of model parameters, data sources, and APIs. This could be done via a level of meta-programming where each instance of Vibes has a set of configuration parameters in the form of XML files that specify the methods of data generation, model building and the signatures of the RESTful web service.

Extensibility The Vibes platform should be able to easily accommodate new modeling methodologies and particular requirements for each customer deployment. The modeling block should be able to incorporate new code (written in any programming language) while the customer interface should have logic to activate business rules to merge, filter, compare and combine the recommendation results as required.

Easy integration The customer should have to take minimum effort both to provide data that feeds into the modeling engine as well as to consume the recommendation outputs. The data input could happen through standardized channels for instrumenting view and click events that flow into the warehouse. The recommendation output would be served through a web service that will be easy to parse and consume.

Quick deployment The platform needs to minimize the man power and incremental effort required for each new deployment. This could be done by having a standard configuration template that could be tailored to each customer's requirement by making localized changes for the input data source and output data format. No new code should have to be written in the common case, thereby alleviating the need for a long QA test cycle. The scheduling of model refreshes should happen automatically.

Quality checks We should anticipate operational issues such as missing or truncated input data, or perhaps changes in data distribution. Models should be evaluated based on metrics such as *precision, recall* and *coverage*. Every time a new model is built, it should be compared with a set of historical models and pushed into production only if the deviation is within tolerance limits.

Scalability The recommendation serving infrastructure hosting the web service should scale horizontally. That means that the total number of requests that can be served per second should be linearly proportional to the number of serving systems. In addition, 99.8% of responses need to be within 20 ms. On the model building front, the platform also needs to exploit data parallelism and should be able to take advantage of multi-CPU machines and grid clusters.

Reporting A recommender system is only as good as the visibility it provides into the effectiveness of its recommendations. Instrumentation needs to be embedded into the recommendations so that we can track the number of views and clicks made by users. The effectiveness of a recommendation module is measured using the click-through-rate (CTR) on the recommended items.

The architecture of Vibes takes into account the above requirements and is graphically shown in Fig.1. It is useful to think of the system as having three main components: the modeling engines, the data gathering and processing framework and the recommendation serving infrastructure. The data processing and modeling

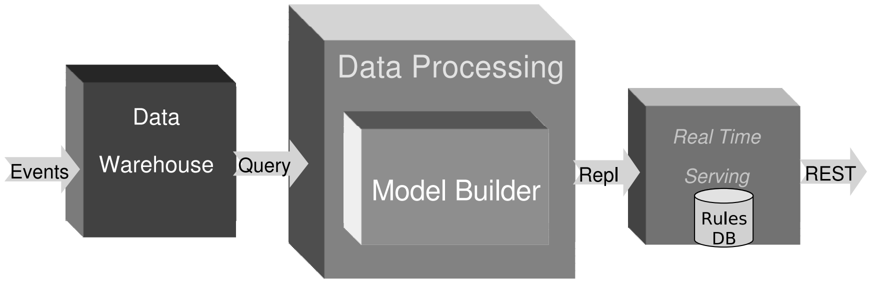


Figure 1: Vibes Platform Architecture

code runs in a central location in close proximity with the data warehouse while the model output is replicated to various data centers where the front-end systems serve it at real-time to co-located customer web servers. Before we delve into the details of each component it is worth noting that the components themselves are loosely coupled together, making it relatively easy to swap in new implementations.

4 Data Modeling

4.1 The Vibes Affinity Engine

The workhorse of Vibes data modeling is the **Affinity Engine** that is used to build item-to-item affinity models. Items could be almost anything in the Yahoo! universe, such as, product pages, auto makes, real estate listings, travel destinations, RSS feeds, computer games, search keywords and so on. User interaction with items is discretized into groups. A *group* (also sometimes called a session, transaction or market basket) is a set of events relating items. For example, a group could be page views by a single user, all the searches in a session, all the RSS subscriptions for a userid, products bought in a single checkout or pretty much any association of items. Item-to-item affinity is nothing but a set of association rules[3]. An association rule $A \rightarrow B$ relating two items A and B implies that we will recommend B when given A as input. To qualify as a rule, the pair (A, B) must co-occur frequently in a group i.e. they must pass certain thresholds of *support* and *confidence*. Support or minimum pair count is the least number of co-occurences of A and B for them to generate a rule. The choice of the support threshold depends on the characteristics of the data, particularly the ratio of the number of items to the number of groups. A higher item to group ratio (i.e. sparser data) may require lowering the support threshold to ensure sufficient number of rules (and item coverage). Typically we choose support thresholds of 9 or above to minimize noise. In our implementation, confidence or the affinity value is not a threshold, but instead an ordering metric. Confidence for a rule $A \rightarrow B$ is the conditional probability: $P(B|A) = P(A \cap B)/P(A)$. We generate all item pairs satisfying the support threshold and then for item A find the top n items X which have the highest $P(X|A)$. This gives rise to n recommendation rules (n being a config parameter for a particular deployment).

The most computationally intensive tasks involve finding the item pairs that have at least the minimum pair-count. At the scale of Yahoo!'s data (3 million items, 100 million groups), this is reasonably hard to do. This is where classic algorithms like Apriori[3] fail to scale. To make the problem tractable, we only consider binary rules, i.e., we only count item *pair* frequency (experiments have shown that the gain from having rules involving more than 2 items is not signficant). At a high level, this involves creating aggregate hash tables in memory mapping item-pairs to current counts and then flushing these tables to disk when memory overflows. Finally a second pass is made to merge and sum up all the item pair counts. There are several optimizations geared toward large data set processing in the actual implementation. Chief among these are encoding all the itemid strings to integers (as well as all itemid pairs to integers). Processing and comparing strings is the biggest consumer of cpu time and we have found this integer encoding method to be the biggest contributor to scalability. There are also optimizations involved in dropping items that fall below the occurrence threshold and dropping item pairs which do not make the cut of the top n affinity rules.

The affinity engine uses the well-known technique of association rule mining. Though this technology is quite mature, we have found it to be remarkably effective in real-life situations. Given enough data (i.e. a low item to group ratio) it is remarkably effective in tracking user behavior and often beats other more sophisticated models in predictive performance. Additionally it can consume large data sets with ease, even when used on a single system. In the near future, we plan to signficantly enhance the data processing capability of Vibes by deploying a version of the affinity engine that runs on a Hadoop[8] grid cluster where it would be able to utilize hundreds of compute nodes.

4.2 The Vibes Attribute Similarity Engine

Affinity based modeling seems to have only two weaknesses. It may not be able to produce an accurate model under these conditions:

- There is not enough data, i.e. the number of items to number of groups is high. This can happen if a particular web store-front does not have enough visitors or transactions. This reduces the probability of two items co-occurring enough number of times to overcome the support threshold.
- There are new items that have not accumulated enough history (in views, buys etc.). This is also known as the *cold start* problem. In general, it is very difficult to apply a behavioral model to such items.

One particular case where these conditions occur is for Yahoo! Real Estate and Autos which have high volume house or car listings. These listings have a finite lifetime and may not accumulate enough views within a 30-90 day window to make affinity based recommendations. In these cases, we plan to leverage structured metadata that is available with the listing. The idea is somewhat similar to the approach proposed in [6], but we use structured metadata instead of textual descriptions. In the real estate case, these would include attributes such as the price, zip, number of bedrooms and number of bathrooms for a house. Using just these metadata we can calculate a similarity score between any two listings based on a linear combination of attribute distances. The exact formulation of the similarity score between two items i and j each having n attributes $(a_{i1}, a_{i2}, \dots a_{in})$ and $(a_{j1}, a_{j2}, \dots a_{jn})$ respectively is:

$$Similarity(i, j) = 1 - \sum_{k=1}^{n} w_k * \delta_k(a_{ik}, a_{jk}) \tag{7}$$

where δ_k is the distance function and w_k is the weight applied to the kth attribute. It is possible to choose from a wide variety of distance functions such as Euclidean, Manhattan, Hamming distance etc. The distance function chosen is normalized to produce an output in the range $[0, 1]$.

Now comes the problem of finding weights. Our current approach is to learn these attribute weights from the data points where affinity model data exists. Based on user click behavior, we can model the similarity function to approximate the affinity (or confidence) of the recommendation $i \rightarrow j$. Given *Affinity(i, j)* and the attribute metadata for items i and j we can construct a set of linear equations of the form given in Eq.8.

$$Affinity(i, j) \simeq 1 - \sum_{k=1}^{n} w_k * \delta_k(a_{ik}, a_{jk}) \tag{8}$$

Since $\delta_k(a_{ik}, a_{jk})$ can be easily calculated, it is relatively straight-forward to do a linear regression to derive the weights w_k. A similar idea applies to constructing distance functions for categorical attributes. We intend to leverage user behavior to calculate affinities between categorical attribute labels and thereby deduce the relative distances between those attribute values.

4.3 User-to-Item Recommendations

This is usually considered the classic collaborative filtering (CF) problem, but the primary problem here is one of scale. Yahoo! has of the order of 500M users, and if we consider 1M items, we are faced with a 500 trillion cell matrix. Most of the algorithms described in the literature tend to break down when faced with data sets of this size. Furthermore, the data tend to be rather sparse, partly because of the rank of the user-item matrix and partly also because of practical problems of cookie churn which tends to inflate the number of users presented to the CF algorithm. The goal of the Vibes platform is to identify a small set of core algorithms that can be applied to a wide domain of user-to-item recommendation problems. We are currently evaluating several promising algorithms like those in [2].

In the meantime, it is possible to simplify the problem somewhat by noting that most of our use cases do not require a numerical rating prediction. We simply need to output a binary prediction of whether to suggest an item to a particular user or not. Of course we want to optimize the click response to our recommendations as well. One simple way of doing this would be to look at the click history of a user in a given vertical (say Yahoo! Shopping), i.e. for a user u, we have a set of items *Viewed(u)*. Then it is possible to expand this set using affinity based item-to-item recommendations:

$$Recommended(u) = \bigcup_{\forall i \in Viewed(u)} AffinityRecos(i)$$

where *AffinityRecos(i)* refers to the set of items recommended for item i by an affinity based model. When faced with new items with no behavioral data, it is possible to bring in a content dimension to the above by substituting (or augmenting) affinity based recommendations with attribute similarity based recommendations.

5 Model Building

The power of the Vibes framework stems not from the sophistication of the modeling engines but rather the ease and rapidity with which they can be deployed in large number of divergent use cases. The model building framework (the Data Processing block in Fig.1) has the job of aggregating, filtering and processing data to feed into the modeling engine. The following are some of the main operators that have been implemented in the Vibes Model Builder.

query Queries the data warehouse using an SQL like query language to extract user behavioral data.

mergesort Takes a set of compressed files as input, merges them and sorts them on the grouping key.

model Encapsulates the modeling engine and is further specialized into affinity, attribute similarity etc.

script Vehicle for plugging in ad-hoc scripts or executables written in any language for data processing.

evaluate Calculates metrics such as *coverage, precision, recall* for each model generated.

compare Compares current model with a set of historical models.

dbload Loads model result into serving database.

dbreplicate Replicates the model database across data centers.

In addition to specific operator properties, each operator has a set of common attributes (derived from a parent operator class in an inheritance hierarchy) such as name, scheduling frequency, data duration, input data source, output file and output schema. The individual operators are orchestrated into a workflow (specified as an

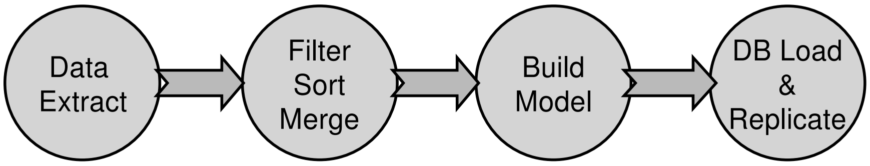


Figure 2: Vibes Data Modeling Pipeline

XML file) for each customer deployment. Fig.2 shows an example linear workflow, but in general this can be a dependency graph in the form of a DAG.

Model refreshes happen in an automated fashion depending on the desired refresh frequency (which in turn depends on the velocity of the data). Checks are built into each stage of the modeling pipeline so that downstream processing is halted in the event of any failure. For example if a signficant difference in the input data causes a substantial change in the number of rules, or the precision and recall of a model, then the model comparison operator fails and stops the dbload and dbreplicate operator preventing the deployment of the (possibly) faulty model into production. The model outputs are currently stored in a MySQL database table having the following schema: `recos: (src_item, dest_item, score)`. The `score` column refers to the degree of relatedness of the source item and the destination item and can either be the affinity or the attribute similarity score. The `recos` table is indexed on the `src_item` column, making it very easy to look up the set of recommended items using the SQL query: `select dest_item from recos where src_item='given item' order by score desc`.

A recommendation platform is only as good as the visibility it provides into its performance. From the outset, Vibes has ensured that appropriate metadata is sent along with the model output so that customers can record the number of views and clicks made by users on the recommendations. These data flow into the data warehouse from which we can report daily performance metrics such as views, clicks, click-through-rate and the number of unique end users targeted. This allows us to have ongoing quality checks tracking model performance.

6 Recommendation Serving

As we have mentioned before, the Vibes serving infrastructure (which is located in various data centers) is loosely coupled with the model building apparatus that is co-located with the central data warehouse. The glue is provided by MySQL replication. After the models are built, they are loaded into a master database that is then replicated asynchronously to slave serving databases in data centers across the country. The slave databases are read-only (except for batch model updates via replication) and this allows us to use the MyISAM storage engine optimized for batch rather than OLTP. The loose coupling betwen the data processing backend and the model serving frontend has several advantages namely better online performance and failure isolation. The Vibes service has to be up and serving recommendations 24 X 7 because of the nature of the traffic coming to our customers, the Yahoo! properties. There is no downtime due to model refreshes because the replication pushes the new model data into a staging area and then switches over in less than a second. Equally importantly, a failure in the model building process would stop the model refresh and replication, but the Vibes frontend would still satisfy recommendation requests using the older model data in the database. Fig.3 shows the detailed architecture of the Vibes recommendation web service.

The Vibes front-end is designed to scale horizontally and to serve 99.8% of requests within 20 ms. This is required to have an acceptable end-user experience. When a user browses a product page, say: `http://shopping.yahoo.com/p:Apple%20iPod%20touch%208GB%20MP3%20Player:1994935518` a call is made from the Y! Shopping web server to the Vibes web service in the form: `http://shopping.vibes.yahoo.com/vibes?method=shopping.recos.getVibes&itemid=1994935518`. After

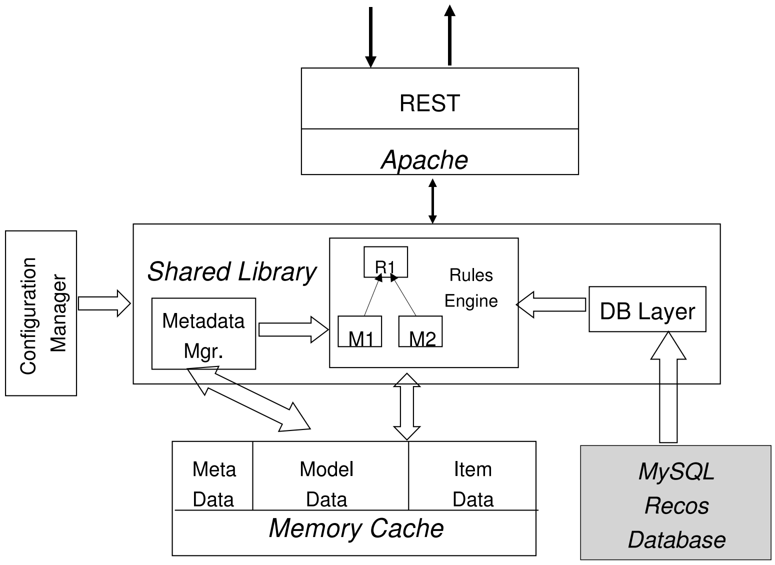


Figure 3: Vibes Serving Architecture

Vibes returns the recommendations in the form of an XML document, the Y! Shopping server parses the response and renders the HTML and graphics for the recommended items before sending them to user's browser. The browser has to load the complete page in 1-2s. One of the critical components in achieving this low latency is to use a large (currently 1GB) in memory cache (Fig.3) that caches responses from the database. The cache contains both positive and negative (no recommendations) results and uses technology similar to the popular *memcached* even though it is not partitioned across machines, (and so does not incur the penalty of an extra hop). It is possible to further optimize this process by using AJAX to make asynchronous calls to Vibes such that the main part of the page can load first while the recommendations are rendered in the background.

In the near future, we will be deploying Vibes also on the home pages of Yahoo! properties where recommendations could be provided without an item context. In that situation we would use information about a user's interests as identified by the Yahoo! cookie sent by the browser. This could be used to *score* the user, i.e. retrieve behavioral and demographic information about the user which are then used to place the user into one or more clusters (with certain probabilities). Finally these user scores are used to suggest items that are considered to be the most popular among members of the selected clusters. This dynamic user scoring for personalized recommendations is also going to be the responsibility of the Vibes front-end.

Vibes uses the Apache web server as the base for its web service. All the logic to parse the request, look up the recommendations from the serving database and formulate the XML response is compiled into a shared library that is loaded by Apache. In addition, there is the flexibility to tailor the recommendations in real-time using a series of rules that can include or exclude items or combine the results of a number of models. For example, in the cases where we have both affinity based and attribute similarity based models, it is possible to make run-time decisions about which kind of recommendations to serve based on either item coverage or the relative magnitudes of the affinity and similarity scores. In the near future we plan to implement a model testing capability that can be configured to partition the requests among a set of available models whose performance would then be evaluated by the Vibes reporting mechanism.

As is required for a system in production, the Vibes serving architecture has several layers of fault tolerance built-in. We are located in several geographically distributed data centers and within each there is redundancy in the web serving layer as well as in the database tier to enable tolerance of single system failures. Just as for the data processing backend, the Vibes front-end meticulously preserves a series of statistics related to service up-time, system load, number of requests coming in, number of recommendations served and the number of cases where there were no recommendations. These data flow into a reporting portal that graphically displays these metrics over time as well as sends alerts to operations teams if they deviate over tolerance bounds.

7 Future Direction and Conclusion

There are significant enhancements planned for each component of Vibes. For the modeling component, we would like to push deeper into personalized recommendations in a way that will scale to millions of items and hundreds of millions of users. These models are going to be extremely computationally intensive to build, so we have started moving our backend infrastructure to a Hadoop[8] grid environment. We have already done experiments for generating affinity rules by mining search query logs which supply this data at a scale that can only be processed on the grid. Having larger models will also put greater stress on the serving infrastructure, which probably will have to handle models that are at least 100X larger - requiring a different caching solution and possibly a different storage architecture.

Longer term, we would like to expose the power of the Vibes recommendation platform to a wider audience. The algorithms and infrastructure should be generic enough to be able to tackle data from a wide variety of domains and satisfy a large number of use cases. Moving towards a self-service capability that allows the customers themselves to configure and deploy the recommender system by defining API parameters and pointing the system to the data source would be a big plus. Our vision for Vibes is that it would be deployed by a simple drag-and-drop into a web-enabled application framework. That is when taking a platform-centric approach to building recommender systems would really pay off.

References

[1] Adomavicius G. and Tuzhilin A., *Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions*, IEEE Trans.on Knowledge and Data Engg., Vol 17, No 6, 2005.

[2] Agarwal D. and Merugu S., *Predictive discrete latent factor models for large scale dyadic data*, Knowledge Discovery and Data Mining (KDD) 2007.

[3] Agrawal R. and Srikant R., *Fast algorithms for Mining Association Rules in Large Databases*, VLDB 1994.

[4] Das A., Datar M, Garg, A, Rajaram S., *Google News Personalization: Scalable Online Collaborative Filtering* World Wide Web Conference (WWW) 2007.

[5] Linden G, Smith B., York, J., *Amazon.com Recommendations: Item-to-Item Collaborative Filtering*, IEEE Internet Computing, Vol 7, No 1, 2003.

[6] Melville P., Mooney R., Nagarajan R., *Content-Boosted Collaborative Filtering for Improved Recommendations*, Proc.of 18th Conf. on Artificial Intelligence, AAAI 2002, Edmonton Canada, July 2002.

[7] The Netflix Prize, `http://www.netflixprize.com`.

[8] Hadoop, `http://hadoop.apache.org`.

User Experiences and Impressions of Recommenders in Complex Information Environments

Juha Leino and Kari-Jouko Räihä
University of Tampere, Department of Computer Sciences
Tampere Unit for Computer-Human Interaction
kjr@cs.uta.fi

Abstract

We studied how actual users find items of interest in today's complex, recommender-rich information environments, what role recommenders play in it, and if recommenders increase perceived social presence. We used applied ethnography, on-location observation and interviewing, and Amazon as the environment to get an accurate picture of user activity. We found that users are increasingly relying on recommenders in finding items of interest. Since they have developed strategies to combine keyword searching with recommenders for discovery, recommenders should not be developed in isolation of the whole because users do not use them in isolation. In addition, while some users feel that recommenders add to the sense of social presence, others feel that they are not enough to create a sense of others being present.

1 Introduction

> *And I think that this feature is good, this 'those who have bought this book have also bought that book.' I have found some books by that. For instance, I think that when I was looking for a book on these mercenaries, it gave me a good list. I found [by keyword searching] something that had something to do with it, and then I could search through it, and it works very quickly, because I can do kind of a cross-search, search for books on mercenaries. Then when I read about some of them, some that I might be interested in, and then I take one and then I go to this 'who bought this also read these,' and it shows books with similar themes.* – Participant 4

Recommender systems have become omnipresent in e-commerce. As Brent Smith, Amazon's director of personalization, says: "Personalized recommendations are at the heart of why online shopping offers so much promise" [10]. Already today, recommenders are affecting where we go for holidays, what newspaper articles we read, and what movies we watch, and there seems to be no limits to how they will be used in future.

While searching is seen as a way to help us find items that we know, recommenders are seen as means to discovery [5]. Combining the two is even touted as a "next Google" concept, and punters see in their mind's eye a future where such applications know more about us than we do ourselves [10].

Consequently, recommender systems have been frantically researched in both academia and industry. At the beginning, the research focused heavily on the algorithms and different accuracy metrics for them [6] while

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

the user issues were not widely researched, perhaps to the detriment of the whole field [9]. Today, however, various user-related aspects are receiving increased attention. Our study is, as far as we know, the first that takes a detailed look on the combined use of user recommendations and searching by keywords.

One interesting user aspect is that of social presence. Research has focused on social presence as a precedent to trust and loyalty in e-commerce [1, 3] since lack of trust is seen as one of the greatest hindrances to the growth potential of e-commerce [3, 4]. E-loyalty, in turn, has been called a "competitive necessity" in e-commerce and shown to boost sales [11].

Nevertheless, the concept of social presence has not been well defined in the literature. In a study of Kalas [14], a social navigation system for food recipes, social presence was defined as a perception of "not being alone in the space", and we use this definition here. Social texture, features that indicate synchronous or asynchronous presence of others in the environment, was seen to provide the basis for social presence [14]. Furthermore, Kumar and Benbasat [8] showed that recommender systems, including customer reviews, increase the perception of social presence in addition to increasing the perception of usefulness.

Much of the research effort still focuses on different aspects of recommenders instead of complete recommender systems, and even less attention is paid to recommenders as parts of complex information environments where different ways of finding items compete for user attention, a scarce resource to begin with. Such e-commerce sites as Amazon offer various ways to browse the items, and recommenders are only one of many. As it is, we know little about how users actually use such complex environments. Research that deals both with search and social aspects has focused on social navigation (e.g., [2]), not on user recommendations. In fact, the only research done within the researcher community on the integral wholes that we know of is that of Kalas, "one of the most complete social navigation systems ever built" [12].

There are various reasons for this lack of research into the complex information systems as integral wholes. First, studying complex information environments is challenging for a number of reasons independent of the methodology used [8]. In addition, the environments need to be used for prolonged periods by numerous users to start to deliver the goods [14]. Even in the Kalas study, where 302 active users used the system for six months, the recommender system never actually started to work properly due to the sparsity problem [14]. Building complex information environments is an onerous task to begin with [8], and the difficulty of finding large numbers of motivated users is enough to discourage even the most intrepid researcher. Arranging financing is another complicating factor in such prolonged studies. Finally, while the commercial systems tend to be superior to the ones built by researchers, their data is not available to researchers [8].

Consequently, we have little knowledge of how users actually use complex information environments and what role recommenders have in finding items of interest. Where Kalas provided quantitative insight into what users did in the complex information environment in question, we were interested in seeing the actual use unfold and peak into the motivations and perceptions behind the actions. In addition, we wanted to see if the environment was indeed perceived as inherently social and if the perceived social presence affected the behavior in the environment. Thus, we used applied ethnography, in our case a combination of on-location observation with verbal protocol and interviewing, to study how six Finns found items of interest in Amazon, the world's biggest online retailer.

We chose Amazon to represent complex information environments because it has consistently been an early adopter and innovator of new e-commerce approaches [7, 8]. In particular, Amazon has used a wide array of recommender approaches for years. Moreover, Amazon has a very rich social texture.

Although our method limited us to six participants, thus limiting us to observing trends at general level rather than at subgroup level, our participants were genuine users with genuine motivation, and thus enabled us to see clear trends and interesting examples of actual user behavior in a complex information environment.

We found that while recommenders play an important role in finding items of interest and that users find them reliable, searching by keywords is not threatened by them. Recommendations are used both opportunistically and strategically. Opportunistic use refers to users using recommenders unpremeditatedly when seeing them while strategic use refers to recommenders being used intentionally even to the point of intentionally manipulating what gets recommended. In fact, while punters are talking of approaches combining recommenders with searches as the next Google, users are already combining searching and recommenders by seeding recommendations with searches.

On the perceived social presence, our participants divided into two distinct groups: Half felt that the social texture in Amazon did result in social presence while the other half felt that it did not. Nevertheless, we did see evidence that the actions of others made visible in the social texture affected the behavior of at least some users.

The rest of the paper is organized as follows. After discussing our method and participants, we take a look at how items of interest were found and what role recommenders played in it. Then we look at how the recommender use can be described as opportunistic or strategic, and the implications of this. Finally, we discuss the perception of social presence and how it affected the use.

2 Method and participants

2.1 Participants

The participants were six Finnish males, aged between 33 and 44. We refer here to Participant 1 as P1, Participant 2 as P2, etc. All were in working life, had at least polytechnic-level education, and were experienced Internet users.

A book purchase from Amazon was required for recruitment to ascertain that all were actual users of Amazon. On average, participants had purchased 10 books (between 2 and 30) from Amazon prior to the study. For the participants, the main reason for using Amazon was the availability of books. Four participants had also bought other items from Amazon. Participants had used Amazon for 4.5 years on average. Thus, our participants were actual users of Amazon. In contrast, many studies have used students acting as consumers [4, 13], which raises questions of external validity [3].

However, while our participants were all "genuine," they were all male, and men and women are known to have at least some differences as e-commerce customers [1]. Additionally, the number of our participants was low, and they ended up using Amazon only for buying non-fiction books during our observation sessions. Finally, cultural issues prevent us from generalizing the results too widely.

2.2 Method

We used applied ethnography, in this case a combination of observation with verbal protocol and interviewing at the participants' homes with them using their own computers, as our method to get an authentic view of real use.

While observation gives a picture of what users do and how they do it, it does not reveal their motivations and other reasons behind their actions. Verbal protocol, in spite of its limitations and potentially behavior-altering influence, is still our only way to get inside the participant's head during the action without the clouding of reflection that interviewing introduces. Thus, verbal protocol provides on-line insight while interviewing provides reflective insight into the actions of the participant. Combining interviewing with observation also avoids the say-do problem, the human tendency to describe what they do differently from what they actually do.

The observation-interview sessions, one per participant, lasted 2–3 hours each. The participants were given four tasks and asked questions before, during, and after each task. Care was taken not to direct participants' attention with the questions: during the tasks only some clarifying questions were asked when a user stayed on a page for a long time without visible or verbalised actions. After the tasks, a semi-structured interview was conducted. Finally, the participants filled in an online demographic questionnaire.

The sessions were videotaped with the video camera pointed at the computer screen to provide the context for verbal protocol. The video camera also recorded the interviews.

The tasks were given to the participants on a web site made for the study with each task on its own page. However, only the first two tasks are within the scope of this paper.

Task 1: "Buy a book (or books) from Amazon. Do not buy a book that you have already decided to buy. Instead, you should find a book that you have not decided to buy beforehand." On average, the participants used 23 minutes on Task 1.

Each participant was given 15 euros towards purchasing the book(s) in Task 1 to make sure that they selected a book they really wanted. This is significant because research suggests that people use "affect or other simple heuristics to guide their decisions" when the task does not involve them, the task is trivial, or they are not motivated, while in high-involvement situations, when people have something to lose or are simply deeply engaged, people use "cognitive analytical processing" [13]. In our study, the users were not paid *per se* or given a chance to win something by participating, which might have motivated them to take part in the study but not engage them any deeper in the tasks. Instead, what they received depended on how they did the task, involving them deeper in the task itself.

The participants were instructed to use the Amazon site that they most typically used. Three participants used .co.uk, two used .com, and one used both .com and .co.uk. The choice of site was given to preserve normal conditions although there are differences between the two interfaces.

Task 2: "You have bought a good digital camera and now you would like to buy a photography guide from Amazon. Which one of the books on the list would you buy?" The task page provided a link to the list page that was constructed to look like a list page in Amazon.co.uk. The page included books with high star rating, low star rating, no star rating, and one book with Search inside function available. A mock-up page was used to make sure that all these different conditions were present. The links on the page led to actual item pages in the .co.uk site. On average, 11 minutes were used on Task 2.

All sessions were transcribed and then contrasted for analyzing. No analysis software was used. Because the study method produced qualitative data, the goal of the analysis was to describe the observed behavior and to find patterns.

3 Results and discussion

3.1 The role of recommenders in finding items of interest

> *I don't know where this Kerouac thing came from. It came some weird route. The system kind of drove me to it. I kept getting closer and closer all the time and when I eventually was about to take the other book that was more at general level, it pushed this Kerouac's memoirs at me [laughs] and I couldn't resist it or ignore it. If I were in a bookstore, how the hell would I end up with something by Kerouac? I'd be there looking at some painting books and the link between Kerouac and tankha-paintings would be hard to draw, it just wouldn't happen, and in that sense I'd be there, probably looking at some impressionistic painting guides [laughs], and think that maybe this is not quite what I wanted.* – P4

In Task 1, seven books were bought. Three were found by recommendations and four by keyword searches. P3 used directly personalized recommendations and found his book. P2 also started with personalized recommendations but he already owned the only interesting book in them and continued with keyword search. Three participants, P1, P4, and P5, used keyword searches directly while P6 started with categories but moved to keyword searching after failing to locate any interesting books.

P1 found a book with keyword searches but after putting it into the shopping cart, he saw an impulse item recommendation for another book and went to its item page. When seeing an offer to buy the book in the basket with the new book ("Perfect Partner" recommendation), he decided to buy both for about £40 even though he had earlier on mentioned wanting to get a book on the subject for about £10. P5 was also interested in the

"Perfect Partner" recommendation, but he already had the recommended book. P4 found the book to buy from "Customers who bought this item also bought" list after a few searches.

In practice, all participants used recommendations in the item-finding process. Table2 summarizes the recommender use in Task 1. Interestingly, all three books found by recommendations could be characterized as serendipitous. Participants found items that they would not have found otherwise and that were exactly what they wanted. Thus, the recommenders were clearly providing discovery.

Task 1	P1	P2	P3	P4	P5	P6	Total
Bought a book offered by algorithmic recommender	•		•	•			3
Bought a book found by keyword searching	•	•			•	•	4
Used keyword search	•	•		•	•	•	5
Used categories for searching						•	1
Used "Perfect Partner"	•			•	•		3
Used personalized recommendations (at the beginning)		•	•				2
Used "Customers who bought/viewed this item..."	•			•		•	3
Used "Explore similar items"						•	1

Table 2: Recommendation use in Task 1 by the participants.

Furthermore, two participants used personalized recommendations as the starting point and several comments by participants showed that they were actively looking for recommendations. Consequently, recommenders have become an integral part of complex information environments in users' minds, and play a significant role in their item-finding strategies. Consequently, our findings are in line with the studies that suggest that recommender systems are necessary and useful in finding items in the era of information overload.

Meanwhile, keyword searching, once the standard tool for item-finding, has clearly given some ground to recommenders. However, it is still a natural starting point when the topic is known but not much more.

The major problem with searching is naturally to come up with correct keywords. For instance, P4 used as keywords "phone tapping government." That search produces 22 results in Amazon.com while a search with "wiretapping government" produces 215 results (March 18, 2008). However, P4 did not come up with "wiretapping," and so he concluded wrongly: "*Perhaps a book with that stuff in the way that I want it has not been written.*"

Finding the right keywords can be even further complicated when the system is not in the native language of the user, as with our participants. For instance, names—many participants used author's name as keyword—and concepts with foreign words caused spelling problems. Some participants had strategies to deal with such situations. For instance, when P6 failed to remember the spelling of an author's name, he instead searched for a book by the author, as he knew how to spell the words in the book's title. Finding a book by the author helped him to access relevant recommendations.

Interestingly, some participants simply searched for a book they knew on a topic to access similar books through recommendations, thus seeding the recommendations with searches. Thus, recommenders can complement searches and inspire new searches, just like searches can be used to seed recommendations.

In the light of our study, searching and recommenders do not compete with each other but complement each other in many ways. However, it is Amazon's ability to make recommendations based on just one item viewed that makes this possible. If recommendations were simply based on previous purchases and did not react to the item at hand, it would be impossible to integrate them into the item-finding process the way the participants did. Thus, to allow recommenders and searches to complement each other, recommenders have to be responsive to the current task context.

Our findings are in line with Hangartner [5] who concludes that searching is not disappearing because of recommenders but can be enhanced with recommenders, and that recommender industry will continue to grow in sophistication and importance.

3.2 Opportunistic use of recommenders versus strategic use

> *Oh, hey, hey, hey! Now I'll still, yeah, now I found a really good one! I mean true enough. I was kinda left feeling a bit vexed about Kerouac. I mean Kerouac is for me kinda like, I mean I notice that I'm chasing after him here. This "Customers Who Bought This Item Also Bought" is throwing at me this* Windblown World: The Journals of Jack Kerouac 1947-1954. *... Well, this showed itself to be useful, you know, something like this can pop up out of nowhere at you.* – P4

The participants used recommendations in two ways, strategically and opportunistically. Strategic use refers to using recommenders intentionally as a part of the current item-finding strategy. The strategy might be accessing personalized recommendations, as P2 and P3 did, or searching for a particular book to see "Customers Who..." recommendations, as P6 did by finding a book by an author to see what other book was recommended on the item page of one of his books. He had no interest in buying that particular book, but he wanted to see similar books. Intentionality shows in two ways in this strategy: in P6's deliberate intention to go to the recommender to see what was recommended and in an attempt to influence the type of books to be recommended.

Opportunistic use refers to users stumbling upon recommendations and using them there and then. It lacks the intention that characterizes the strategic use. Opportunistic use is possible only if recommender features are displayed at the right point of the searching process.

Recommendations that require us to access them intentionally, such as personalized recommendations ("Recommended for You") can only be used strategically. However, recommenders that are displayed as a part of the interface, such as "Customers Who...", can and are used strategically in addition to being used opportunistically.

The secret to helping users use recommenders opportunistically is to deliver them when users are pre-disposed to attending to them. For instance, when P5 when was vacillating between two books, a "Perfect Partner" recommendation that recommended the two books together at what appeared to be a slight discount (it was not) helped P5 to decide to take both. In the same way, delivering recommendations to P1 when he had put one item in the cart caught him at the moment he was not about to do anything else, and so he was pre-disposed to check the suggestions out.

"Customers Who..." recommendations work in a similar manner. If the user is not sure about the book on the item page, he or she is likely to be interested in other options available. Thus, designing recommenders for a complex information environment includes positioning them in the process that they are supposed to support.

Both opportunistic and strategic uses of recommenders are ephemeral, and users cannot be categorized by them. Although P3 did simply look at the personalized recommendation in Task 1 and found a book, thus using recommenders only strategically, in any longer item-finding process users are likely to move from strategic use to opportunistic use and back again. How smooth the transitions are depends on how well the environment is designed to support discovery and what user strategies can emerge from that environment.

3.3 Recommenders and perceived social presence

The participants divided into two groups as far as perceived social presence of the environment was concerned. P1, P2, and P5 felt that they were alone in an online shop and that the social texture did not make the environment any more social. P1: "*It is more like a convey belt than a social environment. ... I don't really see it as social environment and the reviews by anonymous people don't help to make it any more humane.*"

While P2 and P5 found no social aspects whatsoever in Amazon, P1 did relent his position a bit. He remembered having looked at the other reviews of reviewers once or twice and having had a feeling that "*he's interested in the same things as me.*" He thought that if he bought more books, spent more time in Amazon, and consequently looked more at other reviews by reviewers and used other similar features, he might begin to perceive the environment as more social.

For P3, P4, and P6, on the other hand, the social texture made the environment inherently social. P3 felt that the recommendations "*enlivened*" the environment and that without personalization and personalized recommendations it would appear "*dead.*" Likewise, P6 felt that the presence of the community was very positive: "*It's like that, you know, ok, yes, others had felt the same thing about it, about this book, and oh, ok, he thought like that, I don't agree but it's good to know that people can see it like that, too. It goes like that; the community emerges out of it.*"

P4 perceived the environment as even more social than the other two. He explained how the social aspect affected his behavior when he found a review that 95 people out of 95 had found Helpful: "*I felt that it wasn't helpful, but like I said, I won't click the button because then I'd be a killjoy. That's where the sociability kicks in. Then there was one where three had read it, I mean, had evaluated it and all agreed that it was not helpful. So I somehow thought that I'll rebel against it and be the first to think that it is helpful. Then I'd actually do something positive [laughs]. That I didn't click the [first] button or that I would have clicked the [second] button, the motivation didn't have anything directly to do with the book or even the review but all to do with the social context and how I perceived that social situation. . . . the critical mass of Joe Blows, then the social dimension kicks in and those who disagree no longer have the face to disagree [laughs] and do it [vote a review Helpful or not] when the critical mass has been reached.*"

All the participants did use social skills and social cues available to assess the needs, level of expertise, and even personality of the Customer Reviews writers to mirror them against their own to assign relevance and reliability to the reviews. Furthermore, decisions influenced by Customer Reviews to buy or not to buy a book, or to look in more detail a book because it had five stars, were all actions that were influenced by the social texture.

However, we feel that what makes an environment social or not is the perception. If a user perceives that other users are present because of the recommendations, then the environment is social for that user, and if a user perceives recommendations are part of the convey belt shopping environment, then the environment is not social for that user.

Consequently, the arguably rich social texture in Amazon is not alone enough to make a user to perceive the environment as inherently social. How easily people perceive an environment as social is probably related to their personality and personal definition of sociability. For instance, P5 did not even see going to a brick-and-mortar bookstore as social activity: "*I don't go to a bookstore to be social.*" Furthermore, what constitutes social texture might differ from one user to another. Nevertheless, it seems that some people need only a slightest of hint to perceive an environment as social while others require synchronous conversations with video image.

4 Conclusions

Recommenders are integral parts of complex information environments, and their importance is likely to continue to increase in e-commerce as well as in other information environments. While not replacements for keyword searches, they are already an integral part of user strategies for finding items of interest.

Recommenders are used both strategically (intentionally as a part of the item-finding strategy) and opportunistically (when seen without prior intention to use or influence the recommendations). Giving users better ways to influence recommendations and their presentation, such as the order in which Customer Reviews are displayed, is one way to assist users in using recommenders efficiently. The better we understand the underlying and overall process, the better we can assist users to make use of the features in the environment and tailor the tools for actual use.

Recommenders are parts of the social texture that increases the perception of social presence that in turn influences user behavior. However, the effect is not uniform as only half of the participants perceived Amazon as a social environment and half did not. While we saw examples of the social aspects influencing user behavior, the "social effect" cannot be generalized to all users. The behavioral effects and what constitutes social texture to different users require further study, but based on this study, we know that such effects do take place.

While punters talk about combining searching and recommenders, users are already doing it in practice by seeding recommendations with searches to generate discovery. Studies that concentrate in parts need to be accompanied with studies that study the environments as integral wholes. Otherwise, the actual use and predicted use may not meet.

5 Acknowledgments

This work was supported by the Finnish Funding Agency for Technology and Innovation (project 40279/05) and by the NORDUnet project PriMa (Privacy in the Making).

References

[1] Cyr, D., Hassanein, K., Head, M., and Ivanov, A. (2007). The Role of Social Presence in Establishing Loyalty in E-Service Environments. *Interacting with Computers* 19(1), 43–56.

[2] Freyne, J., Farzan, R., Brusilovsky, P., Smyth, B., and Coyle, M. (2007). Collecting Community Wisdom: Integrating Social Search & Social Navigation. In *IUI'07 Proceedings of the 12th International Conference on Intelligent User Interfaces*, 52–61.

[3] Gefen, D. and Straub, D. W. (2004). Consumer Trust in B2C E-Commerce and the Importance of Social Presence: Experiments in e-Products and e-Services. *Omega* 32(6), 407–424.

[4] Grabner-Kräutera, S. and Kaluscha, E. A. (2003). Empirical Research in On-line Trust: A Review and Critical Assessment. *Int. J. Human-Computer Studies* 58, 783–812.

[5] Hangartner, R. (2007). What is the Recommender Industry? Msearchgroove, December 17, 2007.

[6] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. on Information Systems* 22(1), 5–53.

[7] Kotha, S. (1998). Competing on the Internet: The Case of Amazon.com. *European Management Journal* 16(2), 212–222.

[8] Kumar, N. and Benbasat, I. (2006). The Influence of Recommendations and Consumer Reviews on Evaluations of Websites. *Information Systems Research* 17(4), 425–439.

[9] McNee, S. M., Riedl, J., and Konstan, J. A. (2006). Accurate Is Not Always Good: How Accuracy Metrics have Hurt Recommender Systems. In *CHI'06 Extended Abstracts on Human factors in Computing Systems*, 1097–1101.

[10] O'Brien, J. (2006). The Race to Create a 'Smart' Google. *FORTUNE Magazine*, November 20, 2006.

[11] Reichheld, F. F. and Schefter, P. (2000). E-Loyalty: Your Secret Weapon on the Web. *Harvard Business Review* 78(4), 105–113.

[12] Riedl, J. and Dourish, P. (2005). Introduction to the Special Section on Recommender Systems. *ACM Trans. on Computer-Human Interaction* 12(3), 371–373.

[13] Sillence, E. and Briggs, P. (2007) Please Advise: Using the Internet for Health and Financial Advice. *Computers in Human Behavior* 23(1), 727–748.

[14] Svensson, M., Höök, K., and Cöster, R. (2005). Designing and Evaluating Kalas: A Social Navigation System for Food Recipes. *ACM Trans. on Computer-Human Interaction* 12(3), 374–400.

Social Wisdom for Search and Recommendation

Ralf Schenkel, Tom Crecelius, Mouna Kacimi, Thomas Neumann,
Josiane Xavier Parreira, Marc Spaniol, Gerhard Weikum
Max-Planck Institute for Informatics
Saarbruecken, Germany
schenkel@mpi-inf.mpg.de

Abstract

Social-tagging communities offer great potential for smart recommendation and "socially enhanced" search-result ranking. Beyond traditional forms of collaborative recommendation that are based on the item-user matrix of the entire community, a specific opportunity of social communities is to reflect the different degrees of friendships and mutual trust, in addition to the behavioral similarities among users. This paper presents a framework for harnessing such social relations for search and recommendation. The framework is implemented in the SENSE prototype system, and its usefulness is demonstrated in experiments with an excerpt of the librarything community data.

1 Introduction

Social networks and online communities provide a great potential for harnessing the "wisdom of crowds", with social interactions of individual users and user groups taken into account. For example, bookmark-sharing services such as del.icio.us can generate collaborative recommendations based on the quality and trust assessment of web pages as well as users. Social-tagging platforms such as flickr, librarything, or lastfm enable community formation, based on common thematic interests, and thus provide ratings and rankings of photos, books, music, etc., based on the social interactions among many users.

These settings resemble the paradigm of collaborative recommendation [5, 12, 17, 19], which applies data mining on customer-product and similar usage data to predict items that users are likely interested in. Such recommendations leverage user-user similarities as well as item-item similarities. For the first aspect, joint behaviour patterns of two users can be exploited, e.g., the number of items purchased by both users. For the second aspect, the overlap in the interests of users in two items can be exploited, e.g., the number of users who purchased both of two items. A popular approach is to apply data-analysis methods (e.g., spectral decomposition) to a user-item matrix.

Social wisdom for searching, ranking, and recommending items differs from such traditional recommender systems in two important ways:

1. There are explicit *friendship* and *trust* relations among users that are orthogonal to similarities of interests and behavior, and these truly social relations can significantly affect the quality of recommendations.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

In contrast, traditional recommenders consider the user community only in its entirety, whereas social recommenders would discriminate different users based on friendship strengths, mutual trust, etc.

2. There is often a search or discovery *context* like sets of keywords (not necessarily corresponding to the tags of the existing data items) or a real-life task like planning a trip or buying christmas presents, that characterize the ideal results that the user hopes to obtain. This is in contrast to the weakly parameterized nature of traditional recommenders where you can start with a given item to discover new items but not with a vague description that does not match any existing item.

This paper presents a framework for exploiting social wisdom in such a setting, and discusses the ranking of search results and recommendations. Despite the relatively young age of social-tagging communities, there is already a sizable body of literature on a variety of socially enhanced scoring and ranking functions, e.g., [7, 14, 24]. However, most of the prior work has focused on very specific points, such as applying generalized link analysis (i.e., PageRank-style notions of FolkRank, UserRank, SocialRank, etc.) to identify the most central (influential) users or items; and some of the empirical studies have actually raised doubts about the benefit of social tags and friendship relations for improving search [11, 13]. In contrast, this paper aims at a comprehensive framework for scoring, with consideration of both social relations and semantic/statistical relations among items and tags. To this end, we introduce a versatile and richly parameterized scoring model, and we present experiments with librarything data and user-provided quality assessments that demonstrate significant benefits. Throughout the paper we disregard efficiency and scalability issues; these are challenging, too, but out of scope (refer to [4, 8, 20, 23] for efficiency issues).

The paper is organized as follows. Section 2 presents our framework for modeling social-tagging networks and our prototype system coined SENSE (standing for "Socially ENhanced Search and Exploration"). Section 3 presents the socially enhanced scoring model for search and proactive recommendation. Section 4 discusses a user study for experimentally evaluating our approach, based on an excerpt of the librarything community. We conclude with lessons learned and an outlook on further research opportunities.

2 SENSE Framework

We studied a variety of social-tagging platforms, most notably, `del.icio.us`, `flickr`, `librarything`, and `lastfm`, in order to come up with a unified set of abstractions that can model the user-provided data and activities in such communities. The resulting model can be cast into a relational schema of the following form (with unique keys underlined):

$$
\begin{array}{ll}
\texttt{Users}(\underline{\texttt{username}}, \texttt{location}, \texttt{gender}, \ldots) & \text{//abbreviated: U} \\
\texttt{Friendships}(\underline{\texttt{user1}, \texttt{user2}, \texttt{ftype}}, \texttt{fstrength}) & \text{//abbreviated: F} \\
\texttt{Documents}(\underline{\texttt{docid}}, \texttt{description}, \ldots) & \text{//abbreviated: D} \\
\texttt{Linkage}(\underline{\texttt{doc1}, \texttt{doc2}, \texttt{ltype}}, \texttt{lweight}) & \text{//abbreviated: L} \\
\texttt{Tagging}(\underline{\texttt{user}, \texttt{doc}, \texttt{tag}}, \texttt{tweight}) & \text{//abbreviated: T} \\
\texttt{Ontology}(\underline{\texttt{tag1}, \texttt{tag2}, \texttt{otype}}, \texttt{oweight}) & \text{//abbreviated: O} \\
\texttt{Rating}(\underline{\texttt{user}, \texttt{doc}}, \texttt{assessment}) & \text{//abbreviated: R}
\end{array}
$$

We refer to all kinds of data items as *documents*, using IR jargon. These are the items that users explicitly upload (e.g., their own photos) or bookmark and annotate (e.g., web pages, books, songs). Items may be cross-referenced by different types of (possibly weighted) links.

Friendships are user-user relations that come in different forms; this is why we allow multiple types of F relations captured by the *ftype* attribute. *Social friendship* is an explicit, user-provided relation, which can be symmetric or asymmetric; we assume that such a relation exists only if the users know each other by some interaction (in real life or in cyberspace). *Spiritual friendship* among "brothers in spirit", on the other hand, captures similar behavior such as memberships in the same groups or high overlap in tag usage; these are symmetric relations and they do not assume that spiritually related users know each other. The strength of an F relation between two users could be derived from the users' activities such as overlap in tagged documents or trust measures derived from mutual comments and ratings. We treat $fstrength$ as a pluggable building block; it may also be completely absent.

Tagging is a ternary relation between users, documents, and tags. In full generality, it can *not* be decomposed into three binary relations (users-docs, docs-tags, users-tags) without losing information. Nevertheless, binary-relation (or, equivalently, graph or matrix) representations for tagging are very popular in the literature on social networks for convenience. Our approach preserves the full information and feeds it into a scoring model. Independently of tagging activities, the R relation allows users to rate the quality of individual documents. Alternatively, we could aggregate data from the T relation to derive quality measures (e.g., interest or trust in an information source) and keep it as an attribute of R.

Ontology is a light-weight knowledge base that captures different types of "semantic" relations among tags (e.g., synonymy or specialization/generalization). These relations may be provided by domain experts or imported from real ontologies, or they may be built by applying data-mining techniques to the tagging data. The latter case is more realistic for today's types of social tagging communities and is often referred to as "folksonomies" (folklore taxonomies); in this case, the $oweight$ values could be based on tag-usage statistics.

Note that this model is much richer than the datasets in traditional recommender systems. In addition to the shown relations, we can easily add various kinds of aggregation views, for example, document-tag frequencies aggregated over all users. Also note that not all of its aspects apply to every tagging platform (e.g., only few communities would show the users' home locations, some do not facilitate any cross-references among individual items, etc.). In fact, our experimental studies presented in Section 4 utilize only a subset of our model.

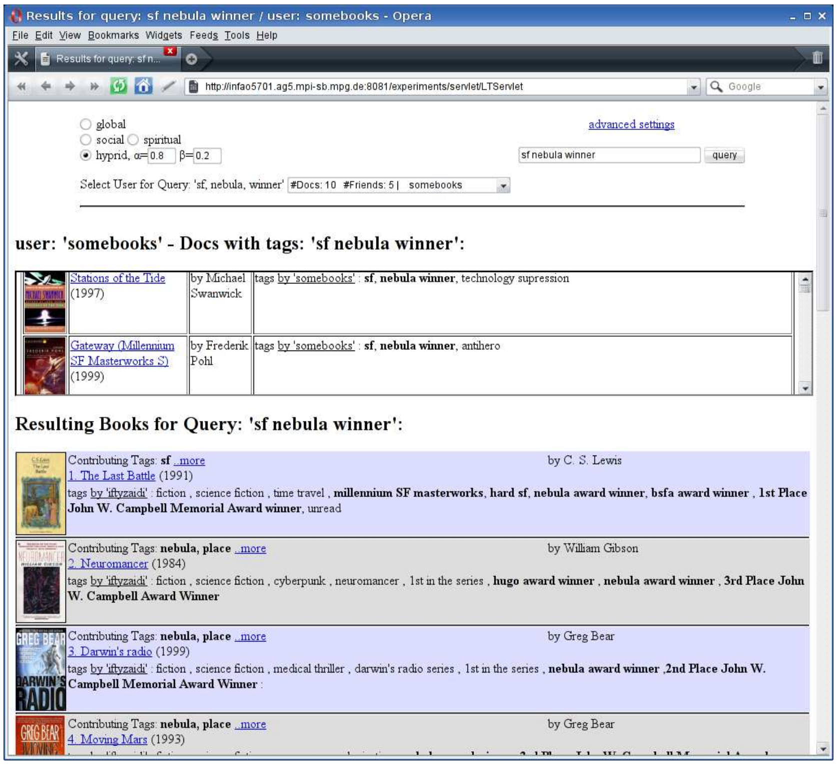


Figure 1: SENSE Screenshot

We have implemented this model based on a relational database system, and populated it with different instances derived from partial crawls of real-life tagging communities. For systematic experimentation with user assessments, we have also built a GUI that supports interactive search and browsing with capabilities for explicit and flexible exploitation of social relations. Figure 1 shows a screenshot of this toolkit, coined SENSE (for "Socially ENhanced Search and Exploration"). The figure shows the ranked results for the query "sf nebula winner" issued by a particular user on an excerpt from the librarything community. The top portion of the results is based on the user's own book collection; the bottom part shows the results obtained by searching other users' collections with consideration of the query initiator's specific friendship relations.

3 Scoring Model

Consider a *query* $Q(u, q_1 \dots q_n)$, issued by a query initiator u with a set of tags $q_1 \dots q_n$. Result documents should contain at least one of the query tags and be ranked according to a *score*. In contrast to standard IR query models, our scoring function can be tuned towards different aspects of social communities. Scores are *user-specific*: they depend on the social and/or spiritual context of the query initiator, according to the configuration of the model. The querying user can decide if her information need is 1) spiritual, 2) social, or 3) global, the latter being agnostic to her social relations.

Friendship Strengths. The core of the scoring model is formed by three different quantizations for user-user affinity or *friendship strength*, corresponding to the three different search modes: spiritual, social, global. Each type of affinity can be implemented in different ways, and our current implementation allows switching between and combining different definitions at run-time. The *spiritual* friendship strength $F_{sp}(u, u')$ of two users u and u', tuned towards spiritual search, is computed based on user-behavior statistics such as overlap of tag usage, bookmarked documents, or commenting and rating activity. The *social* friendship strength $F_{so}(u, u')$, applied for social search, is based on measures like the inverse distance of u and u' in the friendship graph (F relation), but may additionally include other measures. As F_{so} considers also transitive friendships, we have options for different weighting schemes of more distant friends: linearly descending with increasing distance, harmonically descending, geometrically descending, or solely counting immediate friends. The *global* affinity $F_{gl}(u, u') = \frac{1}{|U|}$, used for global searches, gives equal weight to all users. All these measures are normalized such that $\sum_{u' \in U} F(u, u') = 1$ for all u.

The actual friendship strength used to evaluate a query is a linear combination of these three measures:

$$F_u(u') = \alpha \cdot F_{so}(u, u') + \beta \cdot F_{sp}(u, u') + (1 - \alpha - \beta)\frac{1}{|U|}$$

The parameters α and β, $0 \leq \alpha, \beta \leq 1$, can be configured and dynamically adjusted by the user (or an agent on behalf of the user). Extreme choices would be purely spiritual ($\alpha = 0, \beta = 1$), purely social ($\alpha = 1, \beta = 0$), or purely global ($\alpha = 0, \beta = 0$) search; other combinations are reasonable and more interesting.

Score for Tags. To compute the score $s_u(d, t)$ of a document d with respect to a single tag t relative to the querying user u, we use a scoring function in the form of a simplified BM25 score [18]:

$$s_u(d, t) = \frac{(k_1 + 1) \cdot |U| \cdot sf_u(d, t)}{k_1 + |U| \cdot sf_u(d, t)} \cdot idf(t)$$

where k_1 is a tunable coefficient (just like in standard BM25), $|U|$ is the total number of users, $sf_u(d, t)$ is a user-specific tag frequency explained below, and $idf(t)$ is the inverse document frequency of tag t, instantiated as

$$idf(t) = \log \frac{|D| - df(t) + 0.5}{df(t) + 0.5}$$

with $df(t)$ denoting the number of documents that were tagged with t by at least one user. BM25 is a probabilistic IR model that has been very successful and popular in text retrieval. Unlike the original BM25 formula, our model has no notion of document lengths; the number of tags assigned to a document does not vary as much as the length of text documents.

The *socially-enhanced tag frequency* $sf_u(d,t)$, our replacement for the standard term frequency (tf) known from text IR, weights tags by the friendship strength of the query initiator and the user who added the tag to the document. More formally, denoting by $tf_u(d,t)$ the number of times user u used tag t for document d, we define the socially-enhanced tag frequency $sf_u(d,t)$ for a tag t and a document d, relative to a user u, as

$$sf_u(d,t) = \sum_{u' \in U} F_u(u') \cdot tf_{u'}(d,t).$$

For example, if Alice has four (out of her many) friends who have tagged d with t (each once, as it is the norm in social tagging) and each of these immediate friends has weight $1/8$ and only immediate friends matter, then $sf_{Alice}(d,t)$ would be $4 \cdot 1/8 = 1/2$.

Tag Expansion. Even though related users are likely to have tagged related documents, they may have used different tags to describe them. It is therefore essential to allow for an expansion of query tags to "semantically" related tags. To avoid topic drift problems, we adopt the *careful expansion* approach proposed in [22] which considers, for the score of a document, only the best expansion of a query tag, not all of them. More formally, we introduce the *tag similarity* $tsim(t_1,t_2)$ (an instantiation of the $oweight$ attribute of the O relation) for a pair of tags t_1 and t_2, $0 \leq tsim(t_1,t_2) \leq 1$. The final score $s^*_u(d,t)$ of a document d with respect to a tag t and relative to a querying user u, considering tag expansion, is then defined as

$$s^*_u(d,t) = \max_{t' \in T} tsim(t,t') \cdot s_u(d,t')$$

A good source for high-quality tag expansions would be human-made ontologies, however for most applications, it is unlikely that they will be available. Our implementation therefore provides several alternatives to compute the similarity between two tags, none of which requires that an explicit ontology is available. The currently preferred one is based on the co-occurrence of the tags in the entire document collection by estimating conditional probabilities:

$$tsim(t,t') = P[t|t'] = \frac{df(t)}{df(t \wedge t')}$$

where $df(t \wedge t')$ is the number of documents that have been tagged by both tags (but possibly by different users). This asymmetric measure (as opposed to symmetric similarities such as Dice or Jaccard coefficients) aims to identify good *specialization* or *instantiation* tags rather than synonymy or generalization. For example, someone searching for "snake" may be happy to see results that contain the specialized tags "Black Mamba", "Cobra", etc., but is not interested in documents that feature more general tags such as "vertebrate" or "animal" as they will probably lead to results that are too general as well. In fact, one would expect a much higher probability, in the underlying dataset, that a document tagged "Cobra" also has the "snake" tag than, conversely, a document tagged "snake" also having the tag "Cobra" (simply because Cobras are only one of many types of snakes). Similar techniques for mining asymmetric tag relations have been used in different contexts (e.g., [2, 6, 9]). Note that the same similarity measure cculd be applied to measure the strength of relationships in an ontology; here, the pairs of tags under consideration would be those connected by an edge in the ontology.

The above form of tag expansion captures "semantic" associations, but disregards the social relations among users. For *socially-enhanced tag expansion* we compute the similarities between tags in a way that gives a co-occurrence higher weight if the two tags were given by a close friend of the current user u. This idea leads to the formula:

$$tsim_{so}(u,t,t') = \sum_{u' \in U} F_u(u') \cdot \frac{df_{u'}(t')}{df_{u'}(t \wedge t')}$$

where $df_{u'}(t \wedge t')$ is the number of documents tagged by the same user u' with both t and t'. Intuitively, we postulate that user u is interested in seeing Ferraris as a result to a query about sports cars if her friends prefer Ferraris and show this in their tagging activities.

Score for Queries. Finally, the score for an entire query with multiple tags $q_1 \ldots q_n$ is the sum of the per-tag scores:

$$s_u^*(d, q_1 \ldots q_n) = \sum_{q_1 \ldots q_n} s_u^*(d, q_i)$$

Note that this score assumes an IR-style "andish" query evaluation: not all query tags must be matched, but more matches typically lead to higher scores. However, the model can easily be extended to conjunctive evaluation by setting $s_u^*(d, q_1 \ldots q_n) = 0$ when at least one of the $s_u^*(d, q_i) = 0$.

4 Experiments

4.1 Setup

To study the effectiveness of the socially-enhanced scoring model, we performed experiments with data extracted from partial crawls of the `del.icio.us`, `flickr`, and `librarything` sites. We concentrate on the librarything data here, for lack of space and also because this is the most interesting of the three scenarios. We found the social aspects in del.icio.us to be rather marginal, as most bookmarked pages are of fairly high quality anyway; so a user does not benefit from his friends' recommendations more than from the overall community. Flickr has recently grown so much that the tagging quality seems to be gradually degrading; only the owner of a photo provides tags, and these are sometimes relatively unspecific annotations that are given to all photos of an entire series (e.g. vacation July 2007). Librarything, on the other hand, features intensive tagging of a quality-controlled set of items, namely, published books, and its users have built up rich social relations. Finally, book recommendation is a matter of subjective taste, so that social relations do indeed have high potential value. You trust your friends' taste, not necessarily their "technical" expertise.

We extracted the following data from the librarything site: 11,717 users who together own or have read 1,289,128 distinct books with a total of 14,738,646 tagging events (including same tags for the same book by different users), and 17,915 explicit friendships. For the latter, we used the librarything notion of friends (where users mutually agree on being friends) and the notion of referring to an "interesting library". The users included 6 users from our institute who have been contributing to librarything for an extended time period and have made various social connections. These 6 users ran recommendation queries and assessed the quality of the results in our study. Note that such human assessment is indispensable for this kind of experiments, and in our setting it was crucial that a query result was assessed by the same user who posed the query. Altogether, our 6 test users ran 49 queries, shown in Table 3.

Query results were computed for a variety of scoring models: different values of α and β and different strategies for tag expansion. The results from all runs for the same query were pooled; all of them together were shown to the corresponding user in random order (in a browser-based GUI), and the user assessed the quality of each result by assigning one of three possible ratings: 0 = irrelevant or uninteresting, 1 = relevant and interesting, 2 = super-relevant and very appealing. Results that the user already knew, that is, books that she has in her own library, are always discounted.

As for quality measures, we computed, for each run separately:

- the *precision* for the top-10 results, treating both ratings of 2 and 1 as relevant,
- the *normalized discounted cumulative gain (NDCG)* [15] for the top-10 cutoff point. DCG aggregates the ratings (2, 1, or 0) of the results with geometrically decreasing weights towards lower ranks ($DCG \propto$

user 1	user 2	user 3	user 4	user 5	user 6
thailand travel	web learning	time traveler	religion god world	information retrieval	sf nebula winner
asia guide travel	mountain climbing	leonardo vinci	challenge theory	probability statistics	fantasy politics
technology enhanced learning knowledge management	kali death	english grammar	imagination fantasy science	database system	fantasy dragaera
multimedia metadata standards	buddha	romance prague	drama story novel	transaction management	sf nuclear war
knowledge management media theory	houdini	brazilian literature	magic fantasy	data mining	fantasy malazan
social network analysis theory	science illusion magic	shalespeare play	india philosophy	software development	
multimedia social software	mystery magic	stephanie plum	fantasy story		
	religion irony humor	search engines	novel family life		
	yakuza	spanish literature	science fiction future		
	hitman	portuguese literature			
		harry potter			
		wizard			

Table 3: Queries of the user study

α \ β	0.0	0.2	0.5	0.8	1.0
0.0	0.666	**0.698**	**0.688**	**0.682**	0.680
0.2	0.661	0.678	0.686	**0.690**	n/a
0.5	0.637	0.657	0.663	n/a	n/a
0.8	0.612	0.647	n/a	n/a	n/a
1.0	0.549	n/a	n/a	n/a	n/a

Table 4: Precision[10] for all users

α \ β	0.0	0.2	0.5	0.8	1.0
0.0	0.546	**0.572**	**0.568**	0.565	0.565
0.2	0.564	**0.572**	**0.579**	**0.581**	n/a
0.5	0.539	0.552	0.559	n/a	n/a
0.8	0.515	0.546	n/a	n/a	n/a
1.0	0.465	n/a	n/a	n/a	n/a

Table 5: NDCG[10] for all users

$\sum_{rank\ i} \frac{2^{rating(i)}-1}{log_2(1+i)}$) and is then normalized into NDCG by dividing by the DCG of an ideal result (first all results with rating 2, followed by all results with rating 1, followed by results with rating 0). NDCG is a widely adopted standard measure in IR.

4.2 Results

Tables 4 and 5 show the precision and NDCG values for different choices of the configuration parameters α and β, without any form of tag expansion. These are micro-averaged results over all test users. Values printed in boldface are results that were significantly better than the baseline case ($\alpha = \beta = 0$) according to a statistical t-test with test level 0.1.

The results show that both social (increasing α) and spiritual (increasing β) processing can improve the result quality. This holds for each of these two directions individually, and the combined effect is even better with a typical maximum at $\alpha = 0.2$ and $\beta = 0.8$. It may seem that the improvements, for example, from an NDCG value of 0.546 for the baseline to 0.581 for the best case, is not impressive. However, one has to keep in mind that differences in such effectiveness measures generally tend to be small in IR experiments as opposed to efficiency differences (e.g., response times) in the DB literature; we emphasize that the gains are statistically significant. Moreover, it is worth pointing out that for some individual users (i.e., micro-averaging over the queries of one user only) or for individual queries the gains are higher. As anecdotic evidence, the query “science illusion magic” posed by User 2 strongly benefited from the user’s social relations: with global scoring alone, many good results were missed; with spiritual scoring alone, the results drifted towards a big “Harry Potter” cluster which was not what the user wanted; only the combination of social and spiritual similarity gave the excellent results that the user appreciated (which included novels such as “Prestige”, “Labyrinths”, “Invisible Cities”).

α \ β	0.0	0.2	0.5	0.8	1.0
0.0	0.545	0.565	0.565	0.563	0.565
0.2	0.561	0.573	0.581	0.582	n/a
0.5	0.538	0.550	0.554	n/a	n/a
0.8	0.506	0.540	n/a	n/a	n/a
1.0	0.459	n/a	n/a	n/a	n/a

Table 6: NDCG[10] with expansion, up to 5 expansions per tag

α \ β	0.0	0.2	0.5	0.8	1.0
0.0	0.537	0.560	0.558	0.556	0.556
0.2	0.535	0.550	0.567	0.564	n/a
0.5	0.515	0.536	0.545	n/a	n/a
0.8	0.487	0.522	n/a	n/a	n/a
1.0	0.454	n/a	n/a	n/a	n/a

Table 7: NDCG[10] with social expansion, up to 5 expansions per tag

Tables 6 and 7 show the NDCG results with tag expansion enabled, aggregated over all 49 queries of the user study. We compared the purely semantic expansion that ignores social relations against the socially-enhanced tag expansion that prefers tags used by friends. Across the entire query mix of all users, neither of the two expansion methods achieved significant improvements, but again, for individual users such as User 2 there were noticeable gains. For example, the query "Yakuza" created the expansion tags "Cosa Nostra", "Triads", and "nightclub" (among the top-5 expansions); the first and second expansion could have been expected (and created also by an ontology-based method), but the third expansion really reflected tag co-occurrences and implicitly the contents of the kinds of novels that the user wished to discover. Social expansion, on the other hand, did not improve results; in fact, it sometimes reduced the quality. For example, by considering the friendships of User 2, the "Yakuza" query ended up with the expansion "Ninjas" and led to poorer query results.

5 Lessons Learned and Open Issues

In this paper, we have developed a comprehensive framework for socially enhanced search, ranking, and recommendation. Our experimental evaluation exhibits interesting results and indicates the potential of exploiting social-tagging information for scoring and ranking. However, the results reveal mixed insights, and thus also underline the need for further investigating this line of research.

The combination of social and spiritual scoring nicely improved the results of certain queries or users, but also led to result degradation in other cases. On average, there is a significant gain but it is not as impressive as one could have hoped for. It seems that categorizing queries and identifying the query types that can benefit from social and spiritual relations is the key to a robust solution that would choose non-zero values for α and β only when benefits can be expected. In our user study, the queries seem to fall into the following four categories:

1. Queries with a purely *global* information need that perform best when $\alpha = \beta = 0$; examples are "Houdini", "search engines", "English grammar", all fairly precisely characterized topics with objectively agreeable high-quality results.

2. Queries with a subjective-taste and thus *social* aspect that perform best when $\alpha \approx 1$; an example is the query "wizard". This query produces a large number of results but the user may like only particular types of novels such as "Lord of the Rings", for which "wizard" is a relatively infrequent tag overall but was frequent among that user's friends.

3. Queries with a spiritual information need that perform best when $\beta \approx 1$; an example is the query "Asia travel guide" where one can harness the aggregated expertise of the entire user community without consideration of social relations.

4. Queries with a mixed information need that perform best when $\alpha, \beta \approx 0.5$; an example is the query "mystery magic".

Obviously, these lessons are still very preliminary. Our future work aims at developing a principled understanding of query properties and their potential for socially-enhanced recommendation. Other issues that are worthwhile addressing include the *temporal evolution* of tagging and social relations (see, e.g., [3, 10]) and the notion of *diversity* in query results and recommendations (see, e.g., [16]). For interesting and surprising discoveries, you want to benefit from the natural diversity of cultures and tastes in your social network. (Even computer geeks should have some friends who are not in the IT business or in computer science.) Finally, efficiency and scalability in indexing and query processing pose major research challenges as well, and are being addressed in ongoing work such as [1, 20, 23].

References

[1] S. Amer-Yahia, M. Benedikt, P. Bohannon. Challenges in Searching Online Communities. *IEEE Data Eng. Bull.* 30(2), 2007.

[2] R. A. Baeza-Yates, A. Tiberi. Extracting Semantic Relations from Query Logs. KDD 2007.

[3] N. Bansal, N. Koudas. Searching the Blogosphere. WebDB 2007.

[4] S. Bao, G.-R. Xue, X. Wu, Y. Yu, B. Fei, Z. Su. Optimizing Web Search Using Social Annotations. WWW 2007.

[5] R.M. Bell, Y. Koren, C. Volinsky. Modeling Relationships at Multiple Scales to Improve Accuracy of Large Recommender Systems. KDD 2007.

[6] P. Cimiano, A. Hotho, S. Staab. Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis. J. Artif. Intell. Res. (JAIR) 24, 2005.

[7] A. Damian, W. Nejdl, R. Paiu. Peer-sensitive Objectrank - Valuing Contextual Information in Social Networks. WISE 2005.

[8] A. Das, M. Datar, A. Garg, S. Rajaram. Google News Personalization: Scalable Online Collaborative Filtering. WWW 2007.

[9] J. Diederich, W.-T. Balke. The Semantic GrowBag Algorithm: Automatically Deriving Categorization Systems. ECDL 2007.

[10] M. Dubinko, R. Kumar, J. Magnani, J. Novak, P. Raghavan, A. Tomkins. Visualizing Tags over Time. *ACM Transactions on the Web*, 1(2), 2007.

[11] S. Golder, B. A. Huberman. Usage Patterns of Collaborative Tagging Systems. *Journal of Information Science*, 32(2), 2006.

[12] J. L. Herlocker, J. A. Konstan, L. G. Terveen, J. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems*, 22(1), 2004.

[13] P. Heymann, G. Koutrika, H. Garcia-Molina. Can Social Bookmarking Improve Web Search? WSDM 2008.

[14] A. Hotho, R. Jaeschke, C. Schmitz, G. Stumme. Information retrieval in folksonomies: Search and ranking. ESWC 2006.

[15] K. Järvelin, J. Kekäläinen. IR Evaluation Methods for Retrieving Highly Relevant Documents. SIGIR 2000.

[16] J.A. Konstan, S.M. McNee, C.-N. Ziegler, R. Torres, N. Kapoor, J. Riedl. Lessons on Applying Automated Recommender Systems to Information-Seeking Tasks. AAAI 2006.

[17] G. Linden, B. Smith, J. York. Amazon.com Recommendations: Item-to-item Collaborative Filtering. *IEEE Internet Computing*, 7(1), 2003.

[18] S.E. Robertson, S. Walker. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. SIGIR 1994.

[19] J. B. Schafer, D. Frankowski, J. L. Herlocker, S. Sen. Collaborative Filtering Recommender Systems. In *The Adaptive Web*, 2007.

[20] R. Schenkel, T. Crecelius, M. Kacimi, S. Michel, T. Neumann, J.X. Parreira, G. Weikum. Efficient Top-k Querying over Social-Tagging Networks. SIGIR 2008.

[21] J. Stoyanovich, S. Amer-Yahia, C. Marlow, C. Yu. Leveraging Tagging to Model User Interests in del.icio.us. AAAI-SIP 2008.

[22] M. Theobald, R. Schenkel, G. Weikum. Efficient and Self-tuning Incremental Query Expansion for Top-k Query Processing. SIGIR 2005.

[23] M.V. Vieira, B.M. Fonseca, R. Damazio, P.B. Golgher, D. de Castro Reis, B. Ribeiro-Neto. Efficient Search Ranking in Social Networks. CIKM 2007.

[24] S. Xu, S. Bao, Y. Cao, Y. Yu. Using Social Annotations to Improve Language Model for Information Retrieval. CIKM 2007.

Social SQL: Tools for exploring social databases

Marc Smith, Vladimir Barash
Microsoft Corportation
Marc.Smith,t-vladba@microsoft.com

Abstract

Social media are constructed from the collective contributions of potentially millions of individuals and present an increasingly common and large scale form of database. As these databases grow in social and technical importance exploration of their structure is needed. The social nature of much of this data is an added opportunity and challenge, providing contributions to social science questions about large scale social behavior while raising technical thresholds caused by the scale and complexity of the data. The detailed records of the interactions of users of social media form a foundation for higher level representations of the behavior of users and communities in these systems. Social networks are a key structure in social media databases. In the following several visualization tools are used to illustrate social networks and other structures within these data sets. These images highlight behavioral motifs that can be understood through social science theories about roles and social structure. The result is a deeper understanding of the dynamics that drive the creation of user generated content in social media. This process suggests the need for an extension for the structured query language (SQL) that explicitly supports social queries.

1 Introduction

Social life increasingly takes place through computer mediated interaction systems, and these systems are growing in terms of affordances and social importance. Social networking and Web 2.0 services are the most recent examples in a long line of social media. Ever since email and email lists started to accumulate, social media have grown dramatically in volume and function. Many people's experience of internet communication is the product of multiple complex channels. It is now commonplace for many people to use tools like email, email lists, newsgroups, discussion boards, web forums, blog comments, wiki document and talk pages, instant message conversations, SMS messages, Social Networking Services, photo, audio and video sharing services and several other mechanisms for communication and relationship management. These channels allow for the exchange of a rich collection of digital objects among select or global populations.

When people gather and interact in computer mediated spaces they often leave traces behind which record who does what with whom when. Many computer mediated spaces are linked to databases that record these traces for logging and backup purposes. These databases can be processed to reveal patterns of association and patterns of individual differences present in the data. These patterns tell a story about an ecosystem and its inhabitants, a story about variation and the emergence of stable types of social spaces and the roles participants

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

play within them. A great opportunity exists in the river of real data coming from these systems that enables a focus on empirical studies of large scale naturally occurring data sets composed of large interacting populations. Letting this data tell its story is a goal shared by many who want to understand what happens when millions of people interact through computation.

As these databases grow in social and technical importance exploration of their structure is needed. The emergent structures that result from millions of people using these systems are opaque; the systems themselves rarely provide tools to gain a meta-overview of the system itself. The confluence of increasing computer power and the widespread adoption of Internet social media applications offer a particular opportunity within the larger space of network science. Network structures are becoming the focus of a diverse range of disciplines from physics, biology, and information science to sociology and other social sciences. Commonalities across diverse disciplines are emerging as network structures are found within complex processes from protein biology to international finance.

Despite the variety of social media databases in operation, they often share common core structures in the forms of social networks, conversation and document structures, hierarchies and user profiles. Many of these structures have corresponding time stamps. With effort these data sets can be transformed into visualizations that illustrate higher level structures. In social media data sets these higher level structures include attributes like social roles, cliques, communities, and their historical changes.

An integrated view of social media remains elusive, if only because of the distributed way in which such data is stored across multiple systems, services and geographies. Today, only fragmentary images that feature one or a few systems are available. Metaphorically, studies of social life on the Internet remain in a state similar to meteorology prior to satellite photography. Work to build local maps of social databases is occurring in a number of disciplines.

The challenge of mapping even individual social databases is heightened by the absence of standard “social queries”. Most social media database systems lack tools for dealing with higher level social structures in their datasets. A ”Social SQL” is called for that provides higher level forms of interaction with social databases.

Geographic data is a good analogy for what a Social SQL could be. In GeoSQL, a higher level set of relationships for database objects related to location and connection is provided. GeoSQL offers custom support for the geographic properties of roads and bounded regions so that developers avoid calculating these attributes from scratch. This extension to SQL provides for ”topological relationships between two geographic objects with seven spatial predicates. These predicates are: “EQUAL, MEET, INSIDE, CONTAIN, CROSS, OVERLAP and DISJOINT.” [12] A similar set of operations could be applied to social media datasets. Social queries often want to know if a person is a member of a group or if a group is a sub-group of a larger organization. Other queries want to discover if there is an intersection between people or groups in terms of the common relationships or interest they share. It is common to want to know if two slightly different names refer to the same physical person.

The comparison has limits, however. Geographic space is an objective reality that is not in widespread dispute (even if the boundaries sometimes are). In contrast, social media lie in an amorphous data space whose internal structure remains poorly understood. While many of these geographic operators could make sense in a social media space, there is an absence of a unified underlying terrain. Tracking the shifting memberships of well defined organizations is a challenge as people enter and exit and move from one part of the organization to another each day. Building such a map for more informal groupings is even more challenging. Cities and other geographic entities only slowly move from one jurisdiction to another while informal computer mediated groups may resemble more closely flocks of birds that merge and divide while moving from one perch to another [4].

Social media repositories are populated by multiple entities which are interconnected in complex ways. Mapping social media databases requires the production of higher level representations of the potentially large volumes of individual records of transactions and interactions. Each entity must be aggregated in terms of other entities and across time. For example, individual rates of activity over time and the interaction patterns between many individuals at some point in time are two common elements of many social database visualizations. These higher level elements are a form of accounting system for social interaction. The production of social accounting metadata is a prerequisite for all the efforts to produce social database visualizations.

Maps of computer-mediated social interactions reveal the range of social variation taking place within these systems, the variety of social roles being performed, and the different groupings, clusters and communities into which people aggregate. Traces of conflicts or group efforts to collaboratively construct artifacts are topics of great interest to social scientists who can use these computer-mediated social spaces to further the study of these and other group processes. Social databases have many advantages over traditional data collection methods that involve direct observation and manual coding of events. Data collection is often costly, necessarily limited in scope and time period, and error prone. In contrast, social databases are high fidelity records of some elements of social interactions. Attributes like the time of an event, the identifiers of participants and other objects are all likely to be recorded very accurately.

The attractions of social databases are often simultaneously their greatest challenge in terms of building maps. Before any form of Social SQL can be developed the first step is to define the terrain that it would map. Doing that takes tools for surveying social media data sets. The following is a review of several tools that reveal social patterns in these data sets.

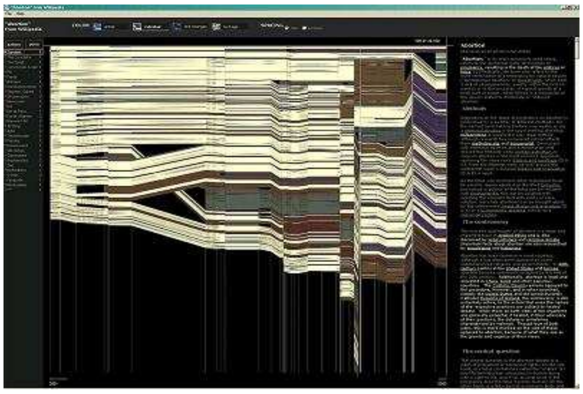
2 Related Work

Visualization has frequently been used in the sciences to illustrate findings of prior quantitative analysis or succinctly summarize those findings to non technical audiences. Visualization can also be used to reveal new relationships, develop hypotheses, refine classification systems, or otherwise discover new insights about how the online social world operates. Many researchers have adopted visualization as an integral strategy of discovery and investigation in research.

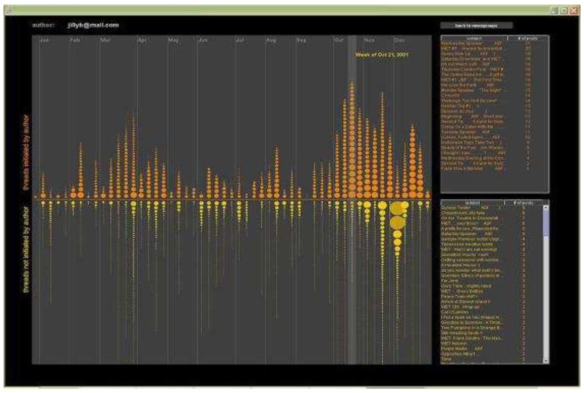
Key data structures in social media are time series, hierarchy, and directed graph. For each of these structures, researchers make use of several visualization strategies are to investigate social media spaces at various scales and levels of detail.

2.1 Time Series

Viegas et al. [10] created “HistoryFlow,” a visualization for examining activity over time on Wikipedia pages. Each version of a Wikipedia page is represented by a colored vertical strip, where different colors represent different editors and the length of the strip represents the volume of contribution in that particular version. Pieces of text that remain the same from version to version are connected by regions of the color corresponding to the respective editor, which allows the viewer to see parts of the page that persist over time. Insertions and deletions manifest themselves as gaps between the connected regions. History Flow analyzes data from online communities to give insight into the evolution of digital artifacts produced by these communities. We present a sample History Flow diagram of the “Abortion” article on Wkipedia below. Notice the shift down and then back up towards the right edge of the graph, which suggests a large (and controversial) section was added to the article, then deleted as status quo was restored.



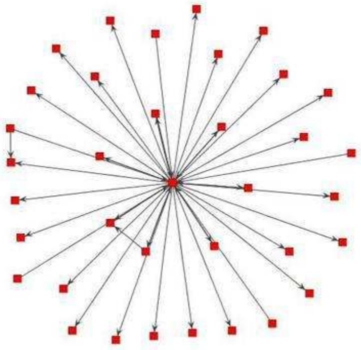
Viegas and Smith [9] take a higher-level approach to visualizing individual activity over time. They develop Author Lines, histograms of user activity in online spaces. These histograms are broken into two halves to represent two contrasting types of activity (e.g. starting newsgroup threads vs. replying to newsgroup threads). The dividing plane is a temporal axis, usually broken down by weeks. For each week, the author line contains zero or more circles of different size. Individual circles represent individual threads started / replied to, and the size of the circle represents the size of the post. Author Lines allow for clear identification of certain iconic roles, such as "Answer person" (his author line would have no activity in the upper half-plane) or "Discussion Person" (his author line would have a small number of large circles). The image below shows the author line for a user with answer person tendencies.

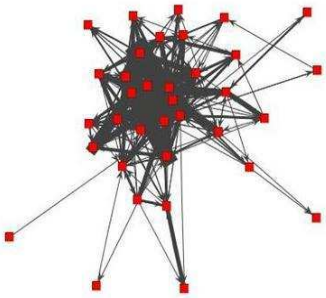


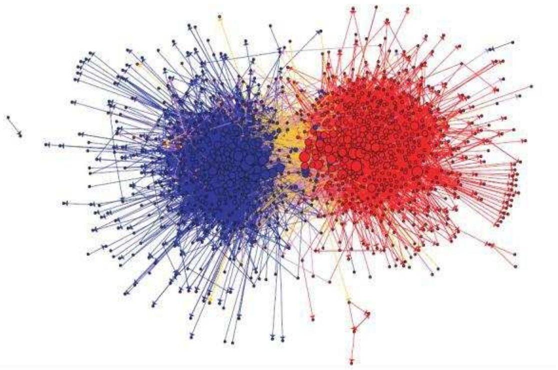
2.2 Directed Graph

A different set of visualizations looks at the social network of interactions in online communities. Welser et al [13] pursue the discovery of iconic roles in Usenet, employing both Author Lines and network diagrams. They construct simple directed networks where nodes represent posters and edges represent replies to other posters. Focusing on individual users, they show only "1.5" degree networks that show only some user, the alters he or she replied to, and the reply edges between those alters. The exemplary networks for an answer person (above, left) and discussion person (above, right) underscore the contrast between these two roles, which is not always visible from Author Lines alone.

At the community level, Adamic and Glance [2] study the network structure of political blogs. In the image below, individual circles are blogs, and edges are URL links between them. Red dots are conservative blogs, blue dots are liberal blogs. The visualization shows not only the clear blog divide along partisan lines, but also the interactions "across the aisle:" orange edges are links from liberal blogs to conservative ones, and purple edges are links from conservative blogs to liberal ones. Given this visualization, a researcher can pinpoint "crossover" blogs that tie the two halves of the online political community together and study them in more detail.

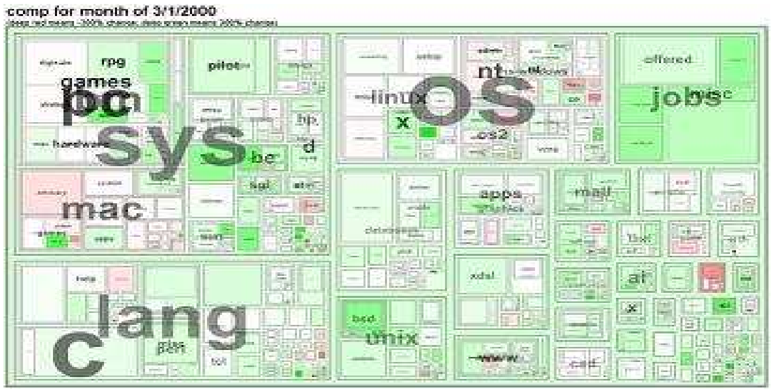






2.3 Hierarchy

Another series of visualizations looks at the community hierarchy. Fiore and Smith [3] use a Tree Map, which collocates all Usenet newsgroups in a rectangle. The highest-level newsgroup labels (alt, soc, comp) partition the rectangle into regions of size proportional to the number of messages that fit under the label. Lower level labels (e.g. soc.culture) partition the region allotted to their parent higher level label in a recursive fashion. The tree map provides a birds eye view of even extremely large communities, such as Usenet. Further, the regions are color coded by the change in number of messages in the respective region since the last time period. Green regions indicate labels with growing numbers of child messages, red regions labels with falling numbers of child messages. The tree map below lays out the Usenet newsgroups with the "comp" label.



3 A Social SQL?

These examples of visualizations of key aspects of social media spaces share common data requirements. A step towards defining and implementing a Social SQL is to enumerate the many facets of social databases. Researchers need to make routine workflow operations on social databases. These frequently repeated operations could be standardized so that researchers can build social accounting metadata in a consistent and simple manner.

Managing social graphs is a database chore that often overwhelms non-specialists. Better tools for storing, indexing, searching, and extracting data from social databases would address the need for managing multiple views of large networks that are changing rapidly.

3.1 Social Queries

Social queries range from standard database operations to those that require the creation of complex and specialized logic to generate more complex data like social networks. It is often a fairly straight forward process to extract a time series of behaviors from social databases. But more complex events over time, like the patterns of connections that develop among participants in social databases, are far more challenging to extract. Social network queries often want to limit or extend the network sub-graph which results from the set of nodes returned by the query. Below, we propose an outline of essential queries, written in natural language, to be supported by a SQL extension for social databases. The highest level of the outline contains three fundamental queries (the third query is an ORDER BY operation). Lower levels of the outline include more specific queries that extract the data necessary for generating visualizations above.

- Extract activity time series for a single person
  - Extract activity time series for all people in a community
  - Extract statistical breakdown of activity for all people in a community by user-defined patterns
    - Extract all activity time series in a community that fit a particular pattern (role)
- Extract the directed social network of all people in a community (user provides definition of relationship between two people)
  - Extract the directed social network of all entities in a community (user provides definition of entity and of relationship between two entities)
  - Extract the 1.5 degree network (ego, alters, ties between them) of all entities in a community
  - Given a node in a social database, find all the other nodes that have a similar network structure (e.g. all the nodes with overlapping connection networks)
- Improve the relevance of search results based on the social network attributes of the author of the result documents

3.2 A meta move to ecological models

As social databases are explored and better tools for studying them become available the effect should be to shift our focus from the detail of events or even of roles to a broader focus on the ecosystem of social media spaces. Once roles are well defined it becomes clear that multiple roles exist and play different and sometimes complementary functions within social databases. Ecologies of interactions become the next unit of analysis as tools lift our focus to the ways whole populations vary in structure and performance over time.

4 Discussion

The mining of directed graphs, particularly social networks, is a topic of growing interest [1]. Visualizing these graphs along with other key structures like hierarchies and time series, enables researchers to observe these patterns and gain deeper insights into the dynamics of social databases. As humans gain these insights, more quantitative approaches can pick up on insights gleaned from the observation of rich visualizations. These quantitative measures can evaluate empirical observations about user and community behavior and provide some measure of the goodness of answers. This approach differs from the typical database research effort that is likely to use algorithms to build and test synthetic data and only later test results on real world systems. Instead, this approach seeks to explore naturally occurring social databases to learn about their basic structure and explore opportunities for enhancement and augmentation.

Naturally occurring social data is the key driver and attraction for many people working around social databases. The goal is to let the data tell its story. Initial work derived from the peculiarities of specific datasets drawn from newsgroups, web boards, email lists, wikis and similar repositories will, over time, compose a picture of social databases in general. Evaluating maps of social databases will become possible once enough of these stories are told and generic structures become visible. Patterns discovered in one social database silo may or may not be corroborated in another silo. Only the widespread collection of data across time and systems will allow for the creation of a systematic taxonomy of social databases. These findings may even apply to other forms of datasets that have similar structures even if not socially constructed. For example, complex networks are present in many large biological datasets. These are a set of tools that help analyze any large collection of data.

5 Conclusion

Picturing the complex data structures that are created when humans interact in and through computational media is a challenging but potentially richly rewarding method for discovery. Information visualization techniques have been increasingly applied to the data generated by social media on the Internet resulting in insights that may have been far more difficult to grasp with either qualitative methods based on reading message content or quantitative statistical methods alone. Finding ideal images for various forms of complex data remains a challenge. Nonetheless, several examples of discoveries about the nature and dynamics of social structures point to the value for research based on graphical representations. Data structures like hierarchies, time series, and directed network graphs are common in most forms of computational social spaces. All of these efforts rest on a common set of queries that, like geographic extensions to databases, should ultimately be supported as a special domain of the structured query language (SQL).

Acknowledgements

We would like to thank Derek Hansen and Eric Gleave for assistance with the paper, and Lada Adamic, Andrew Fiore, Fernanda Viegas, and Ted Welser for screenshots of online community visualizations.

References

[1] Adamic, L. and E. Adar, "How to search a social network," Social Networks, vol. 27, no. 3, pp. 187-203, 2005.

[2] Adamic, L. and N. Glance, "The political blogosphere and the 2004 u.s. election: Divided they blog," Working Paper, 2005.

[3] Fiore, A. and M. Smith. “Treemap visualizations of Newsgroups.” Proceedings of CHI, 2002

[4] Rosen, D., Woelfel, J., Krikorian, D., and Barnett, G. (2003). Procedures for analyses of online communities. Journal of Computer-Mediated Communication, 8 (4). Retrieved July 12, 2005 from http://jcmc.indiana.edu/vol8/issue4/rosen.html

[5] Shneiderman, B. (2004). Treemaps for Space-Constrained Visualization of Hierarchies. Retrieved July 12, 2005 from http://www.cs.umd.edu/hcil/treemap-history/

[6] Tufte, E. R. (1995). Envisioning Information (5th printing, August 1995 ed.). Cheshire, Conn.: Graphics Press.

[7] Tufte, E. R. (1997). Visual Explanations: Images and Quantities, Evidence and Narrative. Cheshire, Conn.: Graphics Press.

[8] Turner, Tammara, Marc Smith, Danyel Fisher and Howard T. Welser. 2005. “Picturing Usenet: Mapping Computer-Mediated Collective Action.” Journal of Computer Mediated-Communication. 10 (4).

[9] Vigas, F. B., and Smith, M. A. (2004). Newsgroup Crowds and Authorlines: Visualizing the activity of individuals in conversational cyberspaces. Proceedings of the 37th Hawai’i International Conference on System Sciences. Los Alamitos: IEEE Press.

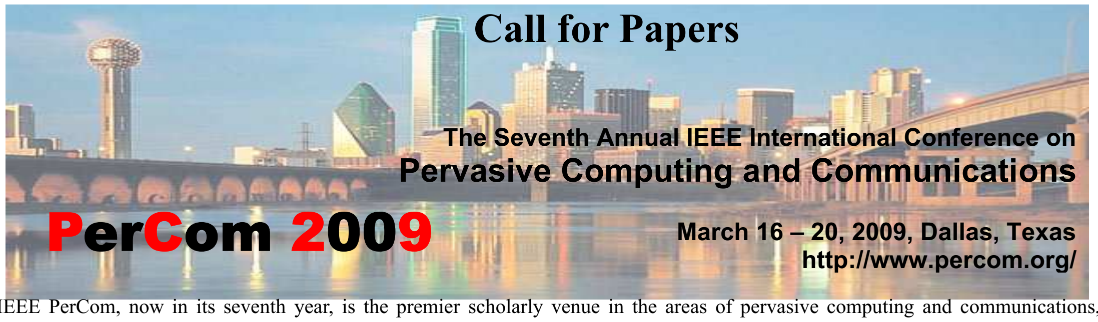
[10] Viegas, F. B., Wattenberg, M. and K. Dave. Studying cooperation and conflict between authors with history flow visualizations. In Proceedings of SIGCHI,pages 575–582, Vienna, Austria, 2004. ACM Press

[11] Wellman, B. 2001. Computer Networks as Social Networks. Science, 293(5537), 2031-2034.

[12] Wang, F., Sha, J., Chen, H. and S. Yang. GeoSQL: A Spatal Query Language of Object-oriented GIS. Retrieved from: http://citeseer.ist.psu.edu./475924.html on 6/16/08.

[13] Welser, Howard T., Eric Gleave, Danyel Fisher, and Marc Smith. 2007. “Visualizing the Signatures of Social Roles in Online Discussion Groups.” The Journal of Social Structure.8(2).

.



IEEE PerCom, now in its seventh year, is the premier scholarly venue in the areas of pervasive computing and communications, which aim at providing a ubiquitous platform for supporting exciting anytime and anyplace services paradigms. Pervasive computing and communications is a natural outcome of the tremendous advances of a broad spectrum of technologies including wireless and sensor networks, mobile and distributed computing, and agent technologies. PerCom 2009 will provide a high profile, leading edge forum for researchers and engineers alike to present state-of-the-art research in the respective fields of pervasive computing and communications. The conference will feature a diverse mixture of presentation forums including core technical sessions, several targeted workshops, demonstrations, keynote speeches and panel discussions from domain experts.

The conference will take place in Dallas-Fort Worth Metroplex, one of the leading technology centers of the United States.

Topics: Research contributions are solicited in all areas pertinent to pervasive computing and communications, including:

- Ad hoc networks for pervasive communications
- Autonomic computing and communications
- Context-aware computing
- Enabling technologies (e.g., wireless BAN, PAN)
- Low power and green pervasive computing
- Middleware services and agent technologies
- Mobile/Wireless computing systems and services in pervasive computing
- Novel/innovative pervasive computing applications
- Pervasive computing and communication architectures
- Pervasive computing in the automotive domain
- Pervasive opportunistic communications and applications
- Pervasive sensing, perception and semantic interpretation
- Positioning and tracking technologies
- Programming paradigms for pervasive systems
- Sensors and RFID in pervasive systems
- Smart devices and intelligent environments
- Trust, security and privacy issues in pervasive systems
- User interfaces and interaction models
- Virtual immersive communications
- Wearable computers

Workshops and affiliated events:

Many workshops will be held in conjunction with the conference. Workshop papers will be included and indexed in the IEEE digital libraries (Xplore), showing their affiliation with IEEE PerCom. As in the past, PerCom 2009 will also feature a PhD Forum, Demonstrations and a Work-in-Progress Session. Please check the conference website frequently for more details.

Submission Guidelines:

Submitted papers must be unpublished and not considered elsewhere for publication. Papers must show a significant relevance to pervasive computing and networking. Only electronic submissions in PDF format will be considered. The page limit is 10 IEEE conference proceedings format 2-column pages (including references, figures and tables, at least 11pt font). Detailed electronic submission procedures will be available at http://www.percom.org. All submitted papers will undergo a rigorous review process managed by the technical program committee. IEEE Press will publish the conference proceedings. The best paper will receive the prestigious Mark Weiser best paper award. Papers of particular merit will be considered for a special issue of Pervasive and Mobile Computing Journal.

Organizing Committee:

General Chair:	Klara Nahrstedt, University of Illinois at Urbana-Champaign, USA
General Vice-chair:	Gergely Záruba, The University of Texas at Arlington, USA
Program Chair:	Chatschik Bisdikian, IBM TJ Watson Research Center, USA
Program Vice-chairs:	Silvia Giordano, SUSPI, Switzerland
	Chandra Narayanaswami, IBM TJ Watson Research Center, USA
	Albrecht Schmidt, University of Duisburg-Essen, Germany
Workshops Co-chairs:	Ben Lee, Oregon State University, USA
	Hao Chu, National Taiwan University
Panels Chair:	Christian Becker, Universität Mannheim, Germany
Steering Committee Chair:	Mohan Kumar, The University of Texas at Arlington, USA

Important Dates:

Paper Registration:	Sept. 15, 2008	Author notification	Dec. 12th, 2008,
Paper submission:	Sept. 22nd, 2008	Camera-ready due:	Jan. 7th, 2009
Workshop proposal:	May 23rd, 2008		

Sponsors: IEEE, IEEE Computer Society (TCPP), UT Arlington

For Further Information, Please Contact: zaruba@uta.edu or visit http://www.percom.org

Call for Participation

34th International Conference on Very Large Data Bases (VLDB 2008)
23-28th August, 2008
Auckland, New Zealand

http://www.vldb.org/2008/

VLDB 2008 is a premier international forum for database researchers, vendors, practitioners, application developers, and users. The conference tracks discuss original results on all aspects of data management, and will cover the most critical issues and views on practical leading-edge database technology, applications, and techniques. We are looking forward to an exciting conference, with tutorials, demonstrations, and for the first time a track on experiments and analyses. VLDB 2008 also hosts a series of workshops that cover important aspects in the context of databases.

Schedule:

Saturday and Sunday, 23 – 24th August:	Workshops
Monday to Thursday, 25 – 28th August:	Main Conference

Information on the conference and registration is available at:

http://www.vldb.org/2008/

See you in Auckland!

IEEE Computer Society
1730 Massachusetts Ave, NW
Washington, D.C. 20036-1903