Proceedings of the ACM RecSys'09 Workshop on

Recommender Systems & the Social Web

EDITORS

Dietmar Jannach, Werner Geyer, Jill Freyne, Sarabjot Singh Anand, Casey Dugan, Bamshad Mobasher, Alfred Kobsa

October 25, 2009

New York, NY, USA

FOREWORD

The Social Web has been enjoying huge popularity in recent years, attracting millions of visitors on sites such as Facebook, Delicious or YouTube. Today, we are no longer mere consumers of information, but we also actively participate in social networks, upload our personal photos, share our bookmarks, write web logs and annotate and comment on the information provided by others. Following the exponential growth in the popularity of Social Web sites, many traditional, non-social sites, are now implementing social features. Likewise many enterprises are deploying internal social media sites to support expertise location and sharing of work-related information and knowledge. The Social Web therefore provides huge opportunities for recommender technology and in turn recommender technologies can play a part in fuelling the success of the Social Web phenomenon.

- New application areas for recommender systems emerge with the popularity of the Social Web. Recommenders can not only be used to sort and filter Web 2.0 and social network information, they can also support users in the information sharing process, e.g., by recommending suitable tags during folksonomy development.
- Social systems by their definition encourage interaction between users and both online content and other users, thus generating new sources of knowledge for recommender systems. Web 2.0 users explicitly provide personal information and implicitly express preferences through their interactions with others and the system (e.g. commenting, friending, rating, etc.). These various new sources of knowledge can be leveraged to improve recommendation techniques and develop new strategies which focus on social recommendation. This social layer can also be used as evidence on which to infer relationships and trust levels between users for recommendation generation.
- The Social Web also presents new challenges for recommender systems, such as the complicated nature of human-to-human interaction which comes into play when recommending people. Or, the design and development of more interactive and richer recommender system user interfaces that enable users to express their opinions and preferences in an intuitive and effortless manner.

- Recommender technology assists social systems through increasing adoption and participation and sustaining membership. Through targeted and timely intervention which stimulates traffic and interaction, recommender technology can play its role in sustaining the success of the Social Web.

The goal of this one-day workshop was to explore, discuss, and understand new opportunities for recommender systems and the Social Web. The workshop consisted both of technical sessions, in which selected participants presented their results or ongoing research, as well as informal breakout sessions on more focused topics.

Papers discussing various aspects of recommender system in the Social Web were submitted and selected for presentation and discussion in the workshop in a formal reviewing process. The topics of the submitted papers included, among others, the following main areas:

- Improved algorithms for tag recommendation for Social Media resources.
- Recommending Social Web resources such as shared bookmarks based on folksonomies, tag contents and personal tagging histories.
- Recommending friends in Social Networks.
- Exploiting trust and other social relationships in Social Networks for improving collaborative filtering recommender systems.
- User modeling and recommendation based on ontologies and Web 2.0 content.
- Attacks on Social Media sites.

The Workshop Organizing Committee

October 2009

CONTENTS

Improving Recommendation Accuracy by Clustering Social Networks with Trust

Tom DuBois
Computer Science Department
University of Maryland, College Park
College Park, MD 20741
tdubois@cs.umd.edu

Jennifer Golbeck
Human-Computer Interaction Lab
University of Maryland, College Park
College Park, MD 20741
jgolbeck@umd.edu

John Kleint
Computer Science Department
University of Maryland, College Park
College Park, MD 20741
jk@cs.umd.edu

Aravind Srinivasan
Computer Science Department
University of Maryland, College Park
College Park, MD 20741
srin@cs.umd.edu

ABSTRACT

Social trust relationships between users in social networks speak to the similarity in opinions between the users, both in general and in important nuanced ways. They have been used in the past to make recommendations on the web. New trust metrics allow us to easily cluster users based on trust. In this paper, we investigate the use of trust clusters as a new way of improving recommendations. Previous work on the use of clusters has shown the technique to be relatively unsuccessful, but those clusters were based on similarity rather than trust. Our results show that when trust clusters are integrated into memory-based collaborative filtering algorithms, they lead to statistically significant improvements in accuracy. In this paper we discuss our methods, experiments, results, and potential future applications of the technique.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Keywords

recommender systems, trust, social networks

1. INTRODUCTION

Trust between users in a social network indicates similarity in their opinions [25, 24, 11]. Hundreds of millions of people are members of social networks online [10] and many of those networks contain trust data. With access to this information, trust has potential to improve the way recommendations are made.

Existing work has used trust to make recommendations by treating it as a weight in collaborative filtering algorithms [3, 9, 15]. This has been effective, but it is not the only way in which trust can be used. If users can be clustered into trusted groups, these may be useful for improving the quality of recommendations made with a variety of algorithms. While clustering can be computationally difficult with social networks, a new trust metric we have developed makes it easy to apply clustering techniques to build these groups. Since trust is related to similarity, we use correlation clustering to identify groups of trusted people.

This leads to the question: since we know that trust can be useful for making recommendations directly, and if we can effectively cluster users by trust, can those clusters be used to improve recommendation accuracy further? There has been some work on using clusters for improving recommendations already. These techniques cluster based on similarity, much as collaborative filtering techniques rely on user similarity. Unfortunately, research has not found an improvement in accuracy using these methods and, in many cases, the recommendation techniques using clusters actually lead to worse performance.

Previous experiments with trust have shown that it leads to improvements over similarity-based recommendation techniques in certain cases [8], and that user-assigned trust ratings capture sophisticated types of similarity [11]. Thus, it is possible that trust clusters may have benefits that are not found when similarity-based clusters are used. Our results show that incorporating trust clusters into collaborative filtering algorithms - including algorithms that use Pearson correlation coefficients and algorithms that use trust - does indeed lead to statistically significant improvemnets.

In this paper, we will present a discussion of our new trust metric and its applications to clustering, the integration of these clusters into recommendation algorithms, the experiment where we tested these algorithms and found the improvement in accuracy, and finally discuss a range of applications where this technique may be beneficial.

2. RELATED WORK

User clustering has been applied to the task of collaborative filtering before, but our work is the first that uses trust as a basis for forming clusters. In this section, we describe work that has been done on using clustering for collaborative filtering and on using trust for collaborative filtering. Then, we introduce the dataset used in our computations and experiments.

2.1 Clustering and Collaborative Filtering

Breese et al. [5] used a Bayesian clustering model to cluster users based on their ratings. Their work showed mixed results; in some cases the clustering approach was competitive in terms of accuracy of the ratings and in others it performed poorly. Ungar and Foster [23] also used a Bayesian approach to cluster users based on their preferences. Their results also showed that clustering users was not a particularly successful approach. Graph theoretic methods for clustering users based on preferences were discussed in [19], however they do not evaluate the impact these clusters have on recommendation accuracy or quality. Finally, in [22] users were clustered using a scalable neighborhood algorithm and, once again, the clustering approach had a higher MAE than the standard collaborative filtering method.

2.2 Trust and Collaborative Filtering

Social networks, and trust in particular, have been used to generate recommendations for users. In these cases, trust is used directly to generate the recommendation. This work follows from the fact that people tend to develop connections with people who have similar preferences [1]. Trusting the opinion of another particularly speaks to this type of similarity. The applicability of this effect to recommender systems has been established in several papers. Ziegler and Lausen [25] that showed a correlation between trust and user similarity in an empirical study of a real online community. Using All Consuming [1], an online community where users rate books. The authors showed that users were significantly more similar to their trusted peers than to the population as a whole. This work was extended in [24] which augmented the analysis of the All Consuming community and added an analysis. The second result in [24] used the FilmTrust system[9] (described below) where users have stated how much they trust their friends in a social network and also rated movies. Within that community, results also showed a strong correlation between trust and similarity in movie ratings. Further work in [11] shows that trust captures similarity in more nuanced ways, such as similarity on items with extreme ratings and large differences.

Empirical results show that using trust from social networks can improve recommendations. O'Donovan and Smyth [20] performed an analysis of how trust impacts the accuracy of recommender systems. Using the MovieLens dataset [18], they create trust-values by estimating how accurately a person predicted the preferences of another. Those trust values were then used in connection with a traditional collaborative filtering algorithm [14], and an evaluation showed significant improvement in the accuracy of the recommendations. Massa and Bhattacharjee [16] also conducted a study on the applicability of trust in recommender systems. Their study relied on the user ratings of products and trust ratings of other users from epinions [2] as their dataset. Using a trust propagation algorithm, similar to that described in section 3, they showed that trust based recommendations could perform significantly better than those based on similarity alone.

In the FilmTrust recommender system mentioned above, trust is used in place of the Pearson correlation coefficient to generate predictive ratings. Results showed that when the user's rating of a movie is different than the average rating, it is likely that the recommended rating will more closely reflect the user's tastes. As the magnitude of this difference increases, the benefit offered by the trust-based recommendation also increases. Moleskiing [3], at http://moleskiing.it, is another real system built to utilize trust ratings in a recommender system. Using a similar approach, it recommends routes to users based on information supplied by trusted peers.

2.3 Dataset

For these experiments, we needed our dataset to have two components:

1. A social network with trust ratings between individuals so we could apply the trust inference algorithm and clustering methods discussed in section 3.
2. Ratings of items by the members of that social network.

We used the FilmTrust dataset [9] for these experiments because it had both these features - a trust network and a set of ratings on movies. Because trust assigned by one user to another is a value that is kept very private, there are no other publicly available datasets with this information. Thus, while FilmTrust provides a good basis for this initial analysis, further analysis with privately held trust networks will likely lead to additional insights.

At the time of analysis, the FilmTrust movie rating dataset has 29,551 ratings by 1,254 unique users over 1,946 movies. That is an average of 15.2 ratings per movie, though some movies have hundreds of ratings. Trust values are assigned on a 1 to 10 scale and are asymmetric; Alice may trust Bob at level n, but Bob may have no trust or a different value of trust in return for Alice. The entire FilmTrust social network has 712 nodes with 1,465 edges and an average of trust rating is 6.83. Many of these nodes are in small groups of two or three disconnected from the main component. For our algorithms, we selected the the giant component and removed nodes with a degree of 1. This left 348 nodes with 1,059 edges in the FilmTrust network. Since our recommendation technique required nodes to be in the social network, we used only the ratings from these 348 nodes. They had 8,457 ratings on 1,558 movies.

3. A PROBABILISTIC TRUST INFERENCE ALGORITHM

As part of our previous work [6], we developed a probabilistic trust inference algorithm that leads nicely to clustering applications. In this section we present an overview of that work and discuss the clustering techniques used.

3.1 The Trust Inference Algorithm

Our work takes a trust network, which may be very sparse since most people will know only a small fraction of the network, and generates inferred trust values between all pairs in the network. We then use these trust values as the basis

[1]http://allconsuming.net/
[2]http://epinions.com

in a trust distance metric space, where the more trust between a pair, the closer they are in the space. One of the major benefits of our approach, and the benefit we exploit here, is the ability to group people into clusters within this metric space. In this section we present our probabilistic trust inference algorithm and describe the properties of its output which make it easy to cluster.

A very intuitive idea motivates this trust inference model. Consider the following scenario:

- Alice knows Bob and thinks he has a $p_{a,b}$ chance of being trustworthy.
- Bob knows Eve and thinks she has a $p_{b,e}$ chance of being trustworthy, and he tells this to Alice if he is trustworthy. If Bob is not trustworthy, he may lie about $p_{b,e}$ and give any value to Alice.
- Alice reasons that Eve is trustworthy if Bob is trustworthy and gives her the correct value $p_{b,e}$ and Eve is trustworthy with respect to Bob.
- This combination happens with probability $p_{a,b}p_{b,e}$ if Bob's trustworthiness and Eve's trustworthiness are independent.

Thus we infer that Alice's trust in Eve should be $p_{a,b}p_{b,e}$. More formally we view any path through the network as a Bayesian chain. Define X_{Bob}, X_{Eve} to be the respective random events that Bob and Eve are trustworthy from Alice's perspective. This is explained in more detail in Figure 1.

The same analysis can be used if trust is a proxy for similarity. Specifically Alice and Bob's mutual trust can be a measure of how similar their tastes in movies are. If trust is interpreted as probability of liking the same film, then Alice will agree with Eve about a movie if (but not necessarily only if) Alice and Bob agree on it and Bob and Eve agree as well.

Our model would not be interesting if it required simple probabilities which can be computed exactly. Fortunately we can quickly estimate trust between individuals in a more complicated network, one with exponentially many, highly correlated paths between pairs of nodes. In these examples, the Bayesian chain view still applies. If there exists a path from Alice to Eve in a random network constructed from trust values, then that path is a chain of people from Alice to Eve who each trust their successor, and Alice can trust Eve. Therefore Alice trusts Eve with the probability that there is a path from Alice to Eve in the random graph.

We define $t_{u,v}$ to be the direct trust between u and v, and $T_{u,v}$ to be our inferred trust value. The direct trust values may be arbitrary, however the inferred trust should obey the axioms in Table 1.

The idea that trust networks can be treated as random graphs underlies this algorithm. For every pair (u, v), we place an edge between them with some probability that depends on $t_{u,v}$. We then infer trust between two people from the probability that they are connected in the resulting graphs. Formally we choose a mapping f from trust value to probabilities. We then create a random graph G where each edge (u, v) exists with probability $f(t_{u,v})$. We then use this graph to generate inferred trust values $T_{u,v}$ such that $f(T_{u,v})$ equals the probability that there is a path from u to v in the random graph. We give a small illustrative example graph in Figure 2. This model is one of many that satisfies our trust axioms.

Axioms of inferred Trust

Local Pessimism	Since $t_{u,v}$ is a pessimistic estimate, indirect information can only increase trust, thus $T_{u,v} \geq t_{u,v}$.
Bottleneck	If all paths from u to v use (a, b), then $T_{u,v} \leq t_{a,b}$, and in general the lower $t_{a,b}$ is, the lower $T_{u,v}$ should be.
Identity	Individuals should completely trust themselves: $T_{u,u} = T_{\max}$.
Complete Trust	If there exists a path $(a_0, a_i, \ldots, a_n)$ such that for all i from 1 to n : $t_{a_{i-1},a_i} = T_{\max}$, then $T_{a_0,a_n} = T_{\max}$.
Monotonicity	For any u, v such that $T_{u,v} < T_{\max}$, augmenting a graph with a new trust path from u to v, or increasing a $t_{a,b}$ value along an existing trust path should increase $T_{u,v}$.
No Trust	For any u, v with no path from u to v, $T_{u,v} = 0$.

Table 1: These rules should apply to any pessimistic system which derives inferred trust from direct trust information.

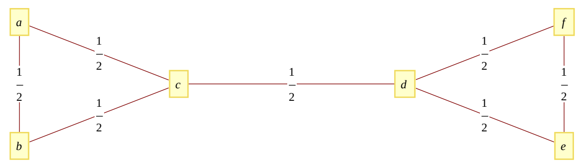


Figure 2: This is an example network with a critical edge. No one from the set $\{a, b, c\}$ can trust anyone in $\{d, e, f\}$ except through the mutual trust between c and d.

$$\begin{aligned} Pr[X_{Eve}] &= Pr[X_{Eve}|X_{Bob}] \cdot Pr[X_{Bob}] + Pr[X_{Eve}|\overline{X_{Bob}}] \cdot Pr[\overline{X_{Bob}}] \\ &\geq Pr[X_{Eve}|X_{Bob}] \cdot Pr[X_{Bob}] = Pr[X_{Bob} \wedge X_{Eve}]. \end{aligned}$$

Figure 1: The second term drops out because Alice has no information about Eve if Bob is not trustworthy. Furthermore, if Eve and Bob are independent, this probability becomes $Pr[X_{Bob}]Pr[X_{Eve}]$.

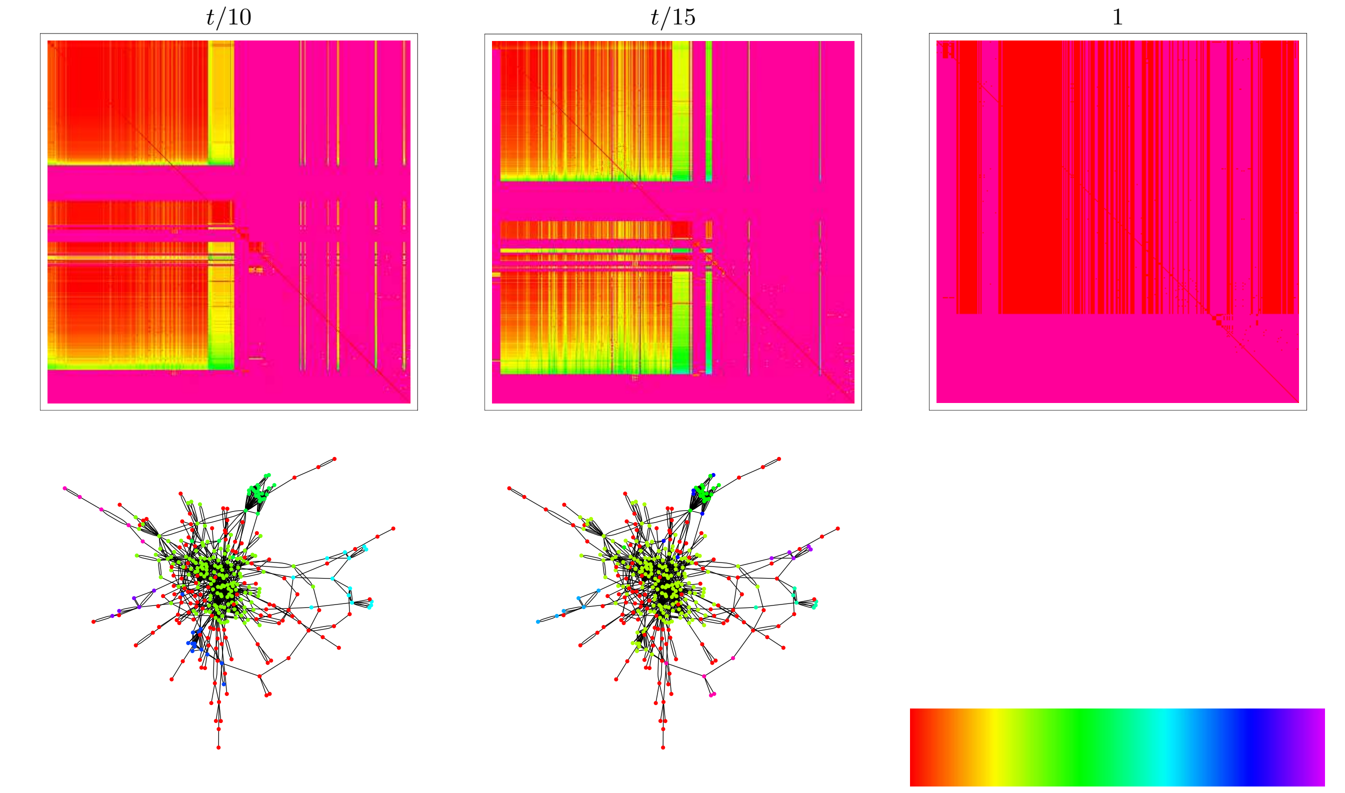


Figure 3: Here we show distance grids, partitionings for the FilmTrust dataset, and the color code for the grids. The i, j pixel in each grid encodes the distance between the ith and jth people accoring to our metric. The color code is given by the bottom rightmost image, with red being 0 distance and violet being distances of 10 and above. The rightmost grid is not probabilistic, but instead shows the transitive closure of the edge set.

3.2 Trust Based Clustering

Given that $f(T_{u,v})$ is the probability of a path connecting u and v. The function $d(u, v) = \log \frac{1}{f(T_{u,v})}$ defines a metric space on the nodes because it satisfies four conditions:

- $d(u, v) \geq 0$
- $d(u, v) = d(v, u)$ (though this condition is not necessary for asymmetric metrics).
- $d(u, u) = 0$
- $d(u, v) + d(v, w) \geq d(u, w)$

Since we have a metric space on the nodes where the further apart two nodes are, the lower the probability of a path between them, we can make use of existing metric clustering algorithm to partition the nodes into groups. A clustering algorithm takes a set of points in a metric space and groups them in a way that tries to optimize some criteria. Examples include, k-centers which finds a set of points S of k points which minimizes the distance from any point to its closest point in S, k-means which partitions the points into k sets in a way that minimizes the variance within each group, and correlation clustering which partitions the points in a way that minimizes the sum of distances within groups minus the sum of distances across groups. Each of these clustering algorithms have good approximation algorithms when applied to points in a symmetric metric space [12, 13, 4], and some even have good approximations in an asymmetric metric space [2].

While any of these clusterings can be applied, we focus on a variant of correlation clustering. Its goal - finding clusters maximizing agreement within and minimizing agreement between clusters - fits naturally with our application. Also, Unlike most clustering algorithms, no k representing the number of clusters is provided as input, since optimizing agreement is independent of the number of clusters. In our application, the trust value from one node to another

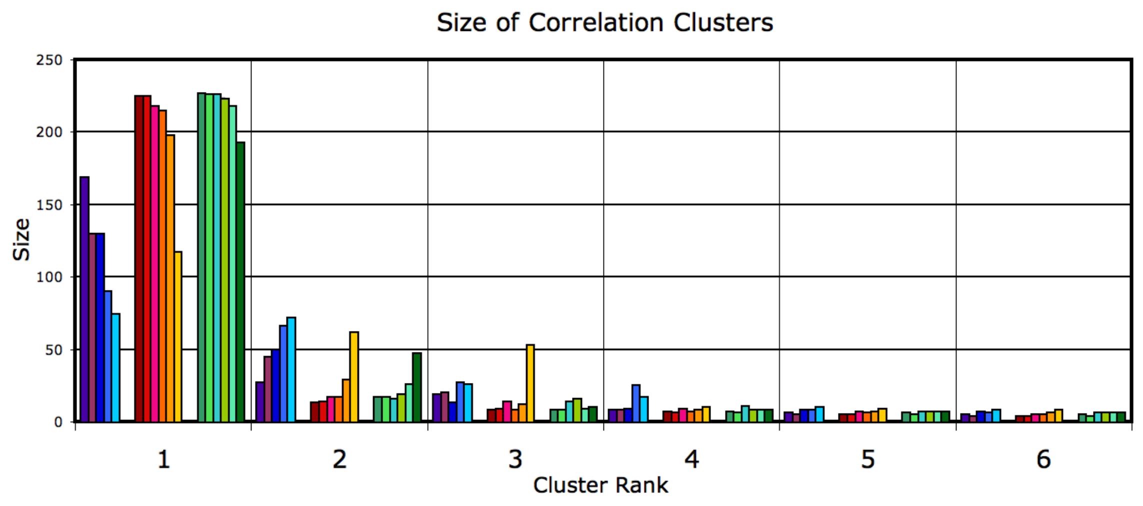


Figure 4: The size of the six largest clusters in for each iteration of the correlation clustering algorithm. The purple/blue bars indicate when the algorithm was run with a maximum radius of 1, the red/orange bars a maximum radius of 2, and the green bars a radius of 3. Six iterations are shown as the bars within each color group.

can be treated as a measure of similarity, with high trust indicating agreement and low trust indicating disagreement. Using the complete graph output from the trust inference algorithm, we can perform a correlation clustering over the graph, grouping people together who have more trust for one another.

Figure 3 shows the results from applying our algorithm to the FilmTrust network. The distance grid shows one large mutually trusting group, as well as several progressively smaller mutually trusting groups. The largest of the groups is trusted by a large portion of the network. The second largest group is well trusted by this largest group. Beyond that, the plot where $f(t) = t/15$ brings out the most difference within the groups.

Finding an optimal correlation clustering is NP-hard [7], but there are efficient constant factor approximation algorithms. We use a variant where while there are nodes left to cluster, we choose one at random to be a ”‘center”’ and create a new cluster out of all nodes within a fixed radius of this center. Since this algorithm is randomized, the output can vary from one execution to another. Thus, in clustering our inferred trust network, we ran several iterations of the algorithm to produce a representative set of clusters to work with.

In order to obtain clusters for our recommender system, we ran the correlation clustering algorithm on the network output by running our trust inference algorithm over the FilmTrust data. We used a maximum radius of 1, 2, and 3, and ran six iterations of the algorithm for each. With this dataset, the algorithm generates one very large cluster, two or three medium sized clusters, and many small clusters. Figure 4 shows the size of the six largest clusters in each iteration for all three maximum radii.

4. EXPERIMENTS AND RESULTS

To test whether or not using trust-based clusters drawn from social networks could be used to improve the quality of recommendations, we ran several experiments. In this section, we discuss the datasets, experimental design, and results that show clusters can indeed improve the accuracy of recommendations.

4.1 Recommendation Algorithms

There are many methods for generating predictive recommendations. Our goal in this work was not to create the next best recommendation algorithm but rather to demonstrate that using clusters based on trust has the potential to improve the accuracy of recommendations.

We used several basic recommendation algorithms to test our hypothesis. The first is a basic ACF algorithm computes a weighted average of ratings using the Pearson Correaltion coefficient between the recommendee and the rater as a weight. To compute a recommendation we required pairs of nodes to have at least four movies in common so we could compute a meaningful correlation coefficient. We also tested a trust-based recommender algorithm. There are a number of approaches to using trust for recommendations [9, 21, 17, 16], and we used a simple variation on user-user automated collaborative filtering (ACF), replacing the correlation coefficient with the inferred trust value computed using the method described above. Thus, people the recommendee trusts more will receive more weight. This approach has been used before and shown to produce equivalent results to ACF overall, and improved results in certain cases [9].

Both algorithms were modified to give more weight to ratings from nodes in the same cluster as the recommendee. Considering *only* ratings by nodes in the same cluster would exclude so much information that recommendations would suffer. However, if we believe that the clustered nodes are more valuable, we can give them more weight than would be afforded using only the trust value. In these experiments, gave an additional 5% weight to nodes in the same cluster. All ratings for a movie were considered and weighted by the inferred trust from the recomendee to the rater. Ratings

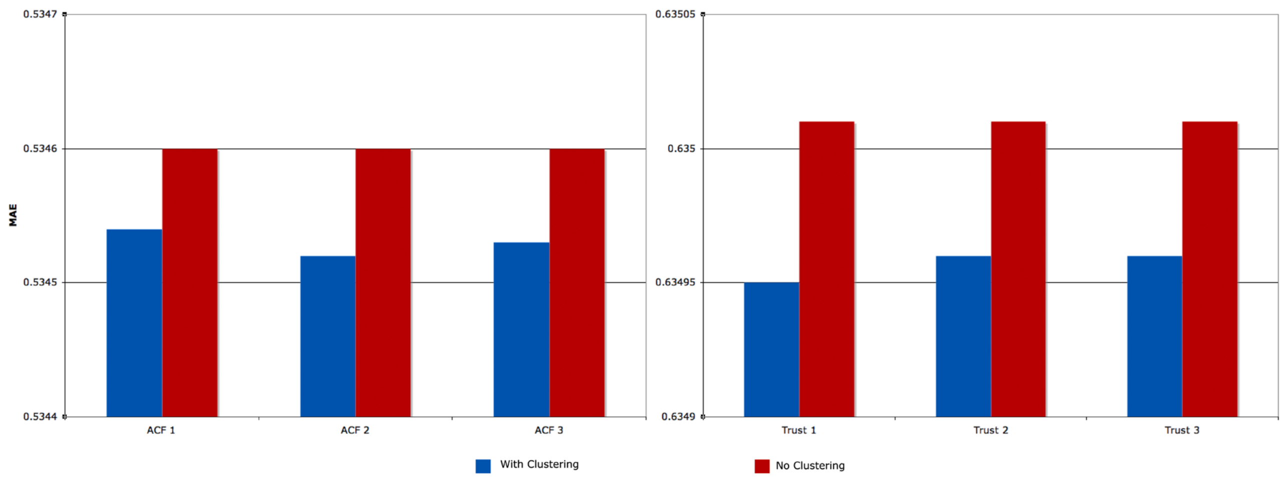


Figure 5: This illustrates the improvement in accuracy. The y-axis indicates Mean Absolute Error (MAE) so lower values are better. Blue bars indicate the results from the cluster-enhanced algorithms and red bars are the control. The numbers on the x-axis indicate the maximum radius of the clusters.

Table 2: Experimental Results of Cluster-Enhanced Recommendations. The cluster-enhanced method significantly outperforms the control in all cases.

Method	Cluster Radius	MAE (method)	MAE (control)	p-value	RMSE (method)	RMSE (control)	p-value
ACF	1	0.53454	0.53460	**0.0031**	0.70199	0.70204	**0.0273**
ACF	2	0.53452	0.53460	**0.0012**	0.70196	0.70204	**0.0022**
ACF	3	0.53453	0.53460	**0.0069**	0.70194	0.70204	**0.0004**
Trust	1	0.63495	0.63501	**0.0018**	0.82614	0.82620	**0.0076**
Trust	2	0.63496	0.63501	**0.0274**	0.82615	0.82620	**0.0493**
Trust	3	0.63496	0.63501	**0.0342**	0.82614	0.82620	**0.0356**

by nodes in the same cluster as the recommendee had their weight multiplied by 1.05. This approach was used with the Pearson corrleation-based ACF method and the trust-based recommendation.

4.2 Experimental Setup

To test our hypothesis that using the trust clusters improves the accuracy of recommendations, we ran the following process.

1. Select an iteration of the clustering algorithm to obtain clusters
2. For each user-movie pair, generate a predictive rating for the movie in two ways:
   (a) Using the standard trust-based recommendation algorithm
   (b) Using a modified trust-based recommendation algorithm that gives more weight to the nodes in the same cluster as the user as described above
3. Compare the MAE and RMSE for the two recommendation methods

This was repeated for each iteration and configuration of the clustering algorithm and for all of the cluster-enhanced algorithms described above. The clustering algorithms produced several large clusters and many very small clusters. We used only clusters with five or more nodes. To run the experiments using trust-based recommendations, it was necessary that we had an inferred trust value between the considered nodes in the network. Thus, nodes that were outside the giant component and thus had no inferred trust values were excluded. For consistency, we used the same set of nodes in all of our experiments.

4.3 Results

Our results showed that both cluster-enhanced recommendations (giving 5% extra weight to the ratings from people in the same cluster as the recommendee) offered a small but statistically significantly improvement in accuracy over the algorithms that did not consider the clusters. All significance results were computed using a Student's t-test for paired samples and were significant for $p < 0.05$.

Since the correlation clustering algorithm is randomized, we ran six iterations of the algorithm to obtain a representative sample. We ran the experiment on each iteration and then took the average rating for each user-movie pair over the six iterations to compare to the known value and judge the impact of the cluster-enhanced approach. This ensured that we could see the true impact of the approach and not be misled by an unusually good or bad clustering.

Table 4 shows the results of our different cluster-enhanced algorithms on the dataset. For both the mean absolute error (MAE) and root mean squared error (RMSE), the algorithm that took advantage of the clusters significantly outperformed the control, which ignored clusters.

4.3.1 Coverage

Clusters will not affect recommendations in all cases; it is possible that no one in the same cluster as the recommendee has rated the movie in question. In those cases, the result will be the same as the control method. However, analysis shows that at least one person who rated the movie is in the cluster approximately 70% of the time, and thus the clustering technique will have an impact.

5. DISCUSSION

These results show a small but statistically significant improvement in accuracy when correlation clusters generated from a trust network are incorporated into a recommender system. While the magnitude of the improvement is small, these results are promising when we consider that similarity-based clustering approaches typically perform significantly worse that their non-clustered counterparts.

Furthermore, we believe that the results of this approach will become more practically significant on larger social networks. Our method requires a social network with trust values and item ratings created by the people in the network. We ran these experiments on a network with 348 nodes, which is small relative to most web-based social networks[3]. We used this network because it is the only one we had access to with the necessary data; since trust values must be kept private to be effective, other datasets are not made public by the large social networks that have them. This lack of *public* data does not limit the applicability of the technique; it can be applied within privately held networks where access to trust data is not a problem. The needed data exists internally on many large networks, such as Orkut. It also can be estimated from rating data on items a pair of users have in common [11].

With the larger social networks, we will see more large clusters. Since the experimental network has one large trusted cluster, and several smaller ones, our results show a significant improvement essentially from giving less weight to nodes outside the cluster. We expect to see this effect magnified when there are more large clusters.

The fact that considering trust-clusters can improve recommendations also suggests that is has potential to help with other applications. By relying on connections in the social network, it is possible to eliminate many types of attacks or gaming in rating systems that rely on creating multiple accounts. While these accounts could all connect to one another with high trust, they would only be clustered wiht "good" users if some of these good users assigned them high trust ratings as well. However, previous work has shown that it is possible to eliminate these confused nodes from consideration [15]. These approaches together have the potential to very effectively eliminate forged ratings and reviews at the same time as they highlight those most relevant to the user.

5.1 Conclusions

Trust is strongly correlated with how similar two users are in their preferences. It reflects similarity in nuanced ways that has been shown to be useful for making recommendations. In this paper, we looked at taking trust a step further. We clustered users based on the trust between them using correlation clustering and then modified a collaborative filtering algorithm to use these clusters.

To test our approach we used a traditional Pearson correlation collaborative filtering algorithm and a recommendation algorithm that used trust for generating recommendations independently of the clusters. In both, we modified the algorithms to give extra weight to ratings from nodes in the same cluster as the user for whom the rating was being generated. We compared the accuracy of these recommendations to those made by the unmodified version of the algorithm. In both cases, our results show a small but statistically significant improvement in the accuracy of recommendations when clusters are used.

This improvement is particularly interesting since previous work on clustering, which was based on user similarity, failed to outperform non-clustered methods and often performed significantly worse. It suggests that trust captures more sophisticated information about the similarity between two people and that it is particularly useful for highlighting more relevant information in recommendation environments. We believe this effect will be magnified in bigger networks and that it has applications to limiting gaming and other attacks in online rating systems.

[3]Among the 250 social networks listed at http://trust.mindswap.org/ the mean size is over 4,600,000 and the median is 22,000.

6. ACKNOWLEDGMENTS

Supported in part by NSF ITR Award CNS-0426683 and NSF Award CNS-0626636, and DARPA.

7. REFERENCES

[1] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 2000.

[2] Aaron Archer. Two $O(log * k)$-approximation algorithms for the asymmetric k-center problem. In *Proceedings of the 8th Conference on Integer Programming and Combinatorial Optimization*, pages 1–14. Springer-Verlag, 2001.

[3] Paolo Avesani, Paolo Massa, and Roberto Tiella. Moleskiing.it: a trust-aware recommender system for ski mountaineering. *International Journal for Infonomics*, 2005.

[4] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. In *Machine Learning*, pages 238–247, 2002.

[5] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. pages 43–52, 1998.

[6] Thomas DuBois, Jennifer Golbeck, and Aravind Srinivasan. Rigorous probabilistic trust-inference with applications to clustering. In *IEEE / WIC / ACM Conference on Web Intelligence*, 2009.

[7] M.R. Garey, D.S. Johnson, et al. *Computers and Intractability: A Guide to the Theory of NP-completeness*. wh freeman San Francisco, 1979.

[8] Jennifer Golbeck. *Computing and Applying Trust in Web-based Social Networks*. PhD thesis, University of Maryland, College Park, MD, April 2005.

[9] Jennifer Golbeck. Generating predictive movie recommendations from trust in social networks. In

Proceedings of the Fourth International Conference on Trust Management, 2006.

[10] Jennifer Golbeck. The dynamics of web-based social networks: Membership, relationships, and change. *First Monday*, 12(11), 2007.

[11] Jennifer Golbeck. Trust and nuanced profile similarity in online social networks. *ACM Transactions on the Web*, in press.

[12] Dorit S. Hochbaum and David B. Shmoys. Best possible heuristic for the k-center problem. *Mathematics of Operations Research*, (2):180–184, May 1985.

[13] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. In *SCG '02: Proceedings of the eighteenth annual symposium on Computational geometry*, pages 10–18, New York, NY, USA, 2002. ACM.

[14] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, 1997.

[15] Raph Levien and Alex Aiken. Attack-resistant trust metrics for public key certification. In *7th USENIX Security Symposium*, pages 229–242, 1998.

[16] P. Massa and B. Bhattacharjee. Using trust in recommender systems: an experimental analysis. In *Proc. of 2nd Int. Conference on Trust Management, 2004.*, 2004.

[17] R. Matthew, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *Proceedings of the Second International Semantic Web Conference.*, 2003.

[18] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 263–266, New York, NY, USA, 2003. ACM.

[19] B.J. Mirza, B.J. Keller, and N. Ramakrishnan. Studying recommendation algorithms by graph analysis. *Journal of Intelligent Information Systems*, 20(2):131–160, 2003.

[20] John O'Donovan and Barry Smyth. Trust in recommender systems. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 167–174, New York, NY, USA, 2005. ACM.

[21] John O'Donovan and Barry Smyth. Trust in recommender systems. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 167–174, New York, NY, USA, 2005. ACM.

[22] B.M. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the Fifth International Conference on Computer and Information Technology*, pages 158–167, 2002.

[23] L.H. Ungar and D.P. Foster. Clustering methods for collaborative filtering. In *AAAI Workshop on Recommendation Systems*, pages 112–125, 1998.

[24] Cai-Nicolas Ziegler and Jennifer Golbeck. Investigating Correlations of Trust and Interest Similarity. *Decision Support Services*, 2006.

[25] Cai-Nicolas Ziegler and Georg Lausen. Analyzing correlation between trust and user similarity in online communities. In *Proceedings of the Second International Conference on Trust Management*, 2004.

Collaborative and Content-based Filtering for Item Recommendation on Social Bookmarking Websites

Toine Bogers
ILK / Tilburg centre for Creative Computing
Tilburg University
P.O. Box 90153, 5000 LE
Tilburg, The Netherlands
A.M.Bogers@uvt.nl

Antal van den Bosch
ILK / Tilburg centre for Creative Computing
Tilburg University
P.O. Box 90153, 5000 LE
Tilburg, The Netherlands
Antal.vdnBosch@uvt.nl

ABSTRACT

Social bookmarking websites allow users to store, organize, and search bookmarks of web pages. Users of these services can annotate their bookmarks by using informal tags and other metadata, such as titles, descriptions, etc. In this paper, we focus on the task of item recommendation for social bookmarking websites, i.e. predicting which unseen bookmarks a user might like based on his or her profile. We examine how we can incorporate the tags and other metadata into a nearest-neighbor collaborative filtering (CF) algorithm, by replacing the traditional usage-based similarity metrics by tag overlap, and by fusing tag-based similarity with usage-based similarity. In addition, we perform experiments with content-based filtering by using the metadata content to recommend interesting items. We generate recommendations directly based on Kullback-Leibler divergence of the metadata language models, and we explore the use of this metadata in calculating user and item similarities. We perform our experiments on three data sets from two different domains: Delicious, CiteULike and BibSonomy.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval; H.3.4 Systems and Software

General Terms

Algorithms, Measurement, Performance, Experimentation

Keywords

Recommender systems, social bookmarking, folksonomies, collaborative filtering, content-based filtering

1. INTRODUCTION

Recommender systems belong to a class of personalized information filtering technologies that aim to identify which items in a catalog might be of interest to a particular user. Recommendations can be made using a variety of information sources related to both the user and the items: past user preferences, purchase history, demographic information, item popularity, the metadata characteristics of the products, etc. *Social bookmarking websites*, with their emphasis on open collaborative information access, offer an ideal scenario for the application of recommender systems technology. They allow users to manage their favorite bookmarks online through a web interface and, in many cases, allow their users to tag the content they added to the system with keywords. The aggregation of these tags, the *folksonomy*, is an extra annotation layer connecting users and items. The underlying application then makes all information sharable among users. The success of such websites partly depends on how the connections between users, items, and tags are exploited.

Social bookmarking websites also offer possibilities for attaching item-specific metadata to each item, such as the item's title or summary. This additional metadata could also be used to support the recommendation process. Using item metadata to boost performance is not new in recommender systems research, but typically such content-related information is attached to the items, and is therefore the same across all users [23]. On the other hand, the tags assigned to items are specific to the user who added them. We know of no approaches to item recommendation for social bookmarking that investigate the use of such metadata.

In this paper we focus on the question how we can make use the information represented by the folksonomy and the item metadata to boost the performance of traditional collaborative filtering algorithms. We make the following contributions. First, we examine different ways of extending the standard nearest-neighbor algorithm with information about tagging behavior. Second, we explore how best to use the metadata for recommendation by proposing four different algorithms. Finally, we perform our experiments on publicly available data sets with standardized evaluation metrics to promote verifiability. The remainder of this paper is structured as follows. We start by introducing our data sets and experimental setup in the next section. In Section 3 we describe and compare different CF algorithms that operate on the folksonomy to generate recommendations. Section 4 we describe how we exploit the metadata in our data sets to generate item recommendations. We describe the related work in Section 5 and conclude in Section 6.

2. METHODOLOGY

2.1 Data Sets

We base our experiments on four data sets that were collected from three different social bookmarking websites with different characteristics: CiteULike, BibSonomy, and Delicious. Two data sets correspond to the domain of Web page bookmarks (Delicious and

BibSonomy) and the other two cover the domain of scientific articles (Delicious and BibSonomy).

CiteULike is a social bookmarking service that allows its users to add their academic reference library to an online profile[1]. Articles can be stored with their metadata, abstracts, and links to the papers at the publishers' sites. Users can also add personal comments and tags. CiteULike offers daily dumps of their core database. We used the dump of November 2, 2007 as the basis for our experiments. A dump contains all information on which articles were posted by whom, with which tags, and at what point in time. It does not, however, contain any other item metadata, so we crawled this ourselves from the CiteULike website using the article IDs. Articles are annotated using the standard BibTeX-like fields, such as title, author names, page numbers, publisher information, etc.

BibSonomy is a social bookmarking service for sharing Web page bookmarks and reference lists of scientific articles[2]. Items are stored and represented by their BibTeX metadata representations. These can include abstracts and links to the papers at the publishers' websites. Users are able to tag their bookmarked content and use these tags to browse and discover related references [13]. BibSonomy's creators organized the 2008 ECML/PKDD Discovery Challenge which focused on social bookmarking, and released the BibSonomy data set to the public in May 2008 as part of this challenge[3]. The organizers made a snapshot of the BibSonomy system consisting of all resources posted to BibSonomy between its inception in 2006 and March 31, 2008. It includes the same type of article metadata as we collected for CiteULike. The distinction between bookmarks and BibTeX records is also made in this snapshot. We therefore split this data dump into a data set containing only web bookmarks (Bibsonomy Bookmarks), and a data set containing only scientific articles (Bibsonomy Articles).

Delicious is a social bookmarking service for storing, sharing, and discovering web bookmarks. It allows its users to manage and tag URLs of web pages[4]. Unlike CiteULike and BibSonomy, Delicious does not offer data dumps of their databases, so we gathered our data set by crawling a subset of the Delicious website. Because of our focus on the task of item recommendation for users, our aim was to collect a balanced, unbiased set of user profiles, i.e. the complete set of bookmarks a user had posted to Delicious. From an earlier breadth-first crawl of Delicious we obtained a list of 300,000 users. We randomly selected around 18,000 of these users to match the size of our CiteULike data set, and crawled their entire profiles.

2.1.1 *Data set filtering*

It is common practice in recommender system evaluation to select realistic subsets of the data sets used to ensure that reliable recommendations can be generated. This also allows for a fair comparisons of different recommendation algorithms [11]. This is typically done by filtering out users or items whose profile size or popularity falls below a certain threshold. We follow this procedure in our preparation of the data sets as well. We only retain the users who have added 20 items or more to their personal profile. In addition, we filter out all items that occur only once, as well as all untagged posts. We were able to identify and filter out most of the spam content in the CiteULike and BibSonomy data sets. We refer the reader to [5] for more details about this process. Table 1 lists the statistics of our four data sets after filtering.

[1]http://www.citeulike.org/
[2]http://www.bibsonomy.org/
[3]http://www.kde.cs.uni-kassel.de/ws/rsdc08/
[4]http://www.delicious.com/

2.2 Experimental Setup & Evaluation

In order to evaluate and compare different recommender algorithms, we need a proper framework for experimentation and evaluation. Recommender systems evaluation—and the differences with IR evaluation—has been addressed by, among others, [11]. We evaluate the "Find Good Items" task, also known as Top-N recommendation, where users are provided with a ranked list of recommended items based on their personal profile. We divide each data set into a training and test set by randomly selecting 10% of the users to be in our test set. Final performance is evaluated on this 10% by withholding 10 items from each of these so-called *active users*, and using the remaining profile items together with the training set to generate the recommendations for those 10%. If the withheld items are predicted at the top of the ranked result list, then the algorithm is considered to do perform well. To prevent overestimation when optimizing algorithm parameters, we use 10-fold cross-validation. We subdivide our training set into 10 folds and use these for 10-fold cross-validation of our parameter optimization. For each fold, 10 items are withheld from the test fold users to be retrieved by the recommendation algorithm. The final values for our evaluation metric on the withheld items are then macro-averaged over the 10 folds.

In our evaluation, we adopt an IR perspective by treating each of the users as a separate query or topic. The 10 withheld items for each user constitute the items for which we have relevance judgments. Herlocker et al. [11] assess the usefulness of different metrics for different types of recommendation tasks. For the Top-N recommendation task, they find that metrics that take into account the ranking of the items are most appropriate. We therefore evaluate our algorithms using Mean Average Precision (MAP), which is defined as the average of the Average Precision values calculated over each relevant retrieved item. For determining significance of differences between runs, we use a two-tailed paired Student's t-test. We report on significant differences against the best baseline runs using $^{\vartriangle}$ (and $^{\triangledown}$) for $\alpha = .05$ and $^{\blacktriangle}$ (and $^{\blacktriangledown}$) for $\alpha = .01$.

3. FOLKSONOMIC RECOMMENDATION

We start by establishing some notation and definitions of the task at hand, based in part on notation by [9]. In the social bookmarking setting, users post items to their personal profiles and can choose to label them with one or more tags. We define a folksonomy to be the tripartite graph that emerges from this collaborative annotation of items. The resulting ternary relations that make up the tripartite graph can be represented as a 3D matrix of users, items, and tags. Figure 1 illustrates this view. We refer to the 3D matrix as $\mathbf{D}(u_k, i_l, t_m)$. Here, each element $d(k, l, m)$ of this matrix indicates if user u_k (with $k = \{1, \ldots, K\}$) tagged item i_l (with $l = \{1, \ldots, L\}$) with tag t_m (with $m = \{1, \ldots, M\}$), where a value of 1 indicates the ternary relation is present in the folksonomy.

In conventional recommender systems, the user-item matrix contains ratings information. These ratings can be *explicit*, when they are entered directly by the user, or *implicit*, when they are inferred from user behavior. In our case we have implicit, unary ratings where all items that were added by a user receive a rating of 1. We extract this ratings matrix $\mathbf{R}(u_k, i_l)$ for all user-item pairs directly from the tripartite graph. We denote its individual elements by $x_{k,l} = \{1, \emptyset\}$. Each user is represented in this matrix as its user profile row vector $\mathbf{u}_k$, which lists the items that user added to his or her profile. Items are represented by the column vectors of $\mathbf{R}$ which represent the item profile vectors $\mathbf{i}_l$ that contain all users that have added that item. As shown in Figure 1, we can also extract a user-item matrix from $\mathbf{D}$ by aggregating over the tag dimension. We then

Table 1: Statistics of the filtered versions of our four data sets.

	bookmarks		articles	
	Delicious	BibSonomy	CiteULike	BibSonomy
# users	1,243	192	1,322	167
# items	152,698	11,165	38,419	12,982
# tags	42,820	13,233	28,312	5,165
# posts	238,070	29,096	84,637	29,720
user-item sparsity	99.8746	98.6427	99.8334	98.6291
avg # items per user	191.5	151.5	64.0	178.0
avg # users per item	1.6	2.6	2.2	2.3
avg # tags per user	192.1	203.3	57.3	79.2
avg # users per tag	5.6	2.9	2.7	2.6
avg # tags per item	4.8	8.4	5.3	3.1
avg # items per tag	17.0	7.1	7.3	7.7

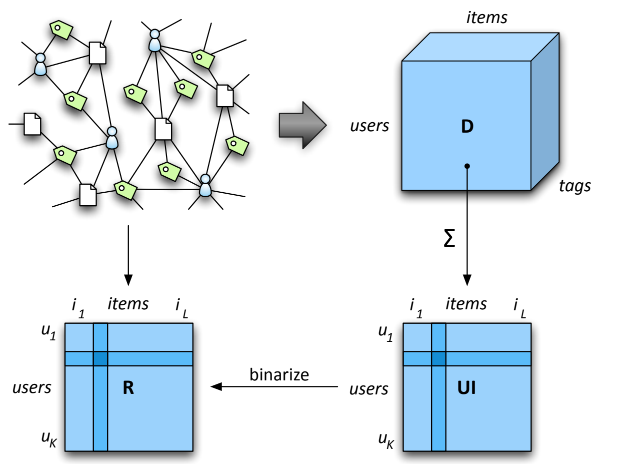


Figure 1: Representing the folksonomy graph as a 3D matrix. The ratings matrix R is derived from the tripartite graph itself, and directly represents what items were added by which users. Aggregation over the tag dimension of D gives us matrix UI, containing the tag counts for each user-item pair. We can obtain R by binarizing the values in UI..

obtain the $K \times L$ user-item matrix $\mathbf{UI}(u_k, i_l) = \sum_{m=1}^{M} \mathbf{D}(u_k, i_l, t_m)$, specifying how many tags each user assigned to each item. Because we filtered our data sets to include only tagged content, our ratings matrix **R** is the same as a binary version of **UI**. Similar to the way we defined **UI** we can also aggregate the content of **D** over the user and the item dimensions. We define the $K \times M$ user-tag matrix $\mathbf{UT}(u_k, t_m) = \sum_{l=1}^{L} \mathbf{D}(u_k, i_l, t_m)$, specifying how often each user used certain tag to annotate his or her items. Individual elements of **UT** are denoted by $y_{k,m}$. We define the $L \times M$ item-tag matrix $\mathbf{IT}(i_l, t_m) = \sum_{k=1}^{K} \mathbf{D}(u_k, i_l, t_m)$, indicating how many users assigned a certain tag to an item. Individual elements of **IT** are denoted by $z_{l,m}$. We also define binary versions of **UT** and **IT** as $\mathbf{UT}_{binary}$ and $\mathbf{IT}_{binary}$. The row vectors of the **UT** and **IT** matrices represent the user tag profiles $\mathbf{d}_k$ and item tag profiles $\mathbf{f}_l$ respectively. They list what tags have been assigned by a user to his items, or to an item by its users. Formally, the goal of each of the recommendation algorithms discussed in this paper is to produce a ranking of all items l that are not yet in the profile of the active user u_k (i.e., $x_{k,l} = \emptyset$). To this end, we predict a score $\widehat{x}_{k,l}$ for each item that represents the likelihood of that item being relevant for the active user. The final recommendations for a user are generated by ranking all items i_l by their predicted score $\widehat{x}_{k,l}$.

3.1 Baseline Recommendation Algorithms

A common and well-understood source of information for recommendation is usage patterns of adding and rating items. The class of algorithms that exploit such patterns for recommendation purposes are called Collaborative Filtering algorithms (CF). In this paper we focus on using and extending the k-Nearest Neighbor (k-NN) algorithm. We pick the k-NN algorithm because it is a well understood algorithm that can intuitively be extended to include other information in addition to transaction patterns [7, 10]. There are two flavors of the k-NN algorithm for CF: *user-based filtering* and *item-based filtering*. In user-based filtering, we locate the users most similar to the active users, and look among their items for new recommendations. In item-based filtering, we locate the most similar items for each of the active user's items and aggregate these into a list of predicted items.

User-based Filtering.

In the first step of user-based filtering we calculate the similarities between pairs of users to identify the most similar users for an active user. Many different similarity metrics have been proposed and evaluated over time, such as Pearson's correlation coefficient and cosine similarity [6]. We use the cosine similarity in our experiments as it has often been used successfully on data sets with implicit ratings [6, 19]. We calculate the cosine similarity between the active user u_k and another user u_a on the user profile vectors $\mathbf{u}_k$ and $\mathbf{u}_a$ as $sim_{cosine}(u_k, u_a) = \frac{\mathbf{u}_k \cdot \mathbf{u}_a}{\|\mathbf{u}_k\| \, \|\mathbf{u}_a\|}$.

The next step in user-based filtering is to determine the top N similar users (or items) for user u_k. We denote this set as the Set of Similar Users $SSU(u_k)$, which are the top N users of the set of all users u_a, ranked by their cosine similarity. For each user u_a, we only consider those items that u_a added to his profile ($x_{a,l} \neq \emptyset$). Using this set of nearest neighbors we generate the final prediction scores $\widehat{x}_{k,l}$ for each unseen item i_l as $\widehat{x}_{k,l} = \sum_{u_a \in SSU(u_k)} sim_{cosine}(u_k, u_a)$. Here, the predicted score of an item i_l is the sum of the similarity values (between 0 and 1) of all N nearest neighbors that actually added item i_l (i.e. $x_{a,l} \neq \emptyset$).

A recurring observation from the literature about CF algorithms is that universally liked items are not as useful for capturing the similarity between users as less common items, see e.g. [6]. We therefore perform two runs: the 'vanilla' base run described above (u-bin-sim) and a run where the values in the user vectors are weighted by the *inverse user frequencies* of the items (u-bin-idf-sim).

Item-based Filtering.

The item-based k-NN algorithm operates analogously to the user-based filtering algorithm [19]. Instead of comparing users directly,

Table 2: Results of the folksonomic recommendation runs. Reported are the MAP scores as well as the optimal number of neighbors *N*. Best-performing runs for each data group of approaches are printed in bold. Best-performing tag overlap runs for both user-based and item-based are printed in bold. The percentage difference between the best baseline CF runs and the best tag overlap runs are indicated after each type.

Runs	bookmarks				articles			
	BibSonomy		Delicious		BibSonomy		CiteULike	
	MAP	*N*	MAP	*N*	MAP	*N*	MAP	*N*
Best baseline UB CF run	**0.0277**	13	**0.0046**	15	**0.0865**	4	**0.0757**	15
	(u-bin-idf-sim)		(u-bin-sim)		(u-bin-sim)		(u-bin-idf-sim)	
ut-jaccard-sim	0.0070	8	0.0015	11	0.0459	6	0.0449$^{\blacktriangledown}$	5
ut-dice-sim	0.0069$^{\triangledown}$	6	0.0007$^{\blacktriangledown}$	6	0.0333$^{\triangledown}$	4	0.0439$^{\blacktriangledown}$	2
ut-bin-sim	0.0102	5	0.0017	11	0.0332$^{\triangledown}$	4	0.0452$^{\blacktriangledown}$	3
ut-tf-sim	0.0069$^{\triangledown}$	2	0.0015$^{\triangledown}$	25	0.0368	4	0.0428$^{\blacktriangledown}$	8
ut-tfidf-sim	0.0018$^{\triangledown}$	6	0.0013$^{\triangledown}$	17	0.0169$^{\blacktriangledown}$	2	0.0400$^{\blacktriangledown}$	2
% Change over best UB CF run	-63.2%		-63.0%		-46.9%		-40.7%	
Best baseline IB CF run	0.0244	34	0.0027	25	0.0737	49	0.0887	30
	(i-bin-idf-sim)		(i-bin-sim)		(i-bin-idf-sim)		(i-bin-idf-sim)	
it-jaccard-sim	**0.0370**	3	0.0083$^{\vartriangle}$	21	0.0909	6	0.0810	14
it-dice-sim	0.0317	2	0.0089$^{\vartriangle}$	25	0.0963	8	**0.0814**	8
it-bin-sim	0.0334	2	**0.0101**$^{\vartriangle}$	23	0.0868	5	0.0779	10
it-tf-sim	0.0324	4	0.0100$^{\vartriangle}$	11	0.0823	4	0.0607$^{\blacktriangledown}$	17
it-tfidf-sim	0.0287	8	0.0058	7	**0.1100**	7	0.0789	21
% Change over best IB CF run	+51.6%		+274.1%		+49.3%		-8.2%	
% Change over best CF run	+33.6%		+119.6%		+27.2%		-8.2%	

we try to identify the best recommendations for each of the items in an active user's profile. In other words, for item-based filtering we calculate the similarities between the test items of the active user u_k and the other items that user has not yet added (so $x_{k,l} = \emptyset$). Similarity between two items i_l and i_b is calculated on the item profile vectors $\mathbf{i}_l$ and $\mathbf{i}_b$ as $sim_{cosine}(i_l, i_b) = \frac{\mathbf{i}_l \cdot \mathbf{i}_b}{||\mathbf{i}_l|| \, ||\mathbf{i}_b||}$. Next, we identify the top N most similar items for each of the active user's items i_l separately. We define this neighborhood as the Set of Similar Items $SSI(i_l)$, where we select the top N of all items not already in the active user's profile, ranked on their cosine similarity $sim_{cosine}(i_l, i_b)$ to item i_l. Using this set of nearest neighbors we generate the final prediction score $\widehat{x_{k,l}}$ for each unseen item i_l as $\widehat{x_{k,l}} = \sum_{i_b \in SSI(i_l)} sim_{cosine}(i_l, i_b)$. Here, the predicted score is the sum of the similarity values (between 0 and 1) of all the most similar items that were added by user u_k (i.e. $x_{k,b} \neq \emptyset$). Analogous to user-based filtering, we can also suppress the influence of the most 'popular' users, i.e. users that have added a disproportionately large number of items to their profile, such as bots or spam users. We refer to the item-based filtering runs weighted with the *inverse item frequency* as i-bin-idf-sim and to the unweighted runs as i-bin-sim.

3.2 Tag Overlap Similarity

The folksonomies present in our four data sets each constitute an extra layer of connections between user and items. We exploit this extra layer for determining another type of similarity between users or items. For instance, users that assign many of the same tags can be seen as similar, and items that are often assigned the same tags can also be seen as similar.

We restrict ourselves to comparing three common similarity metrics: Jaccard overlap, Dice's coefficient, and the cosine similarity. We use these similarities as the basis for item recommendation. The only difference between this approach and the standard CF algorithm is in the first step, where the similarities are calculated. For user-based filtering, we calculate tag overlap on the **UT** matrix or on the binarized version $\mathbf{UT}_{binary}$, depending on the metric. Both the Jaccard overlap and Dice's coefficient are set-based metrics, which means we calculate them on the binary vectors from the $\mathbf{UT}_{binary}$ matrix. The Jaccard Overlap $sim_{Jaccard}(d_k, d_a)$ between two users d_k and d_a is defined as $\frac{|\mathbf{d}_k \cap \mathbf{d}_a|}{|\mathbf{d}_k \cup \mathbf{d}_a|}$. Dice's coefficient $sim_{Dice}(d_k, d_a)$ is defined as $\frac{2|\mathbf{d}_k \cap \mathbf{d}_a|}{|\mathbf{d}_k| + |\mathbf{d}_a|}$. We refer to the user-based runs with the Jaccard overlap and Dice's coefficient as ut-jaccard-sim and ut-dice-sim respectively. The cosine similarity is calculated in three different ways. First, we calculate it on the regular tag count vectors $\mathbf{d}_k$ and $\mathbf{d}_a$ from **UT** as ut-tf-sim, and on the binary vectors from the $\mathbf{UT}_{binary}$ matrix as ut-bin-sim. In addition, we also experiment with *idf*-weighting of the tags in the user tag count vectors as we did before. We refer to this run as ut-tfidf-sim. The item-based versions of these similarity metrics are calculated on the **IT** and $\mathbf{IT}_{binary}$ matrices. We refer to these five item-based runs as it-jaccard-sim, it-dice-sim, it-bin-sim, it-tf-sim, and it-tfidf-sim.

3.3 Results & Discussion

Table 2 compares the results of our baseline CF runs that employ usage-based similarities to the runs that use overlap in tagging behavior as a source of user and item similarity. We see that the user-based filtering baseline outperforms item-based filtering on three of four data sets; only on CiteULike does item-based filtering work better, where this difference is also statistically significant ($p < 0.05$). The other differences between user-based and item-based filtering are not significant. There appears to be no clear or statistically significant advantage to applying *idf*-weighting to the profile vectors. An explanation for the advantage of user-based filtering is that, according to Table 1, the average number of items per user is much higher than the average number of users per item. Calculating a meaningful overlap between user profile vectors could therefore be more robust than between item profile vectors.

As for the results with tag overlap, we observe that item similarities based on tag overlap work well for item-based filtering, as three of our four data sets show considerable improvements over the best CF baseline runs. Performance increases range from 27% on BIBSONOMY ARTICLES to almost 120% on DELICIOUS, but these are only statistically significant on the DELICIOUS data set. We see the opposite trend for user-based filtering, where tag overlap results in significantly worse scores for almost all variants on all data sets, with performance decreases ranging from 40% to 63%. This means that using tag overlap in item-based filtering makes item-based filtering outperform user-based filtering on all four data sets. We believe that it is the reduction in sparsity from using tag overlap that causes this difference in performance. On average, the number of tags assigned to an item is 2.5 times higher than the number of users who have added the item. This means that, on average, item profile vectors from the **IT** matrix are less sparse than item profile vectors from the **UI** matrix, making the possibility of overlap between vectors more likely. Using more values in the similarity calculation leads to a better estimate of the real similarity between two items.

For user-based filtering this difference is not as well-pronounced: in some data sets users have more items than tags on average, and more tags than items in other data sets. This explains why we do not see the same performance increase for the user-based filtering runs based on tag overlap. The results of the different tag overlap metrics tend to be close together and differences between them are not statistically significant. Even though the best-performing similarity metrics are dependent on the data set, we do see that the metrics operating on the binary vectors from the $\mathbf{UT}_{binary}$ and $\mathbf{IT}_{binary}$ matrices are consistently among the top performers.

In general, it appears that bookmark recommendation is more difficult than article recommendation. We believe this is due to a difference in topic specificity. The Delicious and Bibsonomy Bookmarks data sets cover bookmarks of web pages, which encompass many more topics than scientific articles do. Users of Delicious and Bibsonomy Bookmarks can be expected to have more different topics in their profile, making it more difficult to recommend new, interesting bookmarks based on their profiles. We see evidence for this explanation in the average number of unique tags per user: 203.3 and 192.1 for Bibsonomy Bookmarks and Delicious respectively, which is markedly higher than the 79.2 and 57.3 for Bibsonomy Articles and CiteULike.

4. RECOMMENDATION USING METADATA

In addition to the folksonomic structure of the underlying network, social bookmarking services also offer users the possibility to annotate the content of their items with metadata. In this section we investigate the role such metadata can play in recommending interesting bookmarks or references. We propose two different approaches: *content-based filtering* and *hybrid filtering*. Before we move on to describing these in Sections 4.1 and 4.2, we first take a closer look at the metadata we have available.

In our approach we distinguish between *item-intrinsic* and *item-extrinsic* metadata. Item-intrinsic metadata fields relate directly to the content of the item being annotated. For the two data sets dealing with web bookmarks these include `DESCRIPTION`, `TAGS`, `TITLE`, and `URL`. The two scientific article data sets contain the additional intrinsic fields `ABSTRACT` ,`AUTHOR`, `BOOKTITLE`, `EDITOR`, `JOURNAL`, `NOTE`, and `SERIES`. The intuition behind assigning metadata fields to the item-intrinsic category is that these fields can be used as stand-alone sources for recommending other content. For instance, given a certain paper from a user's profile, papers with similar abstracts, papers written by the same author, or papers published at the same workshop are likely to be relevant recommendations. In contrast, item-extrinsic metadata fields—such as `MONTH` or `PAGES`—cannot be used to directly generate appropriate recommendations. We performed experimental runs using the metadata of each of our intrinsic fields separately. In addition, we experimented with the combination of all intrinsic fields, and with runs that combined all intrinsic and extrinsic fields, resulting in a total of 34 runs per algorithm. We did not test the extrinsic fields separately. Due to space restrictions we only report the results of the best runs for each algorithm.

4.1 Content-based Filtering

The first approach we propose is content-based filtering where the focus is on properly representing the content in our social bookmarking data sets. Based on these representations our aim is to construct an interest profile of an active user, and then use this profile to rank-order the unseen items by similarity to the profile, thereby approximating possible interest in those items. Figure 2 illustrates two different algorithms we propose for content-based filtering: *profile-centric matching* and *post-centric matching*.

The difference between our two content-based filtering algorithms is the level of aggregation. In our profile-centric matching approach, we collate all of a user's assigned metadata into a single user profile. The intuition here is that by aggregating all of the metadata assigned by a user we can completely capture his or her interests. Similarly, we construct item profiles that collate all of the metadata assigned to those items by all users in the training set. We then match the active user profiles against the item profiles on similarity to produce a ranking of all items, as illustrated in the top half of Figure 2. After removing the items already in the active user's profile, we are left with the final rank-ordered list of recommendations.

In contrast, post-centric matching operates on the level of individual posts. We match each of an active user's posts separately against all the other posts of unseen items in the training set, as illustrated in the bottom half of Figure 2. This leads to a list of matching posts in order of similarity for each of the active user's posts. Since retrieval scores are not directly comparable between runs, we normalize the original similarity scores sim_{org} into [0, 1] using the maximum and minimum similarity scores sim_{max} and sim_{min} according to $sim_{norm} = \frac{sim_{org} - sim_{min}}{sim_{max} - sim_{min}}$. We then calculate a rank-corrected sum of similarity scores for each item i_l according to $score(i) = \sum \frac{sim_{norm}(i_l)}{log(rank(i_l)) + 1}$. The final list of recommendations ranks every unseen item i_l by their rank-corrected score $score(i_l)$.

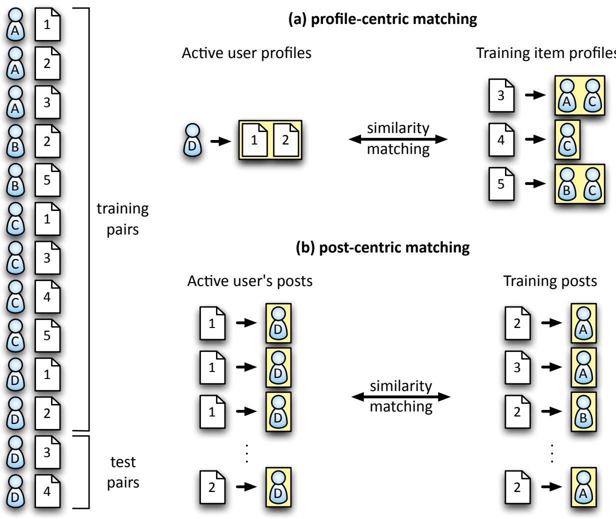


Figure 2: Visualization of our two content-based filtering approaches to item recommendation for a small toy data set.

In both content-based filtering algorithms, we approach the recommendation process from an IR perspective and restrict ourselves to measuring textual similarity. We use the open-source retrieval toolkit Lemur to calculate the similarities between the different user and item profiles. The Lemur toolkit[5] implements different retrieval methods based on language modeling [20]. Preliminary experiments comparing language modeling with the OKAPI model and a tf·idf approach suggested a language modeling approach with Jelinek-Mercer smoothing as the best-performing retrieval method. The language models we used are maximum likelihood estimates of the unigram occurrence probabilities. We filter stopwords using the SMART stopword list and do not perform stemming.

4.2 Hybrid Filtering

In addition to focusing solely on using the metadata for recommendation, we also consider a hybrid approach that joins content-based filtering and CF, in the hope of combining the best of both worlds. Many different combination methods have been proposed in earlier work [7]. In our *hybrid filtering* approach we view metadata in social bookmarking systems as another source of information for locating the nearest neighbors of users and items in CF algorithms. Figure 3 illustrates this approach. Instead of only looking at the overlap in items that two users have in common when calculating user similarities, we can use the overlap in the metadata applied to items to determine the most similar neighbors. Users that describe their profile items using the same terminology are likely share the same interests, making them a good source of recommendations. This is similar to the way we used the tag clouds of users and items to calculate similarity between users and items in the previous section. The user and item similarities we derive in this way are then plugged into the standard memory-based CF algorithms as described in Section 3.1. The resulting algorithm is a feature-augmented hybrid of CF and content-based filtering.

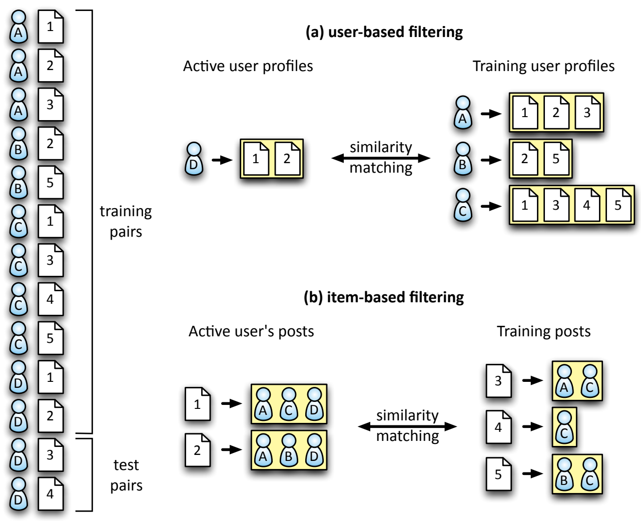


Figure 3: Visualization of our two hybrid filtering approaches to item recommendation for a small toy data set.

Hybrid filtering also consists of two steps: (1) calculating the most similar neighbors of the active user or his items, and (2) using those neighbors to predict item ratings for the active user. The latter prediction step is performed in the same manner as described earlier in Section 3.1. As in CF, with our hybrid filtering algorithms we also distinguish between user-based filtering, where we generate recommendations by determining the most similar users, and item-based filtering, where we recommend the items most similar to the items in the active user's profile. Like in Section 4.1, we approach the first step from an IR perspective and calculate the textual similarities between users or items. For each user and each item we generate user and item profile representations, constructed as follows. All of the metadata text of a user's posts is collated into a single "user profile" for that user. Similarly, for the item-based approach we create item profiles for each item by concatenating all of the metadata assigned to that item by all the users who have the item in their profile. This means that items are represented by their aggregated community metadata and not just by a single user's data. Again, we used the open-source retrieval toolkit Lemur to calculate the similarities between the different metadata representations, with the same experimental settings as described in Section 4.1.

[5]Available at http://www.lemurproject.org

4.3 Results & Discussion

Table 3 contains the best runs for each of the four metadata-based algorithms, as well as our best CF run from Section 3. What we see, is that on three out of four data sets a recommendation algorithm that uses metadata is better than the best CF run using data from the folksonomy. All of our best metadata runs use the combined metadata fields. On their own, each field can be seen as an imperfect representation of the items and users, but combined they alleviate each others weak points and better represent the content than they do separately. Only on the Delicious data set do all metadata-based approaches perform significantly worse than the CF runs. Unfortunately, we do not have an explanation for this. When we compare the metadata-based approaches with each other, we see that most differences are not statistically significant. On the Bibsonomy Articles data set, the item-centric hybrid filtering approach is significantly better than the user-centric approach ($p < 0.05$). On the CiteULike data set, the profile-centric approach also significantly outperforms the post-centric and user-centric approaches.

In general, we observe that the profile-centric approach tends to outperform the post-centric approach on three of our four data sets. This improvement is statistically significant for the CiteULike data set with an improvement of 117% ($p < 10^{-6}$). Only on the Delicious data set does post-centric matching perform significantly better ($p < 0.05$). This advantage of the profile-centric approach is strongest on the article data sets where the profile-centric approach performs best for 75% of the all runs with different fields. In the case of hybrid filtering, the item-centric approach outperforms the user-centric approach on three of our four data sets. On the CiteULike and Bibsonomy Articles data sets these differences are statistically significant and especially large at 268% ($p < 0.05$) and 112% respectively ($p < 0.01$).

While we do not have room to report the results of all individual intrinsic field runs, we can report on our general findings. For all four approaches, the best-performing single fields are AUTHOR, DESCRIPTION, TAGS, and TITLE, which provide the best individual results on all four data sets for all approaches. This is not surprising, as these fields are the least sparsely filled of all the intrinsic fields. In addition, these four fields are also aimed directly at describing the content of the items, more so than the conference or journal titles or the editors. Another interesting observation is that the TITLE field served as a better source of user and item similarity on the article data sets than on the bookmark data sets. This is because titles assigned to bookmarks are more variable than titles assigned to scientific articles, leading to this performance gap.

Table 3: Results comparison of the best metadata-based runs with our best folksonomic CF runs. Reported are the MAP scores as well as the optimal number of neighbors N where applicable. The best-performing runs are printed in bold. The percentage difference between our best meta-data approaches and the best CF runs is listed in the bottom row.

Runs	bookmarks				articles			
	BibSonomy		Delicious		BibSonomy		CiteULike	
	MAP	N	MAP	N	MAP	N	MAP	N
Best CF run	0.0370	3	**0.0101**	23	0.1100	7	0.0887	30
	(it-jaccard-sim)		(it-bin-sim)		(it-tfidf-sim)		(i-bin-idf-sim)	
Profile-centric filtering	**0.0402**	-	0.0014$^{\blacktriangledown}$	-	0.1279	-	**0.0987**	-
	(all intrinsic)		(TITLE)		(all intrinsic)		(all extrinsic)	
Post-centric filtering	0.0259	-	0.0036$^{\triangledown}$	-	0.1190	-	0.0455$^{\blacktriangledown}$	-
	(all intrinsic)		(TAGS)		(all intrinsic)		(all extrinsic)	
User-centric hybrid filtering	0.0218	2	0.0039$^{\triangledown}$	13	0.0410	2	0.0608$^{\blacktriangledown}$	2
	(URL)		(all intrinsic)		(TITLE)		(TITLE)	
Item-centric hybrid filtering	0.0399	11	0.0017$^{\blacktriangledown}$	8	**0.1510**	21	0.0746	21
	(TAGS)		(all intrinsic)		(all intrinsic)		(TAGS)	
% Change over best CF run	+8.6%		-61.3%		+37.2%		+11.3%	

5. RELATED WORK

5.1 Folksonomic Recommendation

One of the first approaches to recommendation for social bookmarking websites was presented by [12], who proposed a graph-based algorithm called *FolkRank*. They generated 2D projections of the tripartite graph and proposed a random walk model similar to PageRank [17] that uses the steady state node probabilities as the basis for ranking their recommendations. Clements et al. [9] also proposed a random walk model for item recommendation, but combine ratings information with tagging information into a single model. They also incorporated self-transition probabilities in the matrix, and used the walk length as an algorithm parameter.

There have also been several adaptations of memory-based algorithms that include information about the tags assigned by users to items. Approaches that resemble our use of tag overlap for calculating similarities between users and items include [2], [16], and [22]. Tso-Sutter et al. [23] proposed a novel tag-aware k-NN algorithm for item recommendation. When calculating the user and item similarities they include the tags as additional items and users respectively. They then calculate cosine similarity on these extended profile vectors and fuse together the predictions of the user-based and item-based filtering runs. This fused model is able to effectively capture the relationship between users, items, and tags.

Symeonidis et al. [21] were among the first to propose a model-based approach to incorporating tagging information in recommendation. They propose an item recommendation approach that performs tensor decomposition on the third-order folksonomy tensor. By performing higher-order SVD, they approximate weights for each user-item-tag triple in the data set, which can then be used to support item recommendation. They compared their algorithm to the FolkRank algorithm [12], and found that tensor decomposition outperforms the latter. Wetzker et al. [24] took a Probabilistic Latent Semantic Analysis (PLSA) approach, which assumes a latent lower dimensional topic model. They extended PLSA by estimating the topic model from both user-item occurrences as well as item-tag occurrences, and then linearly combined the output of the two models. They tested their approach on a large crawl of Delicious, and found that it significantly outperforms a popularity-based algorithm.

5.2 Exploiting Metadata for Recommendation

While a significant amount of research has focused on Collaborative Filtering for recommending interesting items, there has also been considerable work on content-based filtering, which can be seen as an extension of the work done on information filtering. Content-based filtering has been applied to many different domains. Early work on content-based filtering included the NewsWeeder system by Lang et al. [14], which used the words contained in newsgroup messages as its features. Alspector et al. [1] compared a CF approach to movie recommendation with content-based filtering. For their content-based component they built metadata representations of all movies using fields such as directory, genre, and awards, and used linear regression and classification and regression trees to learn user profiles and rank-order the items for those users. They found that CF performed significantly better than the content-based methods, but noted that this was likely due to the poor feature set they used. Mooney et al. [15] describe Libra, a content-based book recommender system. They crawled the book metadata from the Amazon website and represented each book as a bag-of-words vector. They then used a Naive Bayes classifier to learn user profiles and to rank-order unseen books for the user.

We are not the first to suggest the combination of CF with content-based filtering, as the advantages of both approaches are largely complementary. CF is the more mature of the two approaches and works best in a situation with a stable set of items and a dense user base. Content-based filtering methods are better at dealing with sparse, dynamic domains such as news filtering, and are better at recommending for non-average users. Basu et al. [3] were among the first to propose a hybrid recommender system that used both collaborative and content features to represent the users and items. The collaborative features captured what movies a user likes and the content features included metadata fields such as actors, directors, genre, titles, and tag lines. They used Ripper, a rule-based machine learning algorithm to predict which items are interesting, and found that the combination of collaborative and content-based features produced the best results. Claypool et al. [8] presented a weighted hybrid recommender system that calculated a weighted average of the output of two separate CF and content-based filtering components. The CF component received a stronger weight as the data sets grows denser, gradually phasing out the influence of the content-based component. They did not find any significant differences between the performance of the separate components or the combined version. Baudisch [4] proposed an innovative approach to incorporating metadata into CF algorithms by joining the metadata descriptions to the user-item matrix as additional users.

6. CONCLUSIONS

In this paper we have presented a range of collaborative and content-based approaches to item recommendation on social bookmarking websites. Our algorithms were evaluated on four realistic data sets of different domains, and compared to two external, state-of-the-art approaches. Let us step back now and take stock of our findings. Tags represent an additional layer of information in the folksonomy that binds users and items together. These tags can be used successfully to improve the recommendations of standard nearest-neighbor algorithms, but this depends on the algorithm. For item-based filtering, using tags for calculating item similarity alleviates sparsity and results in better performance. At the user level, however, tags do not offer the same benefits.

Metadata can also be used successfully to generate item recommendations for social bookmarking websites. While the best approach seems to be dependent on the data set and the domain, aggregating all of the intrinsic metadata at the user and item level results in algorithms that outperform the algorithms using only information from the folksonomy.

For future work, we intend to examine the benefits of data fusion. The tag-aware fusion approach by Tso-Sutter et al. [23] demonstrates the potential of fusing together the outputs of different recommendations algorithms and representations.

Acknowledgments

The work described in this paper was funded by SenterNovem / the Dutch Ministry of Economics Affairs as part of the IOP-MMI À Propos project, and by the Netherlands Organization for Scientific Research as part of the NWO Vernieuwingsimpuls program.

7. REFERENCES

[1] J. Alspector, A. Koicz, and N. Karunanithi. Feature-based and Clique-based User Models for Movie Selection: A Comparative Study. *User Modeling and User-Adapted Interaction*, 7 (4):279–304, 1997.

[2] S. Amer-Yahia, A. Galland, J. Stoyanovich, and C. Yu. From del.icio.us to x.qui.site: Recommendations in Social Tagging Sites. In *Proceedings of SIGMOD '08*, pp. 1323–1326, New York, NY, USA, 2008. ACM.

[3] C. Basu, H. Hirsh, and W. W. Cohen. Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pp. 714–720, 1998.

[4] P. Baudisch. Joining Collaborative and Content-based Filtering. In *Proceedings of the ACM CHI Workshop on Interacting with Recommender Systems*. ACM Press, May 1999.

[5] T. Bogers and A. Van den Bosch. Using Language Modeling for Spam Detection in Social Reference Manager Websites. In R. Aly, C. Hauff, I. den Hamer, D. Hiemstra, T. Huibers, and F. de Jong, editors, *Proceedings of the 9th Belgian-Dutch Information Retrieval Workshop (DIR 2009)*, pp. 87–94, Enschede, February 2009.

[6] J. S. Breese, D. Heckerman, and C. Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 43–52, 1998.

[7] R. Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4): 331–370, 2002.

[8] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining Content-Based and Collaborative Filters in an Online Newspaper. In *Proceedings of ACM SIGIR Workshop on Recommender Systems*, August 1999.

[9] M. Clements, A. P. de Vries, and M. J. Reinders. Optimizing Single Term Queries using a Personalized Markov Random Walk over the Social Graph. In *Proceedings of ESAIR '08*, 2008.

[10] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An Algorithmic Framework for Performing Collaborative Filtering. In *Proceedings of SIGIR '99:*, pp. 230–237, New York, NY, USA, 1999. ACM.

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

[12] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Information Retrieval in Folksonomies: Search and Ranking. In *Proceedings of ESWC '06*, 2006.

[13] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. BibSonomy: A Social Bookmark and Publication Sharing System. In *Proceedings of the Conceptual Structures Tool Interoperability Workshop at ICCS 2006*, pp. 87–102, 2006.

[14] K. Lang. NewsWeeder: Learning to Filter Netnews. In *Proceedings of ICML '95*, pp. 331–339, San Mateo, CA, USA, 1995. Morgan Kaufmann.

[15] R. J. Mooney and L. Roy. Content-Based Book Recommending Using Learning for Text Categorization. In *Proceedings of DL '00*, pp. 195–204, New York, NY, 2000. ACM Press.

[16] R. Nakamoto, S. Nakajima, J. Miyazaki, and S. Uemura. Tag-Based Contextual Collaborative Filtering. In *Proceedings of the 18th IEICE Data Engineering Workshop*, 2007.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[18] G. Salton and C. Buckley. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[19] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of WWW '01*, pp. 285–295, New York, NY, USA, 2001. ACM.

[20] T. Strohman, D. Metzler, and W. B. Croft. Indri: A Language Model-based Search Engine for Complex Queries. In *Proceedings of ICIA '05*, May 2005.

[21] P. Symeonidis, M. Ruxanda, A. Nanopoulos, and Y. Manolopoulos. Ternary Semantic Analysis of Social Tags for Personalized Music Recommendation. In *Proceedings of ISMIR '08*, pp. 219–224, 2008.

[22] M. Szomszor, C. Cattuto, H. Alani, K. O'Hara, A. Baldassarri, V. Loreto, and V. D. Servedio. Folksonomies, the Semantic Web, and Movie Recommendation. In *Proceedings of the ESWC Workshop on Bridging the Gap between Semantic Web and Web 2.0*, 2007.

[23] K. H. L. Tso-Sutter, L. B. Marinho, and L. Schmidt-Thieme. Tag-aware Recommender Systems by Fusion of Collaborative Filtering Algorithms. In *Proceedings of SAC '08*, pp. 1995–1999, New York, NY, 2008. ACM.

[24] R. Wetzker, W. Umbrath, and A. Said. A Hybrid Approach to Item Recommendation in Folksonomies. In *Proceedings of ESAIR '09*, pages 25–29, New York, NY, USA, 2009. ACM.

Improving FolkRank With Item-Based Collaborative Filtering

Jonathan Gemmell, Thomas Schimoler, Maryam Ramezani,
Laura Christiansen, Bamshad Mobasher
Center for Web Intelligence
School of Computing, DePaul University
Chicago, Illinois, USA
{jgemmell, tschimo1, mramezani, lchris10, mobasher}@cdm.depaul.edu

ABSTRACT

Collaborative tagging applications allow users to annotate online resources. The result is a complex tapestry of interrelated users, resources and tags often called a folksonomy. Folksonomies present an attractive target for data mining applications such as tag recommenders. A challenge of tag recommendation remains the adaptation of traditional recommendation techniques originally designed to work with two dimensional data. To date the most successful recommenders have been graph based approaches which explicitly connects all three components of the folksonomy.

In this paper we speculate that graph based tag recommendation can be improved by coupling it with item-based collaborative filtering. We motive this hypothesis with a discussion of informational channels in folksonomies and provide a theoretical explanation of the additive potential for item-based collaborative filtering. We then provided experimental results on hybrid tag recommenders built from graph models and other techniques based on popularity, user-based collaborative filtering and item-based collaborative filtering.

We demonstrate that a hybrid recommender built from a graph based model and item-based collaborative filtering outperforms its constituent recommenders. Furthermore the inability of the other recommenders to improve upon the graph-based approach suggests that they offer information already included in the graph based model. These results confirm our conjecture. We provide extensive evaluation of the hybrids using data collected from three real world collaborative tagging applications.

1. INTRODUCTION

Collaborative tagging has emerged as a popular method for organizing and sharing online content with user-defined keywords. Delicious[1], Flickr[2] and Last.fm[3] are among the most popular destinations on the Web allowing users to annotate bookmarks, digital photographs and music respectively. Other less popular tagging applications serve niche communities enabling users to tag blogs, business documents or scholarly articles.

At the heart of collaborative tagging is the post; a user describes a resource with a set of tags. A collection of posts results in a complex network of interrelated users, resources and tags commonly referred to as a folksonomy [16]. Users are able to navigate this network free from a rigid conceptual hierarchy.

Despite the freedom users enjoy, the size of a folksonomy often hampers the userŠs exploration. Data mining applications such as recommenders can assist the user by reducing a burdensome number of items to a smaller collection related the user's interests. In this work we focus on tag recommendation, the suggestion of tags during the annotation process.

Tag recommendation reduces the cognitive effort from generation to recognition. Users are therefore encouraged to tag more frequently, apply more tags to a resource, reuse common tags and use tags the user had not previously considered. User error is reduced by eliminating capitalization inconsistencies, punctuation errors, misspellings and other discrepancies. The final result is a cleaner denser dataset that is useful in its own right or for further data mining applications.

Despite the richness offered by folksonomies, they also present unique challenges for tag recommenders. Traditional recommendation strategies, often developed to work with two dimensional data, must be adapted to work with the three dimensional nature of folksonomies. Otherwise they risk disregarding potentially useful information. To date the most successful tag recommenders are graph-based models, which exploits the user-defined links between the users, resources and tags.

In this work we propose augmenting the graph based approach with item-based collaborative filtering. We offer a discussion of information channels in folksonomies to motivate this proposal. The graph based model covers the user-resource, user-tag, and resource-tag channels. Item-based collaborative filtering, on the other hand, focuses on tags previously applied by the user to resources similar to the query resource. It therefore includes resource-resource information not explicitly contained in the graph model. Additionally, the user-tag information utilized by item-based collaborative filtering is more oriented to query resource.

We construct hybrid tag recommenders composed of the graph models and other techniques including popularity models, user-based collaborative filtering and item-based collaborative filtering. The graph based recommender coupled with item-based collaborative filtering produces better results than either produce alone, strengthening our theory that that item-based collaborative filtering contains information that is absent in the graph based model. More-

[1]delicious.com
[2]www.flickr.com
[3]www.last.fm

over the other hybrids do not improve upon the graph based model suggesting that the information they contain are already adequately represented by the graph based approach.

The rest of this paper is organized as follows. In Section 2 we describe related works. A brief survey of the tag recommenders we employ in our experiments is given in Section 3. The use of hybrid recommenders is motivated in Section 4 where we discuss informational channels in folksonomies. Section 5 details how tag recommenders may be compounded to produce hybrid recommenders. Our experimental evaluation is presented in Section 6, including a description of our datasets, our methodology and a discussion of our findings. Finally in Section 7 we present our conclusions and lay a foundation for future work.

2. BACKGROUND AND RELATED WORK

The term *folksonomy* was coined by [28], a play on *folk* and *taxonomy*. While the term is new, [29] argues that collaborative tagging in merely a renaissance of manual indexing. However, the scope and connectivity of the Internet permits tagging to rise to a level heretofore unrealized.

In [16] the attractiveness of tagging is outlined: serendipitous browsing, a low entry cost, utilizing the wisdom of the crowd, and a sense of community. Moreover, he argues that tagging allows objects to be categorized under multiple tags, unfettered from traditional taxonomies. He also discusses two obstacles: tag ambiguity in which a tag has several meanings and tag redundancy in which several tags have the same meaning.

As collaborative tagging applications have gained in popularity researchers have explored and characterized the tagging phenomenon. In [15] and [10] the authors studied the information dynamics of Delicious, one of the most popular folksonomies. The authors discussed how tags have been used by individual users over time and how tags for an individual resource stabilize over time. In [15] the authors provide an overview of the phenomenon and offer reasons why both folksonomies and taxonomies will have a place in the future of information access.

There have been many recent research investigations into recommendation within folksonomies. Unlike traditional recommender systems which have a two-dimensional relation between users and items, tagging systems have a three dimensional relation between users, tags and resources. Recommender systems can be used to recommend each of the dimensions based on one or two of the other dimensions. In [26] the authors apply user-based and item-based collaborative filtering to recommend resources in a tagging system and uses tags as an extension to the user-item matrices. Tags are used as context information to recommend resources in [19] and [18].

In [13] user-based collaborative filtering is compared to a graph-based recommender based on the PageRank algorithm for tag recommendation. The authors in [11] use association rules to recommend tags and introduce an entropy-based metric to define how predictable a tag is. In [14] the title of a resource, the posts of a resource and the user's vocabulary are used to recommend tags.

User-defined tags and co-occurrence are employed by [24] to recommend tags to users on Flickr. The assumption is that the user has already assigned a set of tags to a photo and the recommender uses those tags to recommend more tags. The authors in [6] have completed a similar study and introduce a classification for tag recommendation. Probabilistic models have been used in recommendation in folksonomies in [20] and [30]. Moreover, [20] uses Probabilistic Latent Semantic Analysis for resource discovery and [30] uses single aspect PLSA for tag recommendation.

Previously, in [8, 9], we demonstrated how tag clusters serving as coherent topics can aid in the personalization of search and navigation. Further support for the utility of clustering is offered in [4] where improvement in search through clustering is theorized. In [7] we adapted K-Nearest Neighbor for tag recommendation and showed incorporating user tagging habits into recommendation can improve K-Nearest Neighbor.

General criteria for a good tagging system including high coverage of multiple channels, high popularity and least-effort are presented in [31]. They categorize tags as content-based tags, context-based tags, attribute tags, subjective tags, and organizational tags and use a probabilistic method to recommend tags. In [2] the authors propose a classification algorithm for tag recommendation. Semantic tag recommendation systems in the context of a semantic desktop are explored in [1]. Clustering to make real-time tag recommendation is developed in [25].

3. TAG RECOMMENDATION

Here we first provide a model of folksonomies, then review several common recommendation techniques which we employ in our evaluation. A folksonomy can be described as a four-tuple:

$$D = \langle U, R, T, A \rangle \tag{1}$$

where, U is a set of users; R is a set of resources; T is a set of tags; and A is a set of annotations, represented as user-tag-resource triples:

$$A \subseteq \{\langle u, r, t \rangle : u \in U, r \in R, t \in T\} \tag{2}$$

A folksonomy can, therefore, be viewed as a tripartite hyper-graph [17] with users, tags, and resources represented as nodes and the annotations represented as hyper-edges connecting a user, a tag and a resource.

Aggregate projections of the data can be constructed, reducing the dimensionality but sacrificing information [22]. The relation between resources and tags, RT, can be formulated such that each entry, $RT(r, t)$, is the weight associated with the resource, r, and the tag, t. This weight may be binary, merely showing that one or more users have applied that tag to the resource. In this work we assume $RT(r, t)$ to be the number of users that have applied t to the r:

$$RT_{tf}(r, t) = |\{a = \langle u, r, t \rangle \in A : u \in U\}| \tag{3}$$

Analogous two-dimensional projections can be constructed for *UT* in which the weights correspond to users and tags, and *UR* in which the weights correspond to users and resources.

Many authors have attempted to exploit the data model for recommendation in folksonomies. In traditional recommendation algorithms the input is often a user, u, and the output is a set of items, I. Tag recommendation differs in that the input is both a user and a resource. The output remains a set of items, in this case a set of recommended tags, T_r. Given a user-resource pair, the recommendation set is constructed by calculating a weight for each tag, $w(u, r, t)$, and recommending the top n tags.

3.1 Popularity Based Approaches

We consider two popularity based models which rely on the frequency a tag is used. **PopRes** ignores the user and relies on the popularity of a tag within the context of a particular resource. We define the resource based popularity measure as:

$$w(u, r, t) = \frac{|\{a = \langle u, r, t \rangle \in A : u \in U\}|}{|\{a = \langle u, r, t \rangle \in A : u \in U, t \in T\}|} \tag{4}$$

PopUser, on the other hand, ignores the resource and focuses on the frequency of a tag within the user profile. We define the user based popularity measure as:

$$w(u,r,t) = \frac{|\{a = \langle u,r,t\rangle \in A : r \in R\}|}{|\{a = \langle u,r,t\rangle \in A : r \in R, t \in T\}|} \quad (5)$$

Popularity based recommenders require little online computation. Models are built offline and can be incrementally updated. However both these models focus on a single channel of the folksonomy and may not incorporate otherwise relevant information into the recommendation.

3.2 User-Based Collaborative Filtering

User-based K-nearest neighbor is a commonly used recommendation algorithm in Information Retrieval that can be modified for use in folksonomies. Applications may model users by recency, authority, linkage or vector space models. In this work we focus on the vector space model [21] and describe the user as a vector over either the tag space or the resource space.

KNN_UT models the user, u, as a vector over the set of tags where the weight in each dimension corresponds to the occurrence of the tag in the user profile as it is defined by the two dimensional projection $UT(u,t)$. Other methods may be used to model the user, such as a vector over the set of resources or a combination of tags and resources. Several techniques may be used to calculate the similarity between vectors such as Jaccard similarity or cosine similarity [27]. In this work we rely on cosine similarity.

Using the similarity measure a neighborhood, N, of the k most similar users is constructed such that they have all previously annotated the query resource, r. A weight for each tag is calculated as:

$$w(u,r,t) = \frac{\sum_n^N sim(u,n) * d(n,r,t)}{k} \quad (6)$$

where $d(n,r,t)$ is 1 if the neighbor, n, has annotated the query resource, r, with the tag t. Otherwise it is 0.

Traditional user-based collaborative filtering requires a comparison between the query user and every other user. However, since the adapted algorithm considers only those users that have annotated the query resource, the number of similarities to calculate is drastically reduced. The popularity of resources in folksonomies follows the power law and the great majority of resources will benefit from this reduced reduction in computation, while a few will require additional computational effort. As a result the algorithm scales well with large datasets.

However, since the algorithm relies on the collaboration of other users it may be the case that a tag cannot be recommended because it does not appear in a neighbor's profile. While the personalization offered by user-based filtering is an important component for the recommender, it lacks the ability to reflect the habits and patterns of the larger crowd.

3.3 Item-Based Collaborative Filtering

KNN_RT models resources as a vector over the tag space. Give a resource and a tag, we define the weight as the entry of the two dimensional projection, $RT(r,t)$, the number of times r has been tagged with t. When a user selects a resource to annotate, the cosine similarity between it and every resource in the user profile is calculated. A neighborhood of the k most similar resources, S, is then constructed. We then define the item-based collaborative filtering measure as:

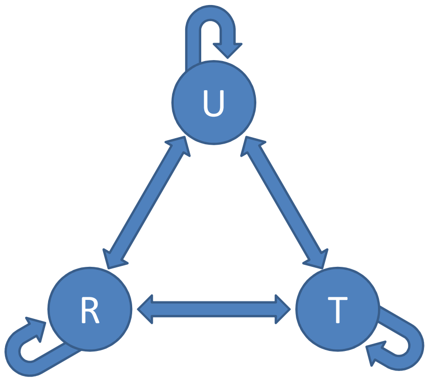


Figure 1: Informational channels of a folksonomy.

$$w(u,r,t) = \frac{\sum_s^S sim(s,r) * d(u,s,t)}{k} \quad (7)$$

where $d(u,s,t)$ will equal 1 if the user has applied t to s and 0 otherwise. This recommender focuses entirely on the user's tagging habits. Unlike the user-based filtering methods, it may be able to identify tags that are common to the user but rarely used by others. However, it lacks the ability to discover relevant tags from other users. Depending on the size of the user profile, this recommender will also scale well to larger datasets, particularly if the resource-resource similarity matrix if calculated offline.

3.4 FolkRank

FolkRank was proposed in [12]. It computes a PageRank vector from the tripartite graph of the folksonomy. This graph is generated by regarding $U \cup R \cup T$ as the set of vertices. Edges are defined by the three two-dimensional projections of the hyper-graph, RT, UR and UT.

If we regard the adjacency matrix of this graph, W, (normalized to be column-stochastic), a damping factor, d, and a preference vector, p, then we iteratively compute the PageRank vector, w, in the usual manner: $w = dAw + (1-d)p$.

However due to the symmetry inherent in the graph, this basic PageRank may focus too heavily on the most popular elements. The *FolkRank* vector is taken as a difference between two computations of PageRank: one with and one without a preference vector. Tag recommendations are generated by biasing the preference vector towards the query user and resource [13]. These elements are given a substantial weight while all other elements have uniformly small weights.

PageRank has proven to be one of the top performing tag recommenders. However, it imposes steep computational costs.

4. INFORMATIONAL CHANNELS OF FOLKSONOMIES

The model of a folksonomy suggests several informational channels which may be exploited by data mining applications such as tag recommenders. The relation between users, resources and tags generate a complex network of interrelated items as shown in Figure 1.

The channel between resources and tags reveals a highly descriptive model of the resources. The accumulation of many users' opinions (often numbered in the thousands or millions) results in a rich-

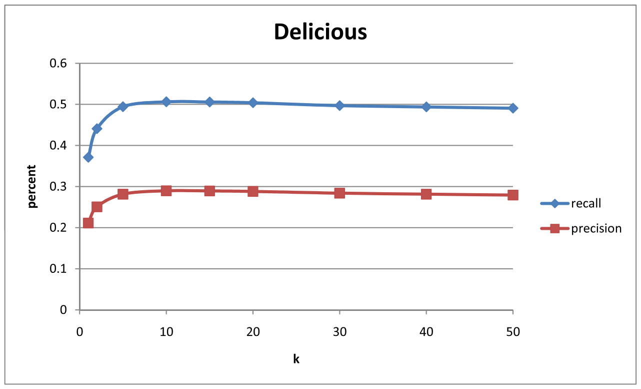


Figure 2: The effect of k in *KNN_UT* on recall and precision for a recommendation set of 5 tags. Users are modeled as a vector over the tag space.

ness which taxonomies are unable to approximate. Conversely the tags themselves are characterized by the resources to which they have been assigned.

As users annotate resource with tags they define their interests in as much as they describe a resource. The user-tag channel therefore reveals the users' interests and provides opportunities for data mining algorithms to offer a high degree of personalization. Likewise a user may be defined by the resources which he has annotated as in the user-resource channel.

These primary channels can be used to produce secondary informational channels. The user-user channel can be constructed by modeling users as a vector of tags or as a vector of resources and applying a similarity measure such as cosine similarity. Many variations exist. However the result reveals a network of users that can be explored directly or incorporated into further data mining approaches. The resource-resource and tag-tag channels provide similar utility, presenting navigational opportunities for users to explore similar resources or neighborhoods of tags.

The success of tag recommenders hinge on their ability to incorporate all of these informational channels. A simple recommender such as *PopRes* focuses only on the tag-resource channel, whereas *PopUser* includes only the information between tags and users.

Collaborative filtering techniques include additional channels but increase the computational overhead. *KNN_UT* discovers a set of neighbors, thereby covering the user-user channel. It then focuses on tags those neighbors applied to the query resource covering the user-resource and resource-tag channels. *FolkRank*, on the other hand, explicitly defines the relation between users, resources and tags in its adjacency matrix. While *FolkRank* has proven to be among most effective tag recommenders, augmenting it with algorithms that incorporate complimentary informational channels may improve its performance.

5. HYBRID RECOMMENDERS

The multiple informational channels of folksonomies present an attractive target for hybrid recommenders. Hybrids combine several recommenders together to produce a new recommender. The constituent recommenders are freed from the burden of the covering all the available informational channels and may instead focus on only a few. The hybrid then ties these recommenders together. A successful hybrid creates a synergistic blend of its constituent parts producing superior results that they could not achieve alone.

In this paper we focus on weighted hybrid recommenders [5] which combine pairs of recommenders in a linear model. Each model is trained separately. Given a user, u, and a resource, r, the hybrid queries both components for each tag in the folksonomy. The results is $W(u, r, t)$ which contains the weights for all tags. In order to ensure that weight assignments for each recommendation approach are on the same scale, we normalize the weights in $W(u, r, t)$ to 1 producing $W'(u, r, t)$.

Originally, these weights were used to select the top n items for the recommendation set. In this case, however, the weights are combined in a linear model as:

$$w(u, r, t) = \beta * w'_a(u, r, t) + \alpha * w'_b(u, r, t) \qquad (8)$$

where $\beta = 1 - \alpha$. These coefficients are used to control the contribution of the two recommenders. When α is set to 0, recommender a acts alone. In the case that α is set to 0.5, each recommender contributes equally to the final weight. For each hybrid, α must be empirically tuned to achieve the maximum synergy between the components. The tags are then resorted by the new weight, and the top n tags are recommended for the annotation.

6. EXPERIMENTAL EVALUATION

In this section we describe the methods used to gather and preprocess our datasets. Our testing methodology is outlined. We provide a discussion of how we tuned variables for each algorithm and describe the experiments on the weighted hybrid recommenders. Finally, we discuss our observations.

6.1 Datasets

Folksonomy	Delicious (5%)	Citeulike	Bibsonomy
Users	7,665	2,051	357
Resources	15,612	5,376	1,738
Tags	5,746	3,343	1,573
Posts	720,788	42,278	19,909
Annotations	2,762,235	105,873	54,848

Table 1: Datasets

We provide an extensive evaluation of the hybrid recommenders using data from three real collaborative tagging applications: Delicious, Citeulike, and Bibsonomy.

6.1.1 P-Core Processing

By P-core processing users, resources and tags are removed from the dataset in order to produce a residual dataset that guarantees each user, resource and tag occur in at least p posts [3]. Here we define a post to include a user, a resource, and every tag the user has applied to the resource.

By removing infrequent users, resources and tags noise in the data is reduced. Uncommon items whether they be tags used by only a few users, unpopular resources, or inactive users are eliminated from consideration. Because of their scarcity these are the very items likely to confound recommenders. Moreover by eliminating infrequent items the size of the dataset is dramatically reduced allowing the application of data mining techniques that might otherwise be computationally impractical.

6.1.2 Delicious

Delicious is a popular collaborative tagging application in which users annotate URLs. On 10/19/2008, 198 of the most popular tags were taken from the user interface. For each of these tags the 2,000

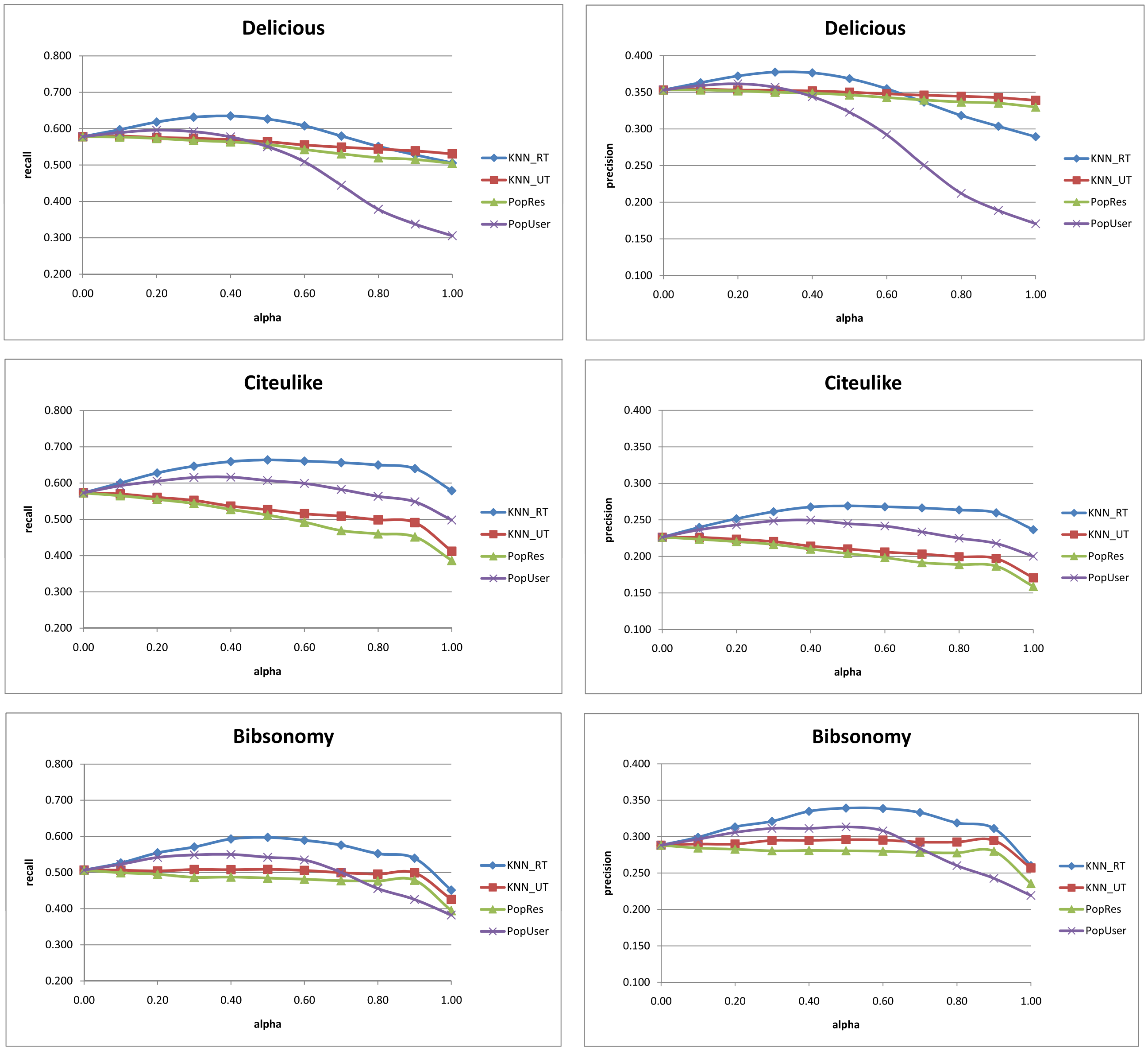


Figure 3: The effect of *alpha* on the hybrid recommenders on the Delicious, Citeulike and Bibsonomy datasets. Results are shown using recall and precision on a recommendation set of five tags.

most recent annotations including the contributors of the annotations were collected. The social network for these contributors was explored recursively collecting 524,790 usernames.

From 10/20/2008 to 12/15/2008 the complete profiles of the users were collected. Each user profile consisted of a collection of annotations including the resource, tags and date of the original bookmark. The top 100 most prolific users were visually inspected; twelve were removed from the data because their annotation count was many orders of magnitude larger than other users and were therefore suspected to be Web-bots.

Due to memory and time constraints, 5% of the user profiles was randomly selected. Still this dataset remains far larger than either the following Bibsonomy or Citeulike datasets. Experiments on larger samplings reveal near identical trends for several of the tag recommendation strategies. Some tag recommendation techniques such as *FolkRank* are so computational intensive that larger samplings of the data are not feasible. In order to best compare the recommenders, the 5% sampling was used on all reported experiments. A P-core of 20 was taken from the sample and is reported in Table 1.

6.1.3 Citeulike

Citeulike is a popular online tool used by researchers to manage and discover scholarly references. They make their dataset freely available to download[4]. On 2/17/2009 the most recent snapshot was downloaded. The data contains anonymous user ids and posts for each user including resources, the date and time of the posting

[4]www.citeulike.org/faq

and the tags applied to the resource. A P-core of 5 was taken. The characteristics of the dataset are described in Table 1.

6.1.4 Bibsonomy

This dataset was provided by Bibsonomy[5] for use in the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD) 2009 Challenge. Bibsonomy was originally launched as a collaborative tagging application allowing users to organize and share scholarly references. It has since expanded its scope allowing users to annotate URLs.

The data includes all public bookmarks and publication posts of Bibsonomy until 2009-01-01. The data was cleaned by removing all characters which are neither numbers nor letters from tags. Additionally the system tags *imported*, *public*, *systemimported*, *nn* and *systemunfiled* where removed. A P-core of 5 was used. Table 1 relates the features of the dataset.

6.2 Experimental Methodology

We have adopted the test methodology as described in [13]. In this approach, called *LeavePostOut*, a single post is randomly removed from each user's profile. The training set is then comprised of the remaining posts, while the test set contains one post per user. Each test case consists of a user, u, a resource, r, and all the tags the user has applied to that resource. These tags, T_h, are analogous to the holdout set commonly used in Information Retrieval. The tag recommendation algorithms accept the user-resource pair and return an ordered set of recommended tags, T_r.

For evaluation we adopt the common recall are precision measures as is common in Information Retrieval. Recall measures the percentage of items in the holdout set that appear in the recommendation set. It is a measure of completeness and is defined as:

$$r = |T_h \cap T_r| / |T_h| \tag{9}$$

Precision measures the percentage of items in the recommendation set that appear in the holdout set. It measures the exactness of the recommendation algorithm and is defined as:

$$p = |T_h \cap T_r| / |T_r| \tag{10}$$

For each evaluation metric the average value is calculated across all test cases.

6.3 Experimental Results

Here we present our experimental results beginning with the tuning of variables. The experiments with user-based collaborative filtering require the tuning of k, the number of neighbors.

Figure 2 shows the relation between k and the evaluation metrics recall and precision for a recommendation set of size 5. The Delicious dataset was used for this experiment. As k increases so does recall and precision. However this improvement suffers from diminishing returns until a k of 50 offers little more benefit than a k of 20. This trend was observed for K-Nearest Neighbor experiments in the other two datasets as well. As such, all *KNN_UT* experiments were completed using a k of 20.

Item-based collaborative filtering also requires the tuning of k, in this case the number of similar resources in the user profile to include in the neighborhood. After empirical analysis we found 15 to produce the best performance on all datasets.

Figure 3 shows the tuning of α for the hybrid recommenders. Each hybrid is a linear combination of *FolkRank* and one of the

[5] www.bibsonomy.org

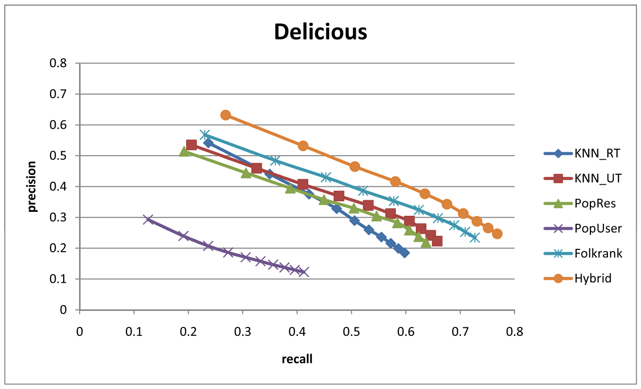


Figure 4: A comparison of tag recommender techniques in Delicious.

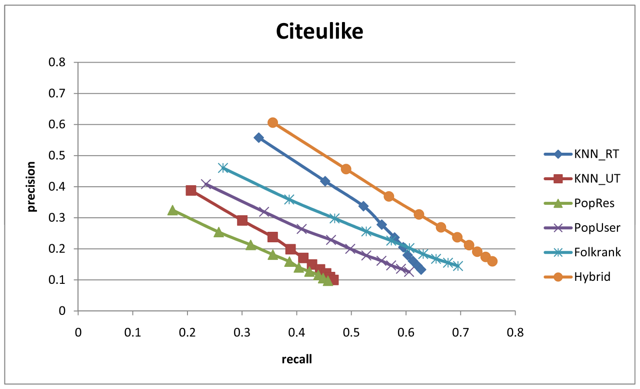


Figure 5: A comparison of tag recommender techniques in Citeulike.

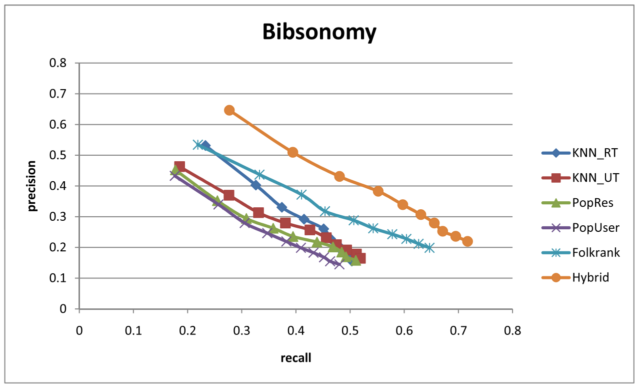


Figure 6: A comparison of tag recommender techniques in Bibsonomy.

other four recommenders. The left hand side of each graph shows the hybrid recommenders when α is set to 0 in which case *FolkRank* dominates the hybrid. As α increases more weight is given to the other recommenders until finally when α reaches 1, *FolkRank* plays no part in the recommendation.

For all datasets, item-based collaborative filtering contributes to recall and precision of its hybrid. For example in the Delicious experiment when α is set to 0.4, recall for a recommendation set of five tags is 6% higher than *FolkRank* achieves alone and 13% higher than *KNN_RT* achieves alone.

In the Delicious experiments, a hybrid built with *PopUser* offers a slight improvement, while it has a more dramatic improvement on Citeulike. These observations reveal that the personalization of the user-tag channel strongly incorporated into *KNN_RT* and *PopUser* offers information lacking in *FolkRank*. While *PopUser* boosts all of the user's tags, *KNN_RT* focuses on tags related to the resource being annotated accounted for its increased performance. On the other hand *PopRes* does not appear to provide any additional benefit to *FolkRank*. Indeed, *FolkRank* contains this information in the utilization of the RT matrix.

These two results reveal that the weights given to the query resource and query user in the *FolkRank* algorithm achieve different results. The weight applied to the resource immediately activates tags strongly associated with the resource. The result is similar to that achieved in *PopRes*, hence *PopRes* offers little assistance to its hybrid. However, the weight applied to the query user disperses through the graph activating all of the user's tags relevant or irrelevant to user's present context. *KNN_RT*, on the other hand, focuses on tags applied to resources similar to the query. Hence, it includes the resource-resource channel missing in *FolkRank*. The hybrid is able to be personalized but also be more context specific.

KNN_UT does not appear to offer any additional information that *FolkRank* did not already contain, even though it includes user-resource information in the neighborhood selection, user-resource information in the cosine similarity and resource-tag information in the recommendation step. This reveals that the way in which the informational channels is equally important. Additionally *KNN_UT* selects neighbors that are similar to the query user, utilizing the user-user channel. However, this channel does not appear to be beneficial to tag recommendation.

After analysis of the effect of α on the hybrids we selected the best α for the *FolkRank-KNN_RT* hybrid. For Delicious we used an α of 0.4. For Citeulike and Bibsonomy used an α of 0.5. Figures 4 through 6 compare tag recommenders along with the hybrid. Recall and precision are plotted for recommendation sets of size one through ten. For all datasets the hybrid outperforms its constituent parts.

We also observe a difference in the effect that constituent recommenders have across the datasets. Delicious users tag Web pages and their topics cover a wide array of topics. Citeulike users tag scholarly articles and often focus on their area of expertise. In fact we can see in Figures 4 and 5 the dramatic difference between *PopRes* and *PopUser*.

In Delicious *PopRes* outperforms *PopUser*, whereas in Citeulike the opposite is true. The user's focus on a narrow subject area in Citeulike make the user-tag channel a informative predictor, whereas the topic variety in the profiles of Delicious users make the resource-tag channel more reliable.

This analysis is underscored by the success *KNN_RT* hybrid has on the Delicious datasets where *PopUser* hybrid fairs poorly. Because *KNN_RT* focuses on those tags applied to resources similar to the query resource it offers context appropriate tags. In Citeulike, where users have a narrow focus, this context provides little additional benefit and the *PopUser* hybrid performs nearly as well as the *KNN_RT* hybrid. Bibsonomy users tags both citations and web pages; its results fall between those of the other two datasets.

7. CONCLUSIONS

We have demonstrated that tag recommenders may be combined to form weighted hybrids that perform better than either performs alone. Moreover *FolkRank* one of the most successful tag recommenders to date can be augmented with item-based collaborative filtering to produce superior results. The resource-resource and personalized user-resource channels covered by item-based collaborative filtering compliment the channels utilized by *FolkRank*. The inability of other recommenders to improve upon *FolkRank* provides evidence that *FolkRank* sufficiently incorporates the informational channels covered by those recommenders.

Future work will involve investigating alternative hybrid tag recommenders. New recommenders that cover other informational channels will be considered. Finally, alternative methods for hybridizing recommenders will be explored.

8. ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation Cyber Trust program under Grant IIS-0430303 and a grant from the Department of Education, Graduate Assistance in the Area of National Need, P200A070536.

9. REFERENCES

[1] B. Adrian, L. Sauermann, and T. Roth-Berghofer. Contag: A semantic tag recommendation system. In T. Pellegrini and S. Schaffert, editors, *Proceedings of I-Semantics' 07*, pages pp. 297–304. JUCS, 2007.

[2] P. Basile, D. Gendarmi, F. Lanubile, and G. Semeraro. Recommending smart tags in a social bookmarking system. In *Bridging the Gep between Semantic Web and Web 2.0 (SemNet 2007)*, pages 22–29, 2007.

[3] V. Batagelj and M. Zaveršnik. Generalized cores. *Arxiv preprint cs/0202039*, 2002.

[4] G. Begelman, P. Keller, and F. Smadja. Automated Tag Clustering: Improving search and exploration in the tag space. *Proceedings of the Collaborative Web Tagging Workshop at WWW*, Volume 6, 2006.

[5] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User Adapted Interaction*, 12(4):331–370, 2002.

[6] N. Garg and I. Weber. Personalized, interactive tag recommendation for flickr. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 67–74, New York, NY, USA, 2008. ACM.

[7] J. Gemmell, T. Schimoler, M. Ramezani, and B. Mobasher. Adapting k-nearest neighbor for tag recommendation in folksonomies. *Intelligent Techniques for Web Personalization & Recommender Systems*, 2009.

[8] J. Gemmell, A. Shepitsen, B. Mobasher, and R. Burke. Personalization in Folksonomies Based on Tag Clustering. *Intelligent Techniques for Web Personalization & Recommender Systems*, 2008.

[9] J. Gemmell, A. Shepitsen, B. Mobasher, and R. Burke. Personalizing navigation in folksonomies using hierarchical tag clustering. In *Proceedings of the 10th international conference on Data Warehousing and Knowledge Discovery*. Springer, 2008.

[10] S. A. Golder and B. A. Huberman. Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32(2):198, 2006.

[11] P. Heymann, D. Ramage, and H. Garcia-Molina. Social tag prediction. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 531–538, New York, NY, USA, 2008. ACM.

[12] A. Hotho, R. Jaschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. *Lecture Notes in Computer Science*, 4011:411, 2006.

[13] R. Jaschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag Recommendations in Folksonomies. *LECTURE NOTES IN COMPUTER SCIENCE*, 4702:506, 2007.

[14] M. Lipczak. Tag recommendation for folksonomies oriented towards individual users. In *Proceedings of the ECML/PKDD 2008 Discovery Challenge Workshop, part of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2008.

[15] G. Macgregor and E. McCulloch. Collaborative tagging as a knowledge organisation and resource discovery tool. *Library Review*, 55(5):291–300, 2006.

[16] A. Mathes. Folksonomies-Cooperative Classification and Communication Through Shared Metadata. *Computer Mediated Communication, (Doctoral Seminar), Graduate School of Library and Information Science, University of Illinois Urbana-Champaign, December*, 2004.

[17] P. Mika. Ontologies are us: A unified model of social networks and semantics. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(1):5–15, 2007.

[18] R. Y. Nakamoto, S. Nakajima, J. Miyazaki, S. Uemura, and H. Kato. Investigation of the effectiveness of tag-based contextual collaborative filtering in website recommendation. In *Advances in Communication Systems and Electrical Engineering*, pages 309–318. Springerlink, 2008.

[19] R. Y. Nakamoto, S. Nakajima, J. Miyazaki, S. Uemura, H. Kato, and Y. Inagaki. Reasonable tag-based collaborative filtering for social tagging systems. In *WICOW '08: Proceeding of the 2nd ACM workshop on Information credibility on the web*, pages 11–18, New York, NY, USA, 2008. ACM.

[20] A. Plangprasopchok and K. Lerman. Exploiting social annotation for automatic resource discovery. *CoRR*, abs/0704.1675, 2007.

[21] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[22] C. Schmitz, A. Hotho, R. Jaschke, and G. Stumme. Mining association rules in folksonomies. In *Proc. IFCS 2006 Conference*, pages 261–270. Springer, 2006.

[23] B. Sigurbjörnsson and R. van Zwol. Flickr tag recommendation based on collective knowledge. pages 327–336, 2008.

[24] B. Sigurbjörnsson and R. van Zwol. Flickr tag recommendation based on collective knowledge. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 327–336, New York, NY, USA, 2008. ACM.

[25] Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W.-C. Lee, and C. L. Giles. Real-time automatic tag recommendation. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 515–522, New York, NY, USA, 2008. ACM.

[26] K. H. L. Tso-Sutter, L. B. Marinho, and L. Schmidt-Thieme. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 1995–1999, New York, NY, USA, 2008. ACM.

[27] C. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann Newton, MA, USA, 1979.

[28] T. Vander Wal. Folksonomy definition and wikipedia. *vanderwal. net*, 2005.

[29] J. Voss. Tagging, Folksonomy & Co-Renaissance of Manual Indexing? *Arxiv preprint cs/0701072*, 2007.

[30] R. Wetzker, W. Umbrath, and A. Said. A hybrid approach to item recommendation in folksonomies. In *ESAIR '09: Proceedings of the WSDM '09 Workshop on Exploiting Semantic Annotations in Information Retrieval*, pages 25–29, New York, NY, USA, 2009. ACM.

[31] Z. Xu, Y. Fu, J. Mao, and D. Su. Towards the semantic web: Collaborative tag suggestions. *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland, May*, 2006.

A Tag Recommender System Exploiting User and Community Behavior

Cataldo Musto
Dept. of Computer Science
University of Bari 'Aldo Moro'
Italy
cataldomusto@di.uniba.it

Fedelucio Narducci
Dept. of Computer Science
University of Bari 'Aldo Moro'
Italy
narducci@di.uniba.it

Marco De Gemmis
Dept. of Computer Science
University of Bari 'Aldo Moro'
Italy
degemmis@di.uniba.it

Pasquale Lops
Dept. of Computer Science
University of Bari 'Aldo Moro'
Italy
lops@di.uniba.it

Giovanni Semeraro
Dept. of Computer Science
University of Bari 'Aldo Moro'
Italy
semeraro@di.uniba.it

ABSTRACT

Nowadays Web sites tend to be more and more social: users can upload any kind of information on collaborative platforms and can express their opinions about the content they enjoyed through textual feedbacks or reviews. These platforms allow users to annotate resources they like through freely chosen keywords (called tags). The main advantage of these tools is that they perfectly fit user needs, since the use of tags allows organizing the information in a way that closely follows the user mental model, making retrieval of information easier. However, the heterogeneity characterizing the communities causes some problems in the activity of social tagging: someone annotates resources with very specific tags, other people with generic ones, and so on. These drawbacks reduce the exploitation of collaborative tagging systems for retrieval and filtering tasks. Therefore, systems that assist the user in the task of tagging are required. The goal of these systems, called tag recommenders, is to suggest a set of relevant keywords for the resources to be annotated. This paper presents a tag recommender system called STaR (Social Tag Recommender system). Our system is based on two assumptions: 1) the more two or more resources are similar, the more they share common tags 2) a tag recommender should be able to exploit tags the user already used in order to extract useful keywords to label new resources. We also present an experimental evaluation carried out using a large dataset gathered from Bibsonomy.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing: Indexing methods; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval: Information filtering

General Terms

Algorithms, Experimentation

Keywords

Recommender Systems, Web 2.0, Collaborative Tagging Systems, Folksonomies

1. INTRODUCTION

We are assisting to a transformation of the Web towards a more user-centric vision called Web 2.0. By using Web 2.0 applications users are able to publish auto-produced contents such as photos, videos, political opinions, reviews, hence they are identified as *Web prosumers*: *pro*ducers + con*sumers* of knowledge. Recently the research community has thoroughly analyzed the dynamics of *tagging*, which is the act of annotating resources with free labels, called *tags*. Many argue that, thanks to the expressive power of *folksonomies* [17], collaborative tagging systems are very helpful to users in organizing, browsing and searching resources. This happens because, in contrast to systems where information about resources is only provided by a small set of experts, the model of collaborative tagging systems takes into account the way individuals conceive the information contained in a resource [18], so they perfectly fit user needs and user mental model. Nowadays almost all Web 2.0 platforms embed tagging: we can cite Flickr[1], YouTube[2], Del.icio.us[3], Last.fm[4], Bibsonomy[5] and so on. These systems provide heterogeneous contents (photos, videos, musical habits, etc.), but they all share a common core: they let users to post new resources and to annotate them with tags. Besides the simple act of annotation, the tagging of resources has also a key social aspect; the connection between users, resources and tags generates a tripartite graph that can be easily exploited to

[1]http://www.flickr.com
[2]http://www.youtube.com
[3]http://delicious.com/
[4]http://www.last.fm/
[5]http://www.bibsonomy.org/

analyze the dynamics of collaborative tagging systems. For example, users that label the same resource by using the same tags might have similar tastes and items labeled with the same tags might share common characteristics.

Undoubtedly the power of tagging lies in the ability for people to freely determine the appropriate tags for a resource [10]. Since folksonomies do not rely on a predefined lexicon or hierarchy they have the main advantage to be fully free, but at the same time they generate a very noisy tag space, really hardly to exploit for retrieval or recommendation tasks without performing any form of processing. Golder et. al. [4] identified three major problems of collaborative tagging systems: *polysemy*, *synonymy*, and *level variation*. Polysemy refers to situations where tags can have multiple meanings: for example a resource tagged with the term *bat* could indicate a news taken from an online sport newspaper or a Wikipedia article about nature. We refer to synonymy when multiple tags share a single meaning: for example we can have simple morphological variations (such as 'AI','artificial_intelligence' and so on to identify a scientific publication about Artificial Intelligence) but also lexical relations (like resources tagged with 'arts' versus 'cultural heritage'). At last, the phenomenon of tagging at different levels of abstraction is defined as *level-variation*. This happens when people annotate the same web page, containing for example a recipe for roast turkey with the tag 'roast-turkey' but also with a simple 'recipe'.

Since these problems are of hindrance to completely exploit the expressive power of folksonomies, in the last years many tools have been developed to assist the user in the task of tagging and to aid at the same time the tag convergence [3]: we refer to them as tag recommenders. These systems work in a very simple way:

1. a user posts a resource;
2. depending on the approach, the tag recommender analyzes some information related to the resource (usually metadata or a subset of the relations in the aforementioned tripartite graph);
3. the tag recommender processes this information and produces a list of recommended tags;
4. the user freely chooses the most appropriate tags to annotate the resource.

Clearly, the more these recommended tags match the user needs and her mental model, the more she will use them to annotate the resource. In this way we can rapidly speed up the tag convergence aiding at the same time in filtering the noise of the complete tag space.
This paper presents the tag recommender STaR. When developing the model, we tried to point out two concepts:

- resources with similar content should be annotated with similar tags;
- a tag recommender needs to take into account the previous tagging activity of users, increasing the weight of the tags already used to annotate similar resources.

In this work, we identify two main aspects in the tag recommendation task: first, each user has a typical manner to label resources; second, similar resources usually share common tags.

The paper is organized as follows. Section 2 analyzes related work. Section 3 explains the architecture of the system and how the recommendation approach is implemented. The experimental evaluation carried out is described in Section 4, while conclusions and future work are drawn in the last section.

2. RELATED WORK

Usually the works in the tag recommendation area are broadly divided into three classes: *content-based*, *collaborative* and *graph-based* approaches.

In the content-based approach, exploiting some Information Retrieval-related techniques, a system is able to extract relevant unigrams or bigrams from the text. Brooks et. al [2], for example, develop a tag recommender system that exploits TF/IDF scoring [13] in order to automatically suggests tags for a blog post. In [5] is presented a novel method for key term extraction from text documents. Firstly, every document is modeled as a graph which nodes are terms and edges represent semantic relationship between them. These graphs are then partitioned using communities detection techniques and weighted exploiting information extracted from Wikipedia. The tags composing the most relevant communities (a set of terms related with the topic of the resource) are then suggested to the user.

AutoTag [11] is one of the most important systems implementing the collaborative approach for tag recommendation. It presents some analogies with collaborative filtering methods: as in the collaborative recommender systems the recommendations are generated based on the ratings provided by similar users (called neighbors), in AutoTag the system suggests tags based on the other tags associated with similar posts. Firstly, the tool exploits some IR techniques in order to find similar posts and extracts the tags they are annotated with. All the tags are then merged, building a folksonomy that is filtered and re-ranked. The top-ranked tags are finally suggested to the user, who selects the most appropriate ones to attach to the post.

TagAssist [15] extends the AutoTags' approach introducing some preprocessing step (specifically, a lossless compression over existing data) in order to improve the quality of the recommendations. The core of this approach is represented by a a Tag Suggestion Engine (TSE) which leverages previously tagged posts providing appropriate suggestions for new content.

Marinho [9] investigates the user-based collaborative approach for tag recommendation. The main outcome of this work shows that users with a similar tag vocabulary tend to tag alike, since the method seems to produce good results when applied on the user-tag matrix.

The problem of tag recommendation through graph-based approaches has been firstly addressed by Jäschke et al. in [7]. The key idea behind their FolkRank algorithm is that a resource which is tagged by important tags from important users becomes important itself. So, they build a graph whose nodes mutually reinforces themselves by spreading their weights. They compared some recommendation techniques including collaborative filtering, PageRank and FolkRank, showing that the FolkRank algorithm outperforms other approaches. Furthermore, Schmitz et al. [14] proposed association rule mining as a technique that might be useful in the tag recommendation process.

In literature we can find also other methods (called *hy-*

brid), which try to integrate two of more sources of knowledge (mainly, content and collaborative ones) in order to improve the quality of recommended tags.

Heymann et. al [6] present a tag recommender that exploits at the same time social knowledge and textual sources. They produce recommendations exploiting both the HTML source code (extracting anchor texts and page texts) and the annotations of the community. The effectiveness of this approach is also confirmed by the use of a large dataset crawled from del.icio.us for the experimental evaluation.

Lipczak in [8] proposes a similar hybrid approach. Firstly, the system extracts tags from the title of the resource. Afterwards, it performs an analysis of co-occurrences, in order to expand the sets of candidate tags with the tags that usually co-occur with terms in the title. Finally, tags are filtered and re-ranked exploiting the informations stored in a so-called "personomy", the set of the tags previously used by the user.

Finally, in [16] the authors proposed a model based on both textual contents and tags associated with the resource. They introduce the concept of *conflated tags* to indicate a set of related tags (like blog, blogs, ecc.) used to annotate a resource. Modeling in this way the existing tag space they are able to suggest various tags for a given bookmark exploiting both *user* and *document* models.

3. STAR: A SOCIAL TAG RECOMMENDER SYSTEM

Following the definition introduced in [7], a folksonomy can be described as a triple (U, R, T) where:

- U is a set of *users*;
- R is a set of *resources*;
- T is a set of *tags*.

We can also define a tag assignment function $\textsc{tas}$: $U \times R \rightarrow T$.

So, a collaborative tagging system is a platform composed of users, resources and tags that allows users to freely assign tags to resources, while the tag recommendation task for a given user $u \in U$ and a resource $r \in R$ can be described as the generation of a set of tags $\textsc{tas}(u, r) \subseteq T$ according to some relevance model. In our approach these tags are generated from a ranked set of *candidate tags* from which the top n elements are suggested to the user.
STaR (<u>S</u>ocial <u>Tag</u> <u>R</u>ecommender) is a content-based tag recommender system, developed at the University of Bari. The inceptive idea behind STaR is to improve the model implemented in systems like TagAssist [15] or AutoTag [11].

Although we agree that similar resources usually share similar tags, in our opinion Mishne's approach presents two important drawbacks:

1. the tag re-ranking formula simply performs a sum of the occurrences of each tag among all the folksonomies, without considering the similarity with the resource to be tagged. In this way tags often used to annotate resources with a low similarity level could be ranked first;
2. the proposed model does not take into account the previous tagging activity performed by users. If two users bookmarked the same resource, they will receive the same suggestions since the folksonomies built from similar resources are the same.

We will try to overcome these drawbacks, by proposing an approach firstly based on the analysis of similar resources capable also of leveraging the tags already selected by the user during her previous tagging activity, by putting them on the top of the tag rank. Figure 1 shows the general architecture of STaR. The recommendation process is performed in four steps, each of which is handled by a separate component.

3.1 Indexing of Resources

Given a collection of resources (*corpus*) with some textual metadata (such as the title of the resource, the authors, the description, etc.), STaR firstly invokes the *Indexer* module in order to perform a preprocessing step on these data by exploiting Apache Lucene[6]. Obviously, the kind of metadata to be indexed is strictly dependant on the nature of the resources. For example, supposing to recommend tags for *bookmarks*, we could index the title of the web page and the extended description provided by users, while for *BibteX* entries, we could index the title of the publication and the abstract. Let U be the set of users and N the cardinality of this set, the indexing procedure is repeated $N + 1$ times: we build an index for each user (*Personal Index*) storing the information on the resources she previously tagged and an index for the whole community (*Social Index*) storing the information about all the tagged resources by merging the singles Personal Indexes.

Following the definitions presented above, given a user $u \in U$ we define $PersonalIndex(u)$ as:

$$PersonalIndex(u) = \{r \in R | \exists t \in T : \textsc{tas}(u, r) = t\} \qquad (1)$$

where $\textsc{tas}$ is the tag assignment function $\textsc{tas}$: $U \times R \rightarrow T$ which assigns tags to a resource annotated by a given user. *SocialIndex* represents the union of all the user personal indexes:

$$SocialIndex = \bigcup_{i=1}^{N} PersonalIndex(u_i) \qquad (2)$$

3.2 Retrieval of Similar Resources

Next, STaR can take into account users requests in order to produce personalized tag recommendations for each resource. First, every user has to provide some information about the resource to be tagged, such as the title of the Web page or its URL, in order to crawl the textual metadata associated on it.

Next, if the system can identify the user since she has already posted other resources, it exploits data about her (language, the tags she uses more, the number of tags she usually uses to annotate resources, etc.) in order to refine the query to be submitted against both the *Social* and *Personal* indexes stored in Lucene. We used as query the title of the web page (for bookmarks) or the title of the publication (for BibTeX entries). Obviously before submitting the query we processed it by deleting not useful characters and punctuation.

In order to improve the performances of the Lucene Querying Engine we replaced the original Lucene Scoring function

[6]http://lucene.apache.org

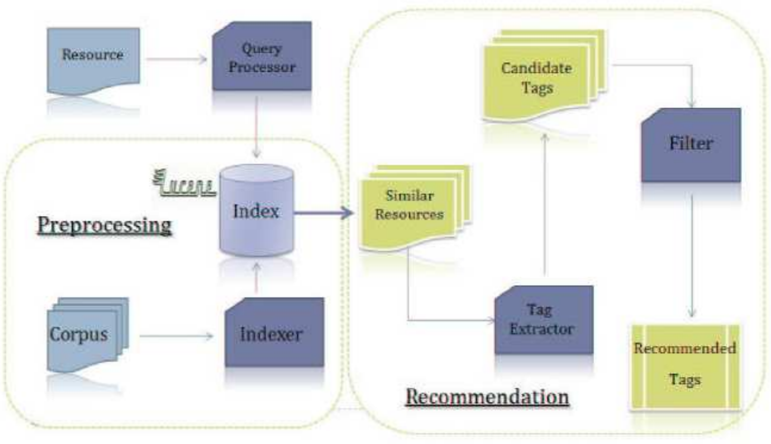


Figure 1: Architecture of STaR

with an Okapi BM25 implementation[7]. BM25 is nowadays considered as one of the state-of-the art retrieval models by the IR community [12].

Let D be a corpus of documents, $d \in D$, BM25 returns the top-k resources with the highest similarity value given a resource r (tokenized as a set of terms $t_1 \ldots t_m$), and is defined as follows:

$$sim(r,d) = \sum_{i=1}^{m} \frac{n^r_{t_i}}{k_1((1-b)+b*l)+n^r_{t_i}} * idf(t_i) \tag{3}$$

where $n^r_{t_i}$ represents the occurrences of the term t_i in the document d, l is the ratio between the length of the resource and the average length of resources in the corpus. Finally, k_1 and b are two parameters typically set to 2.0 and 0.75 respectively, and $idf(t_i)$ represents the inverse document frequency of the term t_i defined as follows:

$$idf(t_i) = log\frac{N + df(t_i) + 0.5}{df(t_i) + 0.5} \tag{4}$$

where N is the number of resources in the collection and $df(t_i)$ is the number of resources in which the term t_i occurs.

Given user $u \in U$ and a resource r, Lucene returns the resources whose similarity with r is greater or equal than a threshold β. To perform this task Lucene uses both the *PersonalIndex* of the user u and the *SocialIndex*. More formally:

$$P\_Res(u,q) = \{r \in PersonalIndex(u) | sim(q,r) \geq \beta\}$$

$$S\_Res(q) = \{r \in SocialIndex | sim(q,r) \geq \beta\}$$

Figure 2 depicts an example of the retrieving step. In this case the target resource is represented by Gazzetta.it, one of the most famous Italian sport newspaper. Lucene queries the *SocialIndex* and returns as the most similar resources an online newspaper (Corrieredellosport.it) and the official web site of an Italian Football Club (Inter.it). The

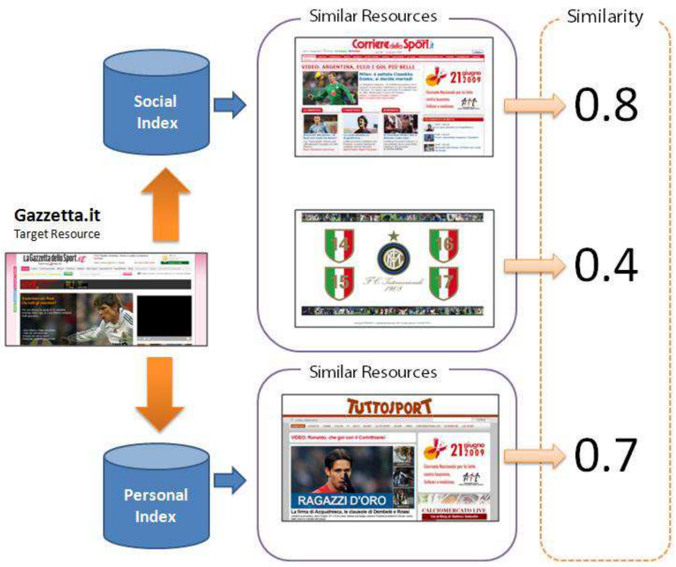


Figure 2: Retrieval of Similar Resources

PersonalIndex, instead, returns another online newspaper (Tuttosport.com). The similarity score returned by Lucene has been normalized.

3.3 Extraction of Candidate Tags

The role of the *Tag Extractor* is to produce as output the list of the so-called "candidate tags" (namely, the tags considered as 'relevant' by the tag recommender). In this step the system gets the most similar resources returned by the Apache Lucene engine and builds their folksonomies (namely, the tags they have been annotated with). Next, it produces the list of candidate tags by computing for each tag from the folksonomy a score obtained by weighting the similarity score returned by Lucene with the normalized occurrence of the tag. If the *Tag Extractor* also gets the list of the most similar resources from the user *PersonalIndex*, it will produce two partial folksonomies that are merged, assigning a weight to each folksonomy in order to boost the

[7]http://nlp.uned.es/ jperezi/Lucene-BM25/

tags previously used by the user.

Formally, for each query q (namely, the resource to be tagged), we can define a set of tags to recommend by building two sets: $candTags_p$ and $candTags_s$. These sets are defined as follows:

$$candTags_p(u,q) = \{t \in T | t = TAS(u,r) \wedge r \in P\_Res(u,q)\}$$

$$candTags_s(q) = \{t \in T | t = TAS(u,r) \wedge r \in S\_Res(q) \wedge u \in U\}$$

In the same way we can compute the relevance of each tag with respect to the query q as:

$$rel_p(t,u,q) = \frac{\sum_{r \in P\_Res(u,q)} n_r^t * sim(r,q)}{n^t} \quad (5)$$

$$rel_s(t,q) = \frac{\sum_{r \in S\_Res(q)} n_r^t * sim(r,q)}{n^t} \quad (6)$$

where n_r^t is the number of occurrences of the tag t in the annotation for resource r and n^t is the sum of the occurrences of tag t among all similar resources.

Finally, the set of *Candidate Tags* can be defined as:

$$candTags(u,q) = candTags_p(u,q) \cup candTags_s(q) \quad (7)$$

where for each tag t the global relevance can be defined as:

$$rel(t,q) = \alpha * rel_p(t,q) + (1-\alpha) * rel_s(t,q) \quad (8)$$

where α (PersonalTagWeight) and $(1-\alpha)$ (SocialTagWeight) are the weights of the personal and social tags respectively.

Figure 3 depicts the procedure performed by the *Tag Extractor*: in this case we have a set of 4 Social Tags (Newspaper, Online, Football and Inter) and 3 Personal Tags (Sport, Newspaper and Tuttosport). These sets are then merged, building the set of *Candidate Tags*. This set contains 6 tags since the tag *newspaper* appears both in social and personal tags. The system associates a score to each tag that indicates its effectiveness for the target resource. Besides, the scores for the Candidate Tags are weighted again according to SocialTagWeight (α) and PersonalTagWeight ($1-\alpha$) values (in the example, 0.3 and 0.7 respectively), in order to boost the tags already used by the user in the final tag rank. Indeed, we can point out that the social tag 'football' gets the same score of the personal tag 'tuttosport', although its original weight was twice.

3.4 Tag Recommendation

Finally, the last step of the recommendation process is performed by the *Filter*. It removes from the list of candidate tags the ones not matching specific conditions, such as a threshold for the relevance score computed by the Tag Extractor. Obviously, the value for the threshold and the maximum number of tags to be recommend is strictly dependent from the training data.

Formally, given a user $u \in U$, a query q and a threshold value γ, the goal of the filtering component is to build $rec(u,q)$ defined as follows:

$$rec(u,q) = \{t \in candTags(u,q) | rel(t,q) > \gamma\}$$

Table 1: Results comparing the Lucene original scoring function with BM25

Scoring	Resource	Pr	Re	F1
Original	bookmark	25.26	29.67	27.29
BM25	bookmark	25.62	36.62	30.15
Original	bibtex	14.06	21.45	16.99
BM25	bibtex	13.72	22.91	17.16
Original	overall	16.43	23.58	19.37
BM25	overall	**16.45**	**26.46**	**20.29**

In the example in Figure 3, setting a threshold $\gamma = 0.20$, the system would suggest the tags *sport* and *newspaper*.

4. EXPERIMENTAL EVALUATION

We designed two different experimental sessions to evaluate the performance of the tag recommender. In the first session we performed a comparison between the original scoring function of Lucene and a novel BM25 implementation, while the second was carried out to tune the system parameters.

4.1 Description of the dataset

We designed the experimental evaluation by exploiting a dataset gathered from Bibsonomy. It contains *263,004 bookmark* posts and *158,924 BibTeX* entries submitted by 3,617 different users. For each of the 235,328 different URLs and the 143,050 different BibTeX entries were also provided some textual metadata (such as the title of the resource, the description, the abstract and so on).

We evaluated STaR by comparing the real tags (namely, the tags a user adopts to annotate an unseen resource) with the suggested ones. The accuracy was finally computed using classical IR metrics, such as Precision, Recall and F1-Measure. *Precision* (Pr) is defined as the number of relevant recommended tags divided by the number of recommended tags. *Recall* (Re) is defined as the number of relevant recommended tags divided by the total number of relevant tags available. The F1-measure is computed by the following formula:

$$F1 = \frac{(2 * Pr * Re)}{Pr + Re} \quad (9)$$

4.2 Experimental Session 1

Firstly, we tried to evaluate the predictive accuracy of STaR comparing difference scoring function (namely, the Lucene original one and the aforementioned BM25 implementation). We performed the same steps previously described, retrieving the most similar items using the two mentioned similarity functions and comparing the tags suggested by the system in both cases. Results are presented in Table 1.

In general, there is an improvement by adopting BM25 with respect to the Lucene original similarity function. We can note that BM25 improved the both the recall (+ 6,95% for bookmarks, +1,46% for BibTeXs entries) and the F1 measure (+ 2,86% for bookmarks, +0,17% for BibTeXs entries).

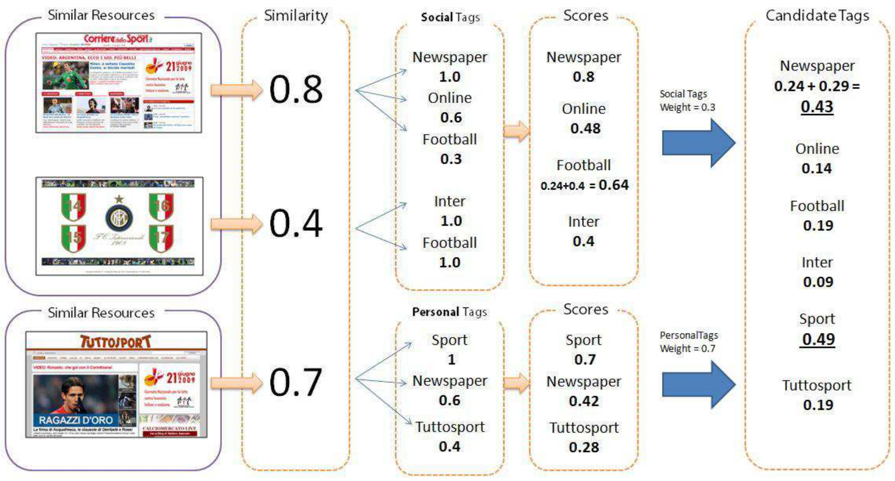


Figure 3: Description of the process performed by the Tag Extractor

4.3 Experimental Session 2

Next we designed a second experimental evaluation in order to compare the predictive accuracy of STaR with different combinations of system parameters. Namely:

- the maximum number of similar documents retrieved by Lucene;
- the value of α for the *PersonalTagWeight* and *SocialTagWeight* parameters;
- the threshold γ to establish whether a tag is relevant;
- which fields of the target resource use to compose the query;
- the best scoring function between Lucene standard one and Okapi BM25.

First, tuning the number of similar documents to retrieve from the *PersonalIndex* and *SocialIndex* is very important, since a value too high can introduce noise in the retrieval process, while a value too low can exclude documents containing relevant tags. By analyzing the results returned by some test queries, we decided to set this value between 5 and 10, depending on the training data.

Next, we tried to estimate the values for *PersonalTagWeight* (PTW) and the *SocialTagWeight* (STW). An higher weight for the Personal Tags means that in the recommendation process the systems will weigh more the tags previously used by the target user, while an higher value for the Social Tags will give more importance to the tags used by the community (namely, the whole folksonomy) on the target resource. These parameters are biased by the user practice: if tags often used by the user are very different from those used from the community, the PTW should be higher than STW. We performed an empirical study since it is difficult to define the user behavior at run time. We tested the system setting the parameters with several combinations of values:
i) PTW = 0.7 STW = 0.3;
ii) PTW = 0.5 STW = 0.5;
iii) PTW = 0.3 STW = 0.7.
Another parameter that can influence the system performance is the set of fields to use to compose the query. For each resource in the dataset there are many textual fields, such as title, abstract, description, extended description, etc. In this case we used as query the title of the webpage (for bookmarks) and the title of the publication (for BibTeX entries).

The last parameter we need to tune is the threshold to deem a tag as relevant (γ).We performed some tests suggesting both 4 and 5 tags and we decided to recommend only 4 tags since the fifth was usually noisy. We also fixed the threshold value between 0.20 and 0.25.

In order to carry out this experimental session we used the aforementioned dataset both as training and test set. We executed the test over 50,000 bookmarks and 50,000 BibTeXs. For each resource randomly chosen from the dataset and for each combination of parameters, we executed the following steps:

- query preparation;
- Lucene retrieval function invocation;
- building of the set of Candidate Tags;
- comparing the recommended tags with the real tags associated by the user;
- computing of Precision, Recall, and F1-measure.

Results are presented in Table 2 and Table 3.

Analyzing the results (see Figure **??**), it emerges that the approach we called *user-based* outperformed the other ones. In this configuration we set PTW to 1.0 and STW to 0, so we suggest only the tags already used by the user in tagging similar resources. No query was submitted against the *SocialIndex*. The first remark we can make is that each user has her own mental model and her own vocabulary: she usually prefers to tag resources with labels she already used. Instead, getting tags from the *SocialIndex* only (as proved by the results of the community-based approach) often introduces some noise in the recommendation process. The hybrid approaches outperformed the community-based one, but their predictive accuracy is still worse when compared with the user-based approach. Finally, all the approaches

Table 2: Predictive accuracy of STaR over 50,000 bookmarks

Approach	STW	PTW	Pr	Re	F1
Comm.-based	1.0	0.0	23.96	24.60	24.28
User-based	0.0	1.0	**32.12**	**28.72**	**30.33**
Hybrid	0.7	0.3	24.96	26.30	25.61
Hybrid	0.5	0.5	24.10	25.16	24.62
Hybrid	0.3	0.7	23.85	25.12	25.08
Baseline	-	-	35.58	10.42	16.11

Table 3: Predictive accuracy of STaR over 50,000 BibTeXs

Approach	STW	PTW	Pr	Re	F1
Comm.-based	1.0	0.0	34.44	35.89	35.15
User-based	0.0	1.0	**44.73**	**40.53**	**42.53**
Hybrid	0.7	0.3	32.31	38.57	35.16
Hybrid	0.5	0.5	32.36	37.55	34.76
Hybrid	0.3	0.7	35.47	39.68	37.46
Baseline	-	-	42.03	13.23	20.13

outperformed the F1-measure of the baseline. We computed the baseline recommending for each resource only its most popular tags. Obviously, for resources never tagged we could not suggest anything.

This analysis substantially confirms the results we obtained from other studies performed in the area of the tag-based recommendation [1].

5. CONCLUSIONS AND FUTURE WORK

Collaborative Tagging Systems are powerful tools, since they let users to organize the information in a way that perfectly fits their mental model. However, typical drawbacks of collaborative tagging systems represent an hindrance, since the complete tag space is too noisy to be exploited for retrieval and filtering task. So, systems that assist users in the task of tagging speeding up the tag convergence are more and more required. In this paper we presented STaR, a social tag recommender system. The idea behind our work was to discover similarity among resources in order to exploit communities and user tagging behavior. In this way our recommender system was able to suggest tags for users and items still not stored in the training set. The experimental sessions showed that users tend to reuse their own tags to annotate similar resources, so this kind of recommendation model could benefit from the use of the user personal tags before extracting the social tags of the community (we called this approach user-based). Next, we showed that the integration of a more effective scoring function (BM25) could also improve the overall accuracy of the system.

This approach has a main drawback, since it cannot suggest any tags when the set of similar items returned by Lucene is empty. So, we plan to extend the system in order to extract significant keywords from the textual content associated to a resource (title, description, etc.) that has not similar items, maybe exploiting structured data or domain ontologies. Furthermore, since tags usually suffer of typical Information Retrieval problem (namely, polysemy, synonymy, etc.) we will try to establish if the integration of Word Sense Disambiguation tools or a semantic representation of documents could improve the performance of recommender. Another issue to analyze is the application of our methodology in different domains such as multimedia environment. In this field discovering similarity among items just on the ground of textual content could be not sufficient. Finally, we will perform also some studies in the area of tag-based recommendation, investigating the integration of tag recommenders for recommendations tasks, since reaching more quickly the tag convergence could help to build better folksonomies and to produce more accurate recommendations.

6. REFERENCES

[1] P. Basile, M. de Gemmis, P. Lops, G. Semeraro, M. Bux, C. Musto, and F. Narducci. FIRSt: a Content-based Recommender System Integrating Tags for Cultural Heritage Personalization. In P. Nesi, K. Ng, and J. Delgado, editors, *Proceedings of the 4th International Conference on Automated Solutions for Cross Media Content and Multi-channel Distribution (AXMEDIS 2008) - Workshop Panels and Industrial Applications, Florence, Italy*, Firenze University Press, pages 103–106, November 17-19, 2008.

[2] C. H. Brooks and N. Montanez. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 625–632, New York, NY, USA, 2006. ACM Press.

[3] C. Cattuto, C. Schmitz, A. Baldassarri, V. D. P. Servedio, V. Loreto, A. Hotho, M. Grahl, and G. Stumme. Network properties of folksonomies. *AI Communications*, 20(4):245–262, December 2007.

[4] S. Golder and B. A. Huberman. The Structure of Collaborative Tagging Systems. *Journal of Information Science*, 32(2):198–208, 2006.

[5] Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. Extracting key terms from noisy and multi-theme documents. In *18th International World Wide Web Conference*, pages 651–661, April 2009.

[6] P. Heymann, D. Ramage, and H. Garcia-Molina. Social tag prediction. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 531–538, New York, NY, USA, 2008. ACM.

[7] R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in folksonomies. In Alexander Hinneburg, editor, *Workshop Proceedings of Lernen - Wissensentdeckung - Adaptivit?t (LWA 2007)*, pages 13–20, September 2007.

[8] M. Lipczak. Tag recommendation for folksonomies oriented towards individual users. In *Proceedings of ECML PKDD Discovery Challenge (RSDC08)*, pages 84–95, 2008.

[9] L. B. Marinho and L. Schmidt-Thieme. Collaborative tag recommendations. pages 533–540. 2008.

[10] A. Mathes. Folksonomies - cooperative classification

and communication through shared metadata. Website, December 2004. http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html.

[11] G. Mishne. Autotag: a collaborative approach to automated tag assignment for weblog posts. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 953–954, New York, NY, USA, 2006. ACM.

[12] S. E. Robertson, S. Walker, M. H. Beaulieu, A. Gull, and M. Lau. Okapi at trec. In *Text REtrieval Conference*, pages 21–30, 1992.

[13] G. Salton. *Automatic Text Processing.* Addison-Wesley, 1989.

[14] C. Schmitz, A. Hotho, R. Jäschke, and G. Stumme. Mining association rules in folksonomies. In V. Batagelj, H.-H. Bock, A. Ferligoj, and A. Őiberna, editors, *Data Science and Classification (Proc. IFCS 2006 Conference)*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 261–270, Berlin/Heidelberg, July 2006. Springer. Ljubljana.

[15] S. Sood, S. Owsley, K. Hammond, and L. Birnbaum. TagAssist: Automatic Tag Suggestion for Blog Posts. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM 2007)*, 2007.

[16] M. Tatu, M. Srikanth, and T. D'Silva. Rsdc'08: Tag recommendations using bookmark content. In *Proceedings of ECML PKDD Discovery Challenge (RSDC08)*, pages 96–107, 2008.

[17] T. Vander Wal. Folksonomy coinage and definition. Website, Februar 2007. http://vanderwal.net/folksonomy.html.

[18] H. Wu, M. Zubair, and K. Maly. Harvesting social knowledge from folksonomies. In *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*, pages 111–114, New York, NY, USA, 2006. ACM Press.

Social Trust as a solution to address sparsity-inherent problems of Recommender systems

Georgios Pitsilis
Q2S, NTNU
O.S. Bragstads plass 2E
NO-7491, Trondheim, Norway
+47 735 92743
pitsilis@q2s.ntnu.no

Svein J. Knapskog
Q2S, NTNU
O.S. Bragstads plass 2E
NO-7491, Trondheim, Norway
+47 735 94328
knapskog@q2s.ntnu.no

ABSTRACT

Trust has been explored by many researchers in the past as a successful solution for assisting recommender systems. Even though the approach of using a web-of-trust scheme for assisting the recommendation production is well adopted, issues like the sparsity problem have not been explored adequately so far with regard to this. In this work we are proposing and testing a scheme that uses the existing ratings of users to calculate the hypothetical trust that might exist between them. The purpose is to demonstrate how some basic social networking when applied to an existing system can help in alleviating problems of traditional recommender system schemes. Interestingly, such schemes are also alleviating the cold start problem from which mainly new users are suffering. In order to show how good the system is in that respect, we measure the performance at various times as the system evolves and we also contrast the solution with existing approaches. Finally, we present the results which justify that such schemes undoubtedly work better than a system that makes no use of trust at all.

Keywords

Recommender Systems, Trust Modeling, data sparsity problem Cold-Start problem, Social network.

1. INTRODUCTION

Services offered by recommender systems tend to be hosted in centralized systems. Beside the benefit that is offered in terms of easiness in managing the resources and the availability of the services, there are issues with regard to how much the new users can receive the benefits of their participation in the system. As new users we consider those who have not contributed enough data to the system and hence makes it difficult for predictions for them to be made. Similarly, the same problem seems to exist with items which the users have not have much experience yet. Recommender system services may be offered by social networking platforms like FaceBook [1] and web-pages like dealtime.com [2] or Amazon [3] and may be using mechanisms of User-Based recommender systems for working out predictions for items that users potentially like.

Any potential solution for alleviating the data sparsity issue should not work at the expense of performance of such system but instead it should provide some substantial benefits to users during their bootstrapping. The inability of new users in supplying sufficient quantities of data to the system for predictions to be computed accurately is described in the literature as *"cold-start problem"*[4].

Our approach for overcoming the above problem is based on the idea of extending the neighboring base of new users so that they can be correlated with more participants, not necessarily linked directly with each other via similarity relationships, but been discovered via "friends" as trustworthy for contributing useful data. The trust for them can be inferred via their similar, and hence common, neighbors in a scheme that is known as 'web-of-trust'. In this way, due to the propagation characteristics of trust, it is plausible that similarity between entities that could not be linked previously is becoming exploitable. Users who can be discovered via friends-of-friends might be useful as they may carry valuable experience about some product that is of interest to somebody else. As trust is mainly used for extending the number of relationships between people, users can now cooperate with more participants than before and thus get access to more recommendations. For short we call our system 'hybrid' as it combines both trust networks and traditional recommender systems approaches. We used a framework called "Subjective Logic" for the reasoning of the virtual trust relationships, and since the adaptation to existing recommender system models is a key issue, we used a model that we built our own for inferring trust from the existing users' experiences.

The evaluation we present shows the twofold benefit of this approach as the accomplished reduction of sparsity is accompanied by improved performance. To show the improvement with regard to the *cold-start* problem we demonstrate how some performance metrics evolve as the user community is growing. The rest of the paper is organized as follows: Section 2 is referred to the description of the problem. In section 3 we present the idea and the logical reasoning behind it. An evaluation of the idea follows in section 4, analysis of work that has been done in the area there is in section 5 and finally we conclude with a discussion of the results.

2. PROBLEM STATEMENT

The sparsity-inherent problems of recommender systems are related to the fact that a satisfactory number of inferences for either users or items can not be extracted, due to lack of gathered information.

User-Based recommender systems that employ a technique called Collaborative filtering (CF) (in which the user preference for some item is computed upon the similarities between the users), mainly use Resnicks's [10] formula (Equation. 1) for working out predictions $r_{a,}(i)$ for user a and items i. With $w_{a,u}$ is denoted the Pearson's similarity of user a with user u, and r_u the rating of user u for the item that is of interest to a. The formula does not give satisfactory accuracy when sparse datasets are used, as the predictions are highly sensitive to the number of similar participants, n.

$$r_a(i) = \bar{r}_a + \sum_{u=1}^{n} w_{a,u}\left(r_u(i) - \bar{r}_u\right) \qquad (1)$$

Even though the exploitation of social networks has been recognized as a potential solution for addressing such problems [18], the applied solutions, at least as it seems, do not fulfill this requirement completely. For example, in Epinions.com [17], a well known commercial recommender system, the formation of the trust network is done explicitly by asking users themselves to express whom they trust. In our opinion that is very unproductive for two reasons: first, because not all users are familiar with the notion of trust and hence they are unable to explicitly express whom they trust, and second because people, as it happens for item ratings, are unwilling to invest much time and effort in contributing with their opinions. The latter is considered as the main reason for having sparse datasets.

The *cold-start* problem has been approached in the past from the aspect of being a problem in social networking and recommender systems. In [20] it is suggested that special criteria should be used for deciding to whom it is best for the new users to connect. The decision in this case is based upon users' so far objective assessment of candidates for their suitability and it is decided on how active they have been in contributing data to the system. Nevertheless, such a decision is based solely on quantitative criteria and in our opinion it would be best if qualitative criteria (such as how useful such recommendations have been found to be by the cold start users) were also used. In addition, in the above mentioned solutions the social trust is not adequately exploited, as the discovery for trustworthy participants is usually done by applying local criteria.

It is crucial that when a new idea is applied to an existing system for enhancing its performance it is done in such a way that adapts best to it. Therefore in our model, no additional data should be required to be supplied by users, but the inferred trust should be implicitly derived from the existing evidence instead. This procedure is also useful from the practical aspect, since by doing so the existing recommendation production cycle would not need to change significantly to include the benefits that *social networking* provides. Similar ways of implicit derivation of user's trust from evidence has been proposed for other purposes before, like the EigenTrust algorithm [21] that was mainly build for peer-to-peer systems. However, the trust in that case was essentially perceived as a global reputation value due to being independent on the point of view.

To our knowledge, the use of trust networks for alleviating sparsity-inherent problems, such as the *cold-start* problem in recommender systems have not been adequately studied so far.

With regard to trust models and frameworks, there are many developed so far, most of them mainly to meet special requirements of particular problems, and other more generic ones such as *Subjective Logic* [11]. We mention this framework explicitly because of the substantial adoption it has received for studying the effects of trust propagation in user communities.

3. DESCRIPTION OF THE APPROACH

User-based recommender systems that use Resnick's formula are limited to computing predictions of ratings for users whose similarity $w_{a,u}$ with the all contributing participants is known. The limited number of neighbors that can contribute during the system bootstrapping is a significant constraint for achieving good performance during the early stages of the recommender system's life.

Since the accuracy of predictions that the querying user receives is dependent on the number of neighbors/predictors that appear to be similar to him/her, it means that a substantial improvement can be achieved if multiple predictors could be involved in the computation of $r_{a,}(i)$. As new users do not have enough experiences to contribute during the bootstrapping it means the performance would be sub-optimal as there would not be enough links between them.

One characteristic pitfall of a conventional recommendation system mechanism is the inability to incorporate prediction ratings of other participants who have experienced some item that is of interest to the querying agent, but their similarity with the querying agent cannot be inferred.

If a recommender system were to be represented graphically, the similar users would appear to be within a distance of one hop away from the querying user. Exploitation of information that resides at longer distances would be plausible if similarity could have propagative characteristics. Since trust is known for providing such property, which is the key idea of social networks, it means inferring trust from similarity could make it possible to overcome the above limitation. Pursuing this idea further, the neighboring base of users could be extended beyond the one hop range by introducing a hybrid system in which the similarity could be inferred from trust.

The above requirement for predicting the rating that some user i would give to item b can be expressed as follows:

$$\begin{aligned} b \in B \mid & \left\{\exists a_j \in n(a_i) : r_i(b) = \perp \wedge r_j(b) = \perp \right\} \\ & \wedge \left\{\exists a_k \in n(a_j) : r_k(b) \neq \perp \right\} \end{aligned} \qquad (2)$$

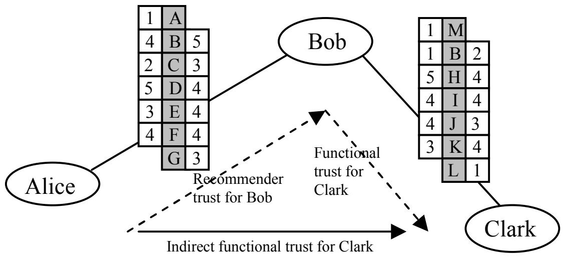
where B is the set of all items in the system, i,j,k are 3 users, a is the set of users and $n()$ is a function that denotes a similar neighbor to i.

In a typical scenario of operation of a User-Based recommender system (see figure 1[1]) we assume that Alice and Bob have both experienced a number of items B_a and B_b respectively and hence it can be known how similar they are. On the other hand Clark has another set of common experiences with Bob, and hence it can also be known how similar Bob is with him. However, Clark's experience about some new item that might be of interest to Alice, is not exploitable in the conventional system (Alice's similarity to Clark's is not computable).

Extending this consideration and inferring trust implicitly from the calculated similarity for every pair of similar users, then

[1] The letters A…L represent the items rated by users and the numbers 1…5 the rates given.

finally a web-of-trust for social partners linked together can be developed.



1 A
4 B 5
2 C 3
5 D 4
3 E 4
4 F 4
G 3
Bob
1 M
1 B 2
5 H 4
4 I 4
4 J 3
3 K 4
L 1
Alice
Recommender trust for Bob
Functional trust for Clark
Clark
Indirect functional trust for Clark

Figure 1. A typical recommendation production.

One important issue that comes from the implicit derivation of trust is concerned with the representation of the existing evidence into trust metrics. For that reason some appropriate mapping is necessary.

3.1 Trust Modeling

In socially aware systems, users benefit from their trust and connections with others as they can find people they need through the people they trust. Links to a person imply some amount of trust for this person. The importance of social networks is found in the exploitation of such network data to produce information about trust between individuals which have no direct network connection. In theory about trust, this requirement is described with a property called *transitivity* which we are attempting to exploit in this work. As trust is not perfectly transitive in the mathematical sense, however it can be useful in the way like in the real world people consider recommendations from others they trust for their choices. Since trust that derives though recommendations is dependent on the point of view it means it is merely subjective.

In contrast to similarity, trust relationships can be propagated transitively throughout the network of users. In this way the common neighbors can act as trustworthy participants for users of whom similarity is not known, but can be approximated via their derived trust. The concept of computing the indirect trust for distant entities requires the employment of some suitable algebra such as *Subjective Logic*. However, it is required that evidence has first been transformed into some form that the trust algebra can use.

Trust in subjective logic is expressed in a form that is called *opinion* and is referred to a metric that originally was introduced in Uncertain Probabilities theory [22], an extension to probabilistic logic. An *opinion* expresses the belief about the truth of some proposition which may represent the behavior of some agent. The ownership of the opinion is also taken into account and this is what makes the assessment of trustworthiness subjective. Furthermore, this theory is suitable for modeling cases where there is incomplete knowledge. In the case of recommender systems the lack of knowledge about some agent's rating behavior comes from the fact that there are usually limited observations of the rating behavior of some person. The lack of knowledge is actually what shapes the subjective trust or distrust towards that entity. The absence of both trust and distrust in opinions is expressed by the *uncertainty* property. The subjective logic framework uses a simple intuitive representation of uncertain probabilities by using a three dimensional metric that comprises belief (b), disbelief (d) and uncertainty (u) into opinions. It is required that evidence comes in such a form that opinions $\omega_p^A = \{b_p^A, d_p^A, u_p^A\}$ about some agent *A* with regard to the proposition *p* can be derived from it, and thus be better manageable due to the quite flexible calculus that the opinion space provides. By convention the following rule holds for *b,d,u*: *b+d+u=1*.

Subjective Logic provides the following two operators: *recommendation* (3) and *consensus* (4). Both can be used for combining opinions and deriving recommendations regarding other agents in the social network.

$$\omega_p^{A,B} = \omega_p^A \oplus \omega_p^B = \{b_p^{A,B}, d_p^{A,B}, u_p^{A,B}\} \qquad (3)$$

A, B are agents and $\omega_p^B = \{b_p^B, d_p^B, u_p^B\}$ is the opinion of B about p expressed as a recommendation to A. $\omega_B^A = \{b_B^A, d_B^A, u_B^A\}$ is the opinion of A about the recommendations of B. The consensus opinion ω_p^{AB} is held by an imaginary agent AB representing both A and B.

$$\omega_p^{AB} = \omega_B^A \otimes \omega_p^B = \{b_p^{AB}, d_p^{AB}, u_p^{AB}\} \qquad (4)$$

The output values *b,d,u* of the combined opinions are derived from simple algebraic operations. More about this can be found in [11]. In our opinion the above considerations of Subjective Logic are quite sufficient for deriving recommendations with regard to the rating behavior of users whose subjective trustworthiness can be computed transitively within the social network of trusted participants.

In the literature trust is also distinguished into *direct* and *indirect trust*, the former when it is derived from personal experience of a trustor, and the latter when it is derived from recommendations of others. Also, another distinction of it is *functional trust*, which expresses the trustworthiness for some agent with regard to some proposition p, and *recommendation trust* which expresses the trustworthiness of some agent as a recommender.

In our example depicted in figure 1, Bob has *functional trust* in Clarke's rating behavior, but the trust that can be derived by Alice for Clark via Bob's recommendation trust is *indirect trust* since Alice does not have her own evidence to support it, but merely trusts Bob's taste. Finally, the trust that Alice is interested in knowing for Clark, is actually an *indirect functional trust,* which indicates how much she would trust him for his taste.

As far as the evidence transformation is concerned, various models for converting ordinary observations into evidence have been proposed [11][19]. For our approach, we have used a simple model which is best suited to recommender system data [12]. In our work we have come up with a solution of deriving the trustworthiness that a pair of agents would place in each other by using existing data such as ratings for items they have gathered experience with. In the same work an approach for mapping trust into similarity is also introduced. This is explained below.

For calculating the uncertainty we used the simplified formula: $u = (n+1)^{-1}$, in which *n* denotes the number of common experiences in a trust relationship between two agents A and B.

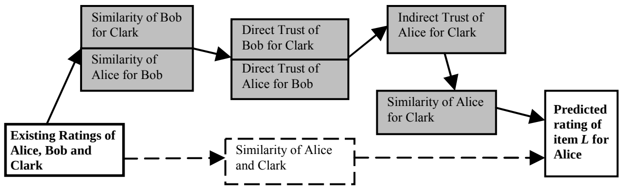
The derivation of opinions from existing user experiences with items can be done by using an appropriate formula such as the one given below which we used for our experiment. This formula was used for shaping the belief property (*b*) of *Subjective Logic* from *User Similarity* ($W_{a,u}$) also known as *Correlation Coefficient*.

$$b = \frac{1}{2}(1-u)\left(1+W_{a,u}{}^{K}\right) \qquad (5)$$

The disbelief property *d* of the opinions can easily be derived from the remainder of *b* and *u* as: *d =1-b-u*.

In our case scenario of recommender system the calculated belief *(b)* is referred to either *recommender* or *functional* trust. In equation 5 the k value denotes the exponent in the equation used for transforming the similarity metric into derived belief (k=1 for linear transformation).

In the figure below we present pictorially a high level view of our hybrid system that could take advantage of this idea. The trust derivation mechanism for predicting the ratings that users would give to products can be easily embedded into the existing ordinary system.



Similarity of Bob for Clark
Similarity of Alice for Bob
Direct Trust of Bob for Clark
Direct Trust of Alice for Bob
Indirect Trust of Alice for Clark
Similarity of Alice for Clark
Predicted rating of item L for Alice
Existing Ratings of Alice, Bob and Clark
Similarity of Alice and Clark

Figure 2. The conceptual view of our hybrid system

The part of the diagram shown in dotted line represents the existing recommendation production mechanism and it applies only if evidence suffices for computing the similarity of Alice to Clark.

4. EVALUATION

We evaluated the performance of our hybrid approach against various alternatives such as the standard CF technique which employs the Pearson similarity and uses Resnick's prediction formula [10].

The results which demonstrate the effectivenes of predicting user likeness express the ability of the system to identify potentially unsatisfactory options. Moreover, we introduce a set of metrics to demonstrate efficiency in terms of sparsity reduction as well as effectiveness against the *cold-start* problem.

For the evaluation we used a subset of a movie recommendation system called Movielens. This original dataset contains more than 1 million movie recommendations submitted during a period of 812 days by around 6000 users for 3100 movies. To capture the dynamic characteristics of performance that evolve over time, we made use of the timestamp (TS) information that is attached on every submitted rating. We divided the rating experiences into 5 sets, each containing ratings submitted within the same period of time. Finally, we performed the experiment for each of those sets separately. As the number of recommendations performed at each stage is more important to be shown than the timestamp information we considered as the best solution to present the adjacent *sparsity* value. That is the percentage of ratings in the users by items matrix for which originally no values had been provided.

To fulfill the requirement for calculations of similarity to be stable we assumed that similarity between two users is calculable only if there are at least 10 items that have been rated by both of them.

It is worth mentioning that in the current experiment, no control has been applied on filtering the number of neighbors that a user can have, and the only limiting factor for forming a trust relationship is the number of common experiences that exist.

4.1 Metrics Used

Predictive Accuracy metrics such as MAE are quite popular for measuring how close the recommender system's predicted ratings are to the true user ratings [13]. However, it is also interesting to know how good the system would be in successfully identifying the items that users would be unhappy with, and therefore we also used *Classification Accuracy* metrics. To justify this decision we claim that in the way the experiment was done, there were no rating predictions attempted for items that had no recorded rating experiences and hence the danger that one might be lead into classification errors is significantly reduced.

4.1.1 Measuring Coverage

With *Coverage* we refer to the percentage of items for which predictions can be made and in [14] it is defined as:

$$C = \frac{\sum_{a_i \in A}\left|\{b \in B \mid \exists a_j \in n(a_i) : r_j(b) \neq \perp\}\right|}{|B|\cdot|A|} \qquad (6)$$

where A and B are the set of users and products respectively, r_j is a rating function and *n(a)* denotes the set of similar neighbors of some user *a*.

We introduce one new metric, *User Coverage Gain* (UCG), to demonstrate the actual benefit that users receive when they make use of the trust graph. This metric relates the cost $|A|$ with the benefit R, expressed as a ratio of the hybrid system by the standard CF. It can be computed using the following formula:

$$UCG = \left[\frac{R_h}{|A_h|}\cdot\left(\frac{R_s}{|A_s|}\right)^{-1} - 1\right]_{TS} \qquad (7)$$

R_h and R_s refer to the number of predictions that the system was capable of performing for the new users in timestamp TS for the hybrid and the standard recommender system respectively. $|A_h|$ and $|A_s|$ refer to the sizes of populations of users which have made use of the trust network for discovering other participants at time *TS,* and correspondingly the adjacent number of users who would use the standard CF for performing those predictions. The populations of users are expressed as in formulas (8) and (9). The formula for A_h is corresponding to the scenario that trust propagation has been restricted to max distance of 2 hops only, which is the case for our experiment. The items recommended in every timestamp can be expressed as in formula (10)

$$A_s = \sum_{b \in B}\left|\{a \in A \mid \exists c \in n(a) : r_a(b) = \perp \wedge r_c(b) \neq \perp\}\right| \qquad (8)$$

$$A_h = \sum_{b\in B} \Big|\{a \in A | \exists c_k \in n(a), \exists d \notin n(a) \wedge \wedge d \in n(c_k): r_a(b) = \perp \wedge r_d(b) = \perp\}\Big| \tag{9}$$

$$\sum_{a_i \in A} \big|\{b \in B : r_i(b) \neq \perp\}\big| \tag{10}$$

For showing the level of contribution of the trusted participants to a prediction to be made, we came up with a metric called *Trust Graph Contribution.* This metric is presented in formula (11) and as can be seen it relates the relative increase in the number of users (due to the use of trust network) used for the produced recommendations, with the actual number of recommendations produced. This relative increase is expressed as the ratio of *trusted neighbors* by all neighbors (trusted and similar). With *"trusted neighbors"* we refer to those users in the social graph for which their similarity with the querying user was derived by propagated trust recommendations and was not calculated directly by correlating the common experiences (ratings) with some neighbor as it is done for the case of "similar" ones.

$$Ctrb = \left(|R|^{-1} \cdot \sum_{i\in R} \left[\frac{|A_h|}{|A_h| + |A_s|} \right] \right)_{TS} \tag{11}$$

In formula (11), A_s and A_h are the same as explained before, R is the set of recommendations produced in some timestamp and is the sum of R_h *and* R_s. This metric demonstrates how effective the discovery of neighbors of interest can be via the social network and it is interesting to know how it contributes in the performance improvement.

Beside new users, items for which there extensive experience is not available are also affected by the *cold-start* problem. Therefore, we found it necessary to measure the improvements that the hybrid solution would offer to them as well.

4.1.2 Measuring Accuracy

Predictive Accuracy is a standard metric for measuring how close the predicted ratings are to the true user ratings. In our experiment we measure the MAE (Mean Average Error) which shows the absolute deviation between the two. However, as MAE can be unimportant for showing the performance for items of interest to users, we considered also using other accuracy metrics in addition to this.

Classification Accuracy metrics are used to measure the frequency with which the system makes incorrect or correct decisions about whether an item is bad or good and it is usually applied in connection with the task of finding lists of top items.

As we mentioned previously we consider it more appropriate to demonstrate how useful the system is in helping users to avoid making choices of products that they might be unhappy with. Therefore we used the metric *F-score*, also called *Harmonic Mean,* and it is used in *information retrieval*. *F-score* measures the effectiveness of retrieval with respect to the cost of retrieving the information [13].

It is necessary that the negative (N) and positive (P) instances are clearly distinguished, and for our particular case we characterize as N the case of experiences with products that the user would be unsatisfied with and would give low rating. The opposite case corresponds to P. *Precision* is defined as: $P = \frac{TP}{TP+FP}$ and represents the ratio of instances that were correctly predicted as non-satisfactory by the user against all instances that were predicted as non-satisfactory. *Recall* is: $R = \frac{TP}{TP+FN}$ and represents the number of instances that were correctly predicted as non-satisfactory ones normalized by the total number of instances that actually received unsatisfactory rating. The *F-score that* shows the relative tradeoff between the benefits (TP) and the costs (FP) is calculated using the formula: $F = \frac{2PR}{P+R}$.

As rating values of 1 and 2 represent an unfortunate choice and 4 and 5 a successful choice, we used the value 3 as threshold for considering an experience as unsatisfactory. In table 1 we present the confusion matrix. We considered as true positives (TP) the instances that where correctly classified as receiving low rating and false negatives (FN) those instances that were classified as having high rating, but still predicted as been non-satisfactory to the users. True negatives (TN) denote those which were correctly classified as giving a high rating. Finally, false positives (FP) are referred to the number of bad items which were mistakenly classified by the system as satisfactory ones for the new users.

Table 1. Confusion Matrix

Actual \ Predicted	Predicted Value $\leq$ 2	Predicted Value > 2
Rating $\leq$ 2	TP	FP
Rating > 2	FN	TN

The evaluation was done using the cross validation technique *leave-one-out* applied on every user rating. The process was the following: every rating provided by each user was removed from the dataset and then its value was tentatively computed using the trust network. The computed and the removed value are then compared and the error was calculated. The evaluation algorithm is presented in the figure below.

```
Algorithm:   Evaluation plan

1.  for all users $i$ who have provided at least 10 ratings
2.   for all items $k$ of user $i$
3.    Pset ← ø
4.     for all users j whose common rated items with $i > 10$
5.        if ( $j$ similar to $i$ )
6.           Pset ← Pset ∪ { $j$ }
7.           Similar ← Similar +1
8.        else if ( trust of $i$ for $j$ is computable)
9.           derive similarity of $i$ for $j$ from trust of $i$ for $j$
10.           Pset ← Pset ∪ { $j$ }
11.           Trusted ← Trusted +1
12.    end for
13.    predict $k$ for over Pset
14.    calculate MAE for $k$
15.    calculate TP,FP,NF,F-score for i
16.   end for
17.  end for
18.  Trust Graph Contribution ← Trusted / (Trusted + Similar )
19.  average MAE for all $i$
```

Figure 3. The evaluation plan in pseudo-code

With regard to coverage, only those values which were possible to compute were considered for contributing to it.
To study more closely the benefits that our system can offer to the entities that are mostly affected by the cold-start problem, we repeated the experiments considering the new items and the new users alone. To achieve that for every timestamp we filtered and counted those entities which committed their first experience at that particular timestamp.

4.2 Results – Discussion

4.2.1 *Overall Performance*

With regard to the overall performance of the system, we first demonstrate the results we obtained for the *Coverage Gain* and the *Contribution of Trust Graph*. In these diagrams, time is represented by its adjacent sparsity value and is shown across the horizontal axis.

In figure 4 is shown how the *Contribution of the Trust Graph* develops over the experiment. For every recommendation produced, the ratio of trusted participants considered for this recommendation divided by all participants (trusted and similar) used for the same recommendation is counted and the results are averaged. As can be seen from the diagram this metric follows in general a decreasing trend throughout the simulation, but more importantly, it gets its maximum value towards the beginning when the trust network is still building up. We interpret this as a good indication that our system can cope well with the cold start problem as the resources of the hybrid system are shown to be exploited better at that timestamp. The decreasing trend followed afterwards is explained as the effect of gradual replacement of trust relationships by computed similarity as more and more recommendations are submitted over time. In the same figure is shown the benefit of using the hybrid system in terms of *User Coverage Gain* in comparison to using the standard Collaborative Filtering. Interestingly enough, this metric follows an increasing trend and more importantly it remains unaffected by the decreasing rate of new users.

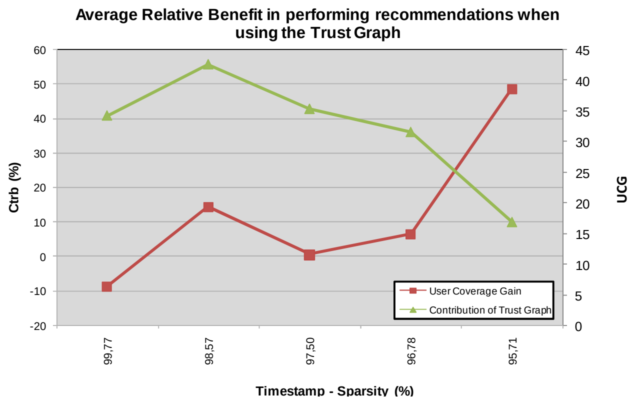


Fig. 4. The Coverage Gain & Trust graph Contribution

That is because the new users who join late get higher support as the friends-of-friends network is then denser than before.

As far as the rating accuracy is concerned the results for both the classification accuracy (F-score) and predictive accuracy (MAE) are shown in table 2 and are also graphically presented in figure 5.

For the classification accuracy, the results show that there is quite a notable advantage of our method over the standard CF in discovering those items that a user would be unhappy to choose.

Table 2. Accuracy expressed in F-score and MAE

Timestamp (sparsity) \ Method	F-score		MAE %	
	Standard	Hybrid	Standard	Hybrid
1- (99,77 %)	0,0836	0,0952	15.944	15.686
2- (98,57 %)	0,1546	0,1832	14.562	15.126
3- (97,50 %)	0,2443	0,2720	14.652	15.350
4- (97,78 %)	0,3030	0,3350	15.228	15.676
5- (95,71 %)	0,3347	0,3598	15.210	16.100

As can be seen this advantage appears from early on (first timestamp), it maximizes at the second timestamp (highest difference between the F-score values of the standard and hybrid models) when the data is still quite sparse, and continues so at all consecutive timestamps. It is interesting to note that this behavior is very unlikely to be coincidental as it appeared at all five different datasets we tested.

One can also see that the predictive accuracy appears to be higher in the proposed system than in the standard one, but here there is temporary decrease during the early timestamps.

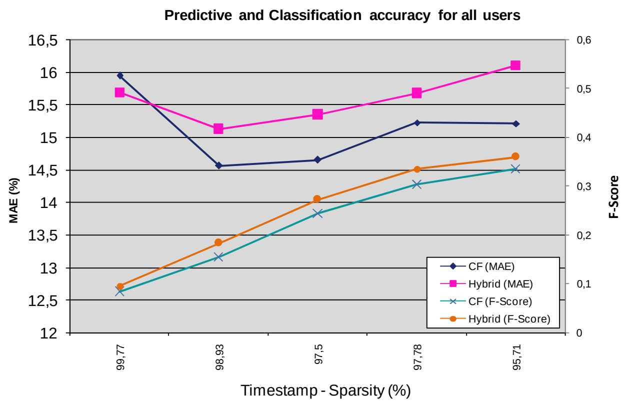


Figure 5. The accuracy for the compared schemes

4.2.2 *Selective Performance*

Next we present our results which demonstrate the behavior of the examined systems when considering only the new users and items. First we demonstrate the sizes of populations of *cold-start* users and items that appeared for the first time on each timestamp.

From the diagrams in figure 6 it can be seen that in the case of the hybrid system, the cold-start users are shown to be committing their first experience with the hybrid system earlier than in the standard CF. In the first two timestamps, the number of new users in the former is higher than in the latter. As expected though, that trend is declining as the system develops over time. This looks quite reasonable from the way our experiment was done, as in contrast to a real world running system, we used restricted size sets of 100 users.

The early emergence of new users that appears in the hybrid system is indicative of its success in exploiting the social network and performing predictions that wouldn't be possible in the conventional one, and hence attract more users. Consequently the same happens for *cold-start* items which are now discovered and rated by users earlier in the hybrid system than in the standard recommnder system.

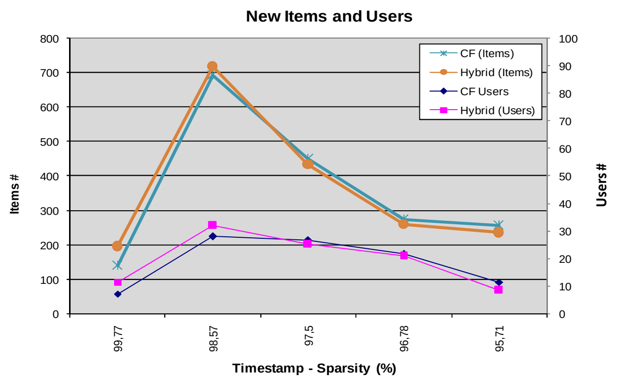


Figure. 6. The new users and items

In table 3 we present the prediction and classification accuracy for *cold-start* users and items and the results are pictorially presented in figures 7 and 8. As far as prediction accuracy is concerned, from the results it can be seen that the deployment of the trust system into the existing one has no impact on the accuracy of ratings prediction, as the error is kept low (below 15%) during the early stages of the system. For the new items the situation looks quite a lot better as there is no noticeable penalty in the prediction error against using the standard recommender system. In comparison to the overall performance results of figure 5, the new users receive higher benefit (MAE 14.87%) than the average user (MAE 15.13%) during the early timestamps (at TS:2). However, this benefit is diluted as the time progresses. Instead, new users loose this advantage if using the standard system. (MAE:14.56% opposed to 14.58%).

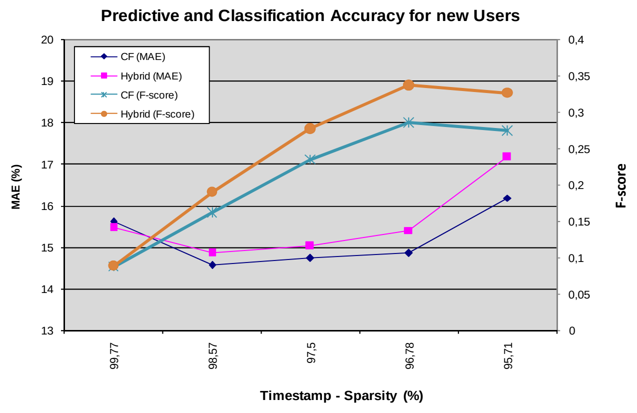


Figure. 7. The benefit of hybrid on new users

Finally, it is also important to note the increasing trend in the average error as seen in fig. 7 which means that new users who join the system late are less likely to receive good service than those who join early.

Regarding classification accuracy for new users and new items, our measurements show that the proposed hybrid system outperforms the traditional one at all time instances.

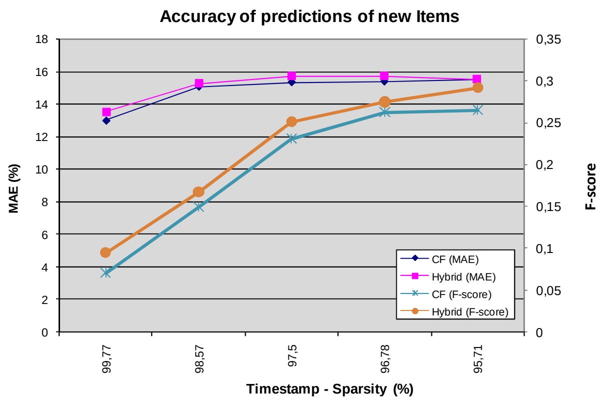


Figure. 8. The benefit of hybrid system on new items

The growing trend for F-score as it appears in fig. 7 is indicative of the increasing benefit that new users receive as the system develops. In comparison to the classification performance for all users presented in fig. 5, the new users receive higher benefit than anyone else as they are potentially guided better to avoid products they will be unhappy with.

We consider the above two observations as a positive consequence for the proposed system for compensating the users early on. It is important that new users receive the highest benefit as they are assumed to be less tolerant in receiving poor recommendations. Experiencing poor recommendations consistently over time may reduce their trust towards the system and make them reluctant to rely on it for delivering good service. If the original trust disappears, the users interest in using the system may vanish altogether.

Table 3. Accuracy for new Users and new Items.

Predictive accuracy in MAE				
Method Timestamp (sparsity)	New Users		New Items	
	Standard	Hybrid	Standard	Hybrid
1 - (99,77 %)	15.62	15.8	13.00	13.52
2 - (98,57 %)	14.58	14.87	15.06	15.27
3 - (97,50 %)	14.75	15.04	15.32	15.71
4 - (97,78 %)	14.86	15.42	15.37	15.72
5 - (95,71 %)	16.19	17.18	15.52	15.51
Classification accuracy in F-score				
1 - (99,77 %)	0.088	0.090	0.071	0.095
2 - (98,57 %)	0.162	0.190	0.149	0.167
3 - (97,50 %)	0.235	0.277	0.230	0.250
4 - (97,78 %)	0.286	0.337	0.262	0.274
5 - (95,71 %)	0.275	0.326	0.265	0.291

5. BACKGROUND RESEARCH

Trust has been the subject of investigation by many researchers in the past for alleviating issues connected with the use of sparse datasets in recommender systems. Singular Value Decomposition has been proposed by other researchers and found to be better than the standard collaborating filtering [5] for alleviating sparsity problems. Other approaches are based on the idea of removing global effects and estimating the interpolation weights for each weighting factor for improving the accuracy of recommender systems [6]. Hybrid systems which combine content and collaboration have also been proposed in which various weights are set on the contribution of similarity [7]. In such an approach, the weight is dependent on the number of common items. In [15], O'Donovan and Smyth study the effects of using trust models in the recommendation process and they demonstrate how it behaves against various attack scenarios. In [8], a solution for computing trust in CF systems has been investigated, but in the proposed model the trustworthiness of the recommender is not taken into account. In [9], in the work done by Lathia et.al., it is suggested that collaboration groups could better be formed by k-trusted neighbors rather than k-similar ones. In [16], the cold-start problem is approached using some idea based on machine learning. Massa et al. in [23] has published a similar idea with ours, but based on different working hypothesis which requires that users would provide the trust statements themselves. To our knowledge trust has not been studied adequately so far as a solution to the cold-start problem. In addition, even though all the studies performed can demonstrate the advantages of using trust, they are merely static and do not capture the characteristics of the community as it evolves. Since the cold-start problem is a time related issue we chose to demonstrate our proposed solution in a way that it can be shown if the advantage actually becomes available when the system needs it the most.

6. CONCLUSION

We have proposed a hypothetical hybrid recommender system which uses trust to exploit the latent relationships between users and we have measured its performance. In this way, also knowledge that exists at distant participants can be discovered and used by users who do not need to be known to each other. We used our modeling technique to build trust from existing evidence. The evaluation results show a significant benefit against the standard technique both in terms of coverage and in accuracy of predictions. It is interesting to note that the benefit is more distinguishable for new users and items which traditionally are mostly affected by the sparsity problem. Furthermore, the higher values achieved for F-score are indicative of improved ability in protecting users from choosing products that they may not like. With regard to the challenge of alleviating the cold-start problem, it can be seen that the benefits of using the trust enabled system are particularly visible early on when they are actually needed. A future challenge is to extend even further the period of time that the benefit is received.

REFERENCES

[1] Facebook Social Network service, http://www.facebook.com

[2] General Consumer Review Site, http://www.dealtime.com

[3] Electronic Commerce Company, http://www.amazon.com

[4] Maltz D., Ehrlish K., Pointing the Way: Active Collaborative filtering, In proc. of CHI-95,pp.202-209,New York, ACM Press (1995)

[5] Sun X., Kong F., Ye S., A Comparison of Several Algorithms for Collaborative Filtering in Startup Stage, In proc Networking Sensing and Control, IEEE, pp.25-28 (2005)

[6] Bell R., Koren Y., Improved Neighborhood-based Collaborative Filtering, In proc IEEE International Conference on Data Mining, pp.7–14 (2007)

[7] Melville P., Mooney R.L. , Nagarajan R., Content-Boosted Collaborative Filtering for Improved Recommendations, In proc of Eighteenth national conf. of Artificial Intelligence, pp.187-192, ISBN:0-262-51129-0 (2002)

[8] Quercia D., Heiles S., Carpa L., B-trust: Bayesian Trust Framework for Pervasive Computing. In proc 4th International Conf. iTrust 2006. Lecture Notes in Computer Science (Vol.3986/2006). Springer, pp. 298-312 (2006)

[9] Lathia N., Hailes S., Carpa L., Trust-Based Collaborative Filtering, in proc IFIPTM, Springer, Vol. 263, pp.119-134, Trondheim, Norway (2008)

[10] Resnick P., Varian H.R., Recommender Systems, Communications of the ACM. 40(3), pp. 56-58 (1997)

[11] Jøsang A., A Logic for Uncertain probabilities, International Journal of Uncertainty, fuzzi-ness & Knowledge based systems,Vol.9, No.3 (2001)

[12] Pitsilis G., Marshall L.F., Modeling Trust for Recommender Systems Using Similarity Metrics, in proc. IFIPTM, Springer, Vol. 263, pp.103-118, Trondheim, Norway (2008)

[13] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. 2004. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.* 22, 1 (Jan. 2004)

[14] Ziegler C-N., Towards Decentralized Recommender Systems, ISBN 363901149X, Vdm-Verlag, (2008)

[15] O'Donovan J., Smyth B. ,Is trust Robust?: An Analysis of Trust-Based Recommendation, in Proc. of 11th International Conf. on Intelligent User Interfaces IUI '06, pp.101-108, (2006).

[16] Xuan N. L., Thuc V., Trong D. L., Anh D. D., Addressing cold-start problem in recommendation systems, in Proc. of the 2nd international conf. on Ubiquitous information management and communication, Jan.31-Feb.01, Suwon, Korea, (2008).

[17] General consumer review site. http://www.epinions.com,

[18] Golbeck J., Hendler J.. Inferring Trust Relationships in Web-Based Social Networks, *ACM Transactions on Internet Technology,* 6(4). (2006)

[19] Josang A.,Bhuiyan T.,Xu Y.,Cox C., Combining Trust and Reputation Management for Web-Based Services, in Proc. of the 5th international conf. on Trust, Privacy and Security in Digital Business,Turin, Italy, pp.90-99 (2008)

[20] Victor P., Cornelis C., Teredesai A. M., De Cock M., Whom should I trust?: the impact of key figures on cold start recommendations , In proc. SAC '08: ACM symposium on Applied computing, pp. 2014-2018. (2008),

[21] Kamwar S. D, Schlosser M. T, Hector Garcia-Molina. "The EigenTrust algorithm for reputation management in P2P networks". In: Proceedings of the 12th International Conference on World Wide Web, Budapest, Hungary, 640-651 (2003)

[22] Shafer G., A Mathematical Theory of Evidence, Princeton University Press (1976)

[23] Massa P., Avesani P.,Trust Aware Bootstrapping of Recommender Systems, in proc. ECAI Workshop on Recommender Systems (2006).

Ontology Guided Dynamic Preference Elicitation

Gil Chamiel and Maurice Pagnucco
School of Computer Science and Engineering
The University of New South Wales and NICTA
NSW, Sydney 2052, Australia
{gilc|morri}@cse.unsw.edu.au

ABSTRACT

A challenge for preference based recommender systems is to elicit user preferences in an accurate and efficient manner. Eliciting preferences from the user in the form of a query that is then used to filter items from a database can result in a coarse recommendation with numerous results returned. The problem lies in the user's knowledge concerning the items among which they are searching. Unless the user is a domain expert, their preferences are likely to be expressed in a vague manner and so vague results (in the form of irrelevant alternatives) are returned. On the other hand, the advent of the world wide web has delivered an abundance of data at our fingertips. Information gathered from the web, reduced to structured ontologies, can prove useful in focussing preference elicitation mechanisms.

In this paper we present a preference elicitation process which allows users to communicate their preferences in a simple manner, through examples presented to them. The system then makes use of an ontology model, based on expert information and social web resources. It elicits the user's preferences guided by this ontology in an interactive and dynamic manner. We show that this leads to more effective recommendations.

We evaluate our work through empirical experiments and discuss the results in terms of preference elicitation coverage as well as the prediction accuracy of the preference model.

1. INTRODUCTION

Modelling user preferences and exploiting preferential information to assist users in searching for items has become an important issue in product recommendation. Eliciting user preferences in an accurate manner is a difficult and challenging task. In most cases the user lacks deeper, expert knowledge of the domain to allow for a more discriminating recommendation to be determined but they know what they like. Furthermore, even if they are aware of some of their preferences, it may be difficult for them to express these explicitly in a formal language.

In this paper we develop techniques for eliciting formal preferences from the user in a seamless fashion that hides the technical details. In our work a crucial desiderata is that the user does not need to know anything about the attributes that describe items. We focus on the problem of determining the most appropriate queries to present to the user in order to accurately elicit their preferences. We do so in an interactive manner which focuses on the user experience by utilising ontological information available on the World Wide Web through social web resources and expert libraries. By so doing we develop a complete system for personalisation that cushions the user from having to know the formal details of how preferences are represented and how preference queries over a database of items are formulated. We define four types of preference elicitation queries and show how the dynamics of these methods are designed to gather sufficient information from the user to quickly and accurately determine their preferences. This is the main contribution of the paper. We also show how these processes can be achieved more efficiently by clustering the item space. Finally, we briefly discuss how we use standard statistical methods together with these ideas in order to establish the user preference profile.

Research on personalisation has provided a rich literature in recommendation systems [1, 2, 3]. In more direct relation to preference elicitation, [4] provides a significant framework for formalising an elicitation process. However, this framework assumes user familiarity with the domain which contradicts the assumption of this research. [5, 6] model preference elicitation in terms of utility elicitation but these processes are not directly guided by a model of the data. Previous work in supplementing user preferences with ontological information is limited. Ontology based similarity systems have been presented in [7, 8] but provide for only basic features. [9] provide methods for augmenting collaborative product recommendation with information derived from taxonomies. However, none of these approaches have tackled the entire problem of eliciting formal preferences from a user and enhancing them with ontological information in an interactive manner. It is this problem that we address here in its entirety.

1.1 Motivating Example

One of the most obvious domains in which to evaluate the techniques developed here is that of user musical preferences. When it comes to music selection, people often express their preferences in terms of individuals, either via their favourite artists or simply by pointing to pieces of music which they prefer or do not prefer. However, people seldom possess a deep understanding about the features and characteristics behind the music and thus may find it difficult to search for new music based on their personal tastes. Fortunately, in the World Wide Web there exist digital libraries which contain information composed by experts in this domain or by users with special interests. One such major public library is MusicBrainz[1] which is essentially a social web service in the mu-

[1]`http://musicbrainz.org/`

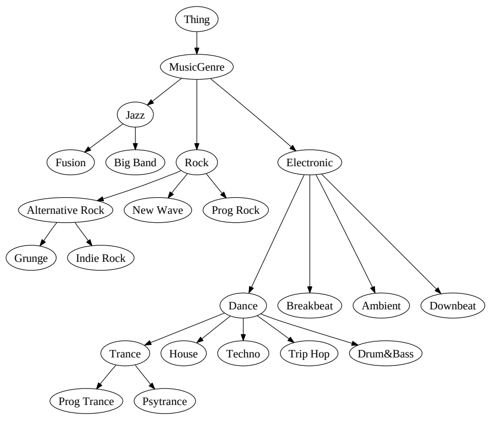


Figure 1: An Example Ontological Concept Hierarchy of Music Styles

sic domain. Through this service, users can input information in a structured manner from which it is relatively easy to infer an ontology.

For example, consider the concept hierarchy in Figure 1. This information is composed by musical experts and describes a hierarchy of musical styles. While information such as this is well understood and formalised by domain experts, we do not assume the user has extensive knowledge of the domain at all. Still this information can be readily exploited when reasoning with preferences. Note that in reality these structures tend to be far richer than this example and often describe an attribute in the domain in a comprehensive manner. In our work, we remedy this problem by eliciting complex user preferences via simple queries which the user may feel more comfortable answering. We illustrate our ideas through examples from the music domain and provide experimental results in music preference elicitation to evaluate our approach. However, it is important to note that the techniques we develop are general and can be applied to a variety of domains.

2. BACKGROUND

We wish to express preferences formally and in a way that allows us to query a database of items using these preferences. In this section we cover the necessary background to understand how we can query a database with preferences and, more specifically, how we can query an ontology-based database with preferences. In particular, we will see how we can utilise the structure of an ontology to more accurately reason with preferences so as to provide the user with a recommendation or an elicitation query.

2.1 Basic Preference Querying

In the context of database systems, [10] introduce the ability to query with preferences using an extension of SQL. The user can express their preferences as soft constraints and will receive tuples which *best match* those constraints. This approach is referred to as the BMO (*Best Match Only*) query model in which a tuple will find its way into the final result set if there does not exist any other tuple which *dominates* it, i.e. better satisfies the preference constraints. Preference constraints in this framework can be expressed through standard operators in terms of *likes*/*dislikes*(e.g. `=`, `<>`, IN) and numeric constraints (e.g. `<`, `>=`, `BETWEEN` and `AROUND`). In order to allow complex preference construction, two binary preference assembly operators are introduced, namely, the *Pareto* operator for considering two preference constructs as equally important, and the *Cascade* operator for prioritising one preference constructor over another.

The SPARQL query language is a W3C Recommendation and considered to be a vital tool for querying ontological information. [11] introduces an extended version of SPARQL (referred to as P-SPARQL) which follows the same ideas as [10] in introducing preferences as soft constraints into the language. It forms the basis of the implementation of our approach.

2.2 Utilising Ontological Structure for Querying with Preferences

In previous work [12] we extended P-SPARQL by exploiting the information in an expert supplied ontology to further refine the results of preference queries. This work solves the problem that plagues coarse preference queries, namely matches are not sufficiently distinguished. It also allows for the construction of similarity-based queries. We only present the basic ideas here and refer the interested reader to [12] for a more extensive coverage of these details.

In order to exploit the hierarchical structure of an ontology, we developed a method for computing categorical similarity between concepts in a $TBox$. We use this similarity method for performing preference querying over ontological information. We introduced a new Boolean operator $Sim(C_1, C_2)$ (is similar to) that tells whether one binding result (b_2) is preferred to another (b_1) w.r.t the user preference P:

$$b_1 \prec_{P(C_0)} b_2 \Leftrightarrow \\ Sim(C(b_1), C_0) < Sim(C(b_2), C_0) \quad (1)$$

where $C_0 \in Concepts$ is a user preference concept, $b_1, b_2 \in ResultBindings$ and $C(b_i)$ is the value bound to the relevant variable in the result binding b_i w.r.t C. In other words result binding b_2 is preferred to result binding b_1.

EXAMPLE 1. *Suppose we would like to query music albums while preferring albums of style similar to Alternative Rock music (as the highest priority) and then albums released around the year 2001. Our query has a* `PREFERRING` *section as follows:*

```
PREFERRING
  ?style ~= music:AlternativeRock
CASCADE
  ?year AROUND 2001
```

Where $\sim=$ *is the syntactic version of the above similarity operator* $Sim(C_1, C_2)$.

Note that by introducing a new Boolean operator we do not change the notion of domination querying. We still have the ability to compare two result bindings to obtain the preference domination relationship between them. There are many ways to compute the similarity between concepts in an ontology, each reflecting a different rationale. In [12] we develop a novel similarity method, based on [13], which has three interesting properties:

1. It considers two concepts more similar if they share more specific information.

2. It respects the *IS-A* relation axiom which means that a concept will always be considered more similar to any of its subconcepts than other concepts.

3. Within a sub-graph and given a preference concept, it will consider a concept more similar to this preference concept according to the communicated level of specificity described by this preference concept.

Property 3 means that when the user specifies a certain preference concept, the sub-concepts below this concept will be ordered according to their distance to this preference concept (the 'closer' the distance, the more similar they are). The intuition behind this is to respect the user's communicated level of specificity (given by the depth of this preference concept in the ontology) considering concepts which are 'closer' to this level of specificity to be more similar. This is measured by the following similarity metric which determines the similarity of concepts C_0 and C_1.

$$Sim(C_1, C_0) = \frac{2 * N_3}{N_1 + N_2 + 2 * N_3 + AVG} \tag{2}$$

Where N_1, N_2 are the distances from the concepts C_0 and C_1 to their MRCA (most recent common ancestor) respectively and N_3 is the distance from this MRCA and the root of the ontology (assuming the most general concept is the OWL concept *Thing*), AVG is the average distance of MAX to the depth of the concepts C_0 and C_1 and MAX is the length of the longest path from the root of the ontology to any of its leaf concepts. Note that in practice we then normalise the similarity measurement to be between 0 and 1 by dividing it by the similarity of the preference concept to itself. This way we ensure that the similarity between the preference concept and itself will always be 1 (which accords with intuition).

EXAMPLE 2. *In our example (Figure 1), suppose we would like to calculate the similarity values of music styles in relation to the user preference concept Trance. The concept Ambient will have the similarity value* $Sim(Trance, Ambient) = \frac{2*2}{2+1+2*2+1.5} = 0.47$ *while the concept class Techno will have the similarity value* $Sim(Trance, Techno) = \frac{2*3}{1+1+2*3+1} = 0.67$*. Therefore, Techno will be considered more similar to Trance than Ambient. However, the similarity value between Trance and Techno will be smaller than the similarity value between Trance and any of its sub-classes, e.g.* 0.84 *for the similarity between Trance and ProgressiveTrance.*

2.3 Querying with Complex Preferences

This idea can be easily extended to the case where we have elicited the user preferences in terms of a partial pre-order rather than a single concept [14]. We would still like to be able to query our database of items with these preferences. The idea is to turn this partially specified preference ordering into a total pre-order by "filling out" (or completing) the user preferences with information from the ontology while utilising the notion of similarity. The position of every concept in the total pre-order will be determined by looking at their maximal similarity to any of the user ordered concepts. Concepts which are most similar to a user ordering concept will be then ordered according to their similarity to it. The intuition behind this is that we exploit the ordering given to us by the ontology (w.r.t a similarity measurement) without contradicting the preference ordering explicitly expressed by the user. Note that this also allows expressing indifference between two concepts.

EXAMPLE 3. *The preference*

{AlternativeRock, ProgRock} `THEN` *{Electronic}*

means that the user prefers a song with style `AlternativeRock` *or* `ProgRock` *to one with style* `Electronic`*. Given a similarity method with the semantics mentioned above, and given the ontology of music styles above, the total pre-order created will be:*

{AlternativeRock,ProgRock}
{Grunge,IndieRock}
*{*other rock concepts*}*
{Electronic}
{Ambient,Breakbeat,Dance,Downbeat}
*{*other Electronic concepts*}*

AlternativeRock *and* ProgRock *perfectly match the first preference and appear at the top of the total pre-order. Concepts are then ordered according to their similarity to these preference concepts until we reach a concept more similar to the second preference: this is the concept* Electronic *(which perfectly matches the second preference). Concepts are then ordered according to their similarity to* Electronic *so as to complete the total pre-order.*

Another very essential enhancement this work provides is the ability to query the top-k elements in relation to the user preferences (in addition to querying the best match only) while preserving the qualitative nature of our reasoning. An implementation of these methods has been completed based on the *ARQ* SPARQL query engine (a *Jena* based query engine).

3. DYNAMIC PREFERENCE ELICITATION

The goal of our work is to provide users with personal recommendations that accurately reflect their preferences. In this paper, we concentrate on the *preference elicitation problem*, i.e. the problem of selecting *preference elicitation queries* to present the user in order to elicit their preferences. The resulting preferences are subsequently used to query a database of items as explained in the previous section. More accurately, we focus on the problem of selecting a series of such queries in an iterative and dynamic manner, i.e. with respect to both the user response to the queries presented to them as well as the system aim to cover certain possible preferences during this process. The attribute space of the domain, in our case, is defined via a domain ontology provided by a domain expert or social web resources which classify items in the domain according to multiple attributes. It is also important to point out that these items could be multiply classified to some attributes, i.e. have more than one value on certain attributes. This will influence the type of P-SPARQL queries we will choose in order to select and present items to the user (as opposed to more straightforward query relaxation techniques).

3.1 User Interaction

As described above, a basic assumption in our work is that users do not possess the expert knowledge which allows them to precisely and explicitly communicate their preferences in terms of the attributes in a domain. However, they may well have preferences which it is our job to elicit from (and for) them. Since we cannot query the users about those attributes directly, we will limit our preference elicitation queries to ranking individuals only and collect the information associated with them, building a preference model as we go. More specifically, the user is presented with individuals and asked to rate them as either *liked*, *disliked* or *neutral*. With this feedback (see top-left of Figure 2) the user's predicted preference order is modified by combining the statistical score and confidence interval for how much an attribute value is liked by the user to produce a partial pre-order representing their preferences. This process continues again with the elicitation method moving between *exploration* and *exploitation* phases in order to determine which item to present to the user next for their consideration and also to ensure that a sufficient cross-section of the item space is presented in order to obtain an accurate preference order.

We focus on the elicitation of user preferences by developing an elicitation technique that is itself guided by the expert supplied ontology. We will then show how this process can be made more efficient by clustering the items in the ontology. In the rest of the paper we will present this preference elicitation process and

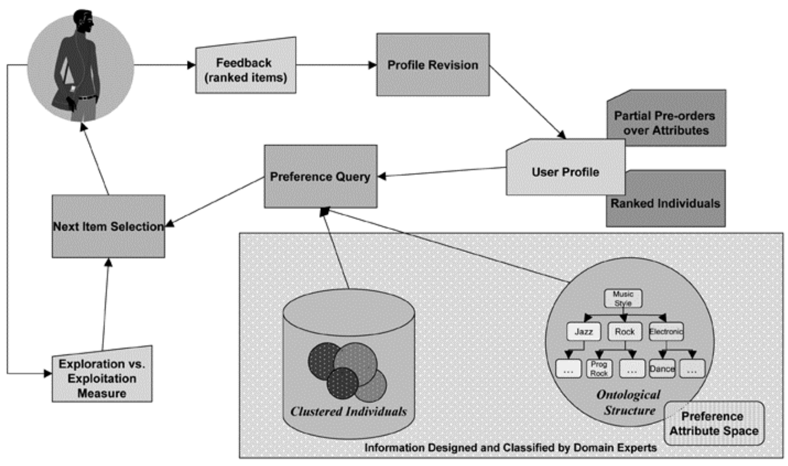


Figure 2: A High-Level Schema of the Preference Elicitation Process: users express their preferences over individuals presented to them ('Feedback' box in the figure). The system will then update their preference profile consisting of a partial pre-order over certain attributes as well as the user ranking history. The system will then consult an ontology in order to guide the next elicitation query (in terms of a new individual to be ranked – 'Next Item Selection' box). The system goal is to obtain confidence in previously elicited preferences (exploitation) as well as to cover new preferences (exploration).

its clustering-based extension. We also briefly discuss the building of a preference model on behalf of the user and its evaluation.

3.2 Ontology Guided Preference Elicitation

As described above, the goal is to elicit implicit user preferences over some predefined attribute space through the rankings of instances that the system offers to the user. Once we have enough information about the user's rankings, we can start building the user preference profile. The dynamics of this process are designed so we can confidently elicit as many preferences as possible and at the same time allow for an interesting interaction between the user and the system. The idea here is to have a trade-off between wanting to learn more about the current hypothesis regarding the user's preferences and wanting to more widely explore the user preference hypothesis space as well as keeping the user happy and engaged. In the first case, we would like to become more confident about the user preferences in a particular area of the ontology while in the second case we would like to explore more areas in order to discover more preferences. From the user perspective we can view this trade-off in terms of *exploitation* of previously elicited preferences and *exploration* of new (as yet unrevealed) user preferences. Algorithm 1 shows the main interaction loop the system follows. At any given stage, we will look at a fixed size window of the user's last interactions (via the global variable $window$) in order to determine the next *preference elicitation query type*. We define four types of preference elicitation queries, namely: *similarity queries*, *queries with controlled dissimilarity*, *exploration queries* and *exploitation queries*. *Similarity queries* will search and offer the user an item similar to those last seen; *queries with controlled dissimilarity* will return an item which is less similar to these items (as determined by the ontology); *exploration queries* will return an item which is classified to sufficiently different attribute values to items last presented while *exploitation queries* will return an unseen item which holds a high value of information (see Section 3.2.3). At any stage the system has to make a decision in regard to the next preference elicitation query (see Algorithm 2), which determines the dynamics of the system. In each of these queries we call the function $getItemRelated()$ to return items at the top, middle or bottom

Algorithm 1 Elicit Preferences

```
$elicit()$
  $window \leftarrow \{\}$
  **loop**
    $query \leftarrow nextPreferenceQuery()$
    $response \leftarrow userResponse(query)$
    $window \leftarrow window \cup \{query, response\}$
    $updatePreferenceOrdering(query, response)$
  **end loop**
```

of the ordered result set. We discuss the main ideas behind these decisions in the remainder of this section.

3.2.1 Similarity Queries

When receiving a positive response from the user, it is quite natural to form a hypothesis which entails that the information we are encountering at the current stage of elicitation is classified to attribute values which will be highly ranked in the user preference profile. Querying the user about similar information is desirable in order to either gain further confidence in this hypothesis or alternatively contradict it in which case we will conclude that these high rankings were due to noise. The way we achieve this is by executing similarity-based P-SPARQL queries which return the top-k individuals in relation to the current attribute values. Due to the nature of these selections, the resulting individuals will be classified not only to the exact same attribute values the user has ranked highly but also to values which are highly similar: the procedure $getItemsRelated(item, x, y)$ executes a P-SPARQL query and returns items x to y when ordered w.r.t the attributes $attr(1) \ldots attr(n)$ associated with a given $item$ where x and y specify an integer range of records. A simplified version of such a query is given below:

```
SELECT RECORD $x$ TO $y$
PREFERRING
  attr(1) $\sim=$ item.attr(1)
AND
```

Algorithm 2 Next Item Selection

```
$nextPreferenceQuery()$
  $possibleQItems \leftarrow \{\}$

  $queryType \leftarrow establishQueryType()$

  **if** $queryType = Similar$ **then**
    $possibleQItems \leftarrow getItemsRelated(lastQItem, 1, k)$
  **else if** $queryType = ControlledDissimilarity$ **then**
    $possibleQItems \leftarrow getItemsRelated(lastQItem, k, l)$
  **else if** $queryType = Explore$ **then**
    $possibleQItems \leftarrow getItemsRelated(lastQItem, l, m)$
  **else if** $queryType = ExploitPreferences$ **then**
    $possibleQItems \leftarrow getItemsWithHVI()$
  **end if**
  **return** $rand(possibleQItems)$
```

Algorithm 3 Establish Preference Query Type

```
$establishQueryType()$
 1: **if** $|window| < winSize$ or $negResponse(window) \leq negThr$ **then**
 2:   $queryType \leftarrow Similar$
 3: **else if** $globalNegResponse() \leq globalNegThr$ **then**
 4:   **if** $previousQueryType = Similar$ **then**
 5:     $queryType \leftarrow ControlledDissimilarity$
 6:   **else**
 7:     $queryType \leftarrow Explore$
 8:   **end if**
 9:   $window \leftarrow \{\}$
10: **else**
11:   $queryType \leftarrow ExploitPreferences$
12:   $window \leftarrow \{\}$
13: **end if**
14: **return** $queryType$
```

```
  attr(2) ~= item.attr(2)
...
AND
  attr(n) ~= item.attr(n)
```

We keep gathering preferential information about these values until a certain threshold is met. This threshold will also be influenced by the number of negative responses we receive from the user where in case this number is high, the number of iterations we will spend on the current values (e.g. current branch in the ontology) will be reduced. Algorithm 3 shows the management of these parameters which will establish the type of preference elicitation query we will use next. We adopt this type of query when the window of user interactions has not yet reached its maximal size and the response we get from the user is not negative enough (line 1).

EXAMPLE 4. *Suppose the user has ranked an item classified to music styles* House *and* Techno *and with the release year 2002. The P-SPARQL query with respect to this item will then be:*

```
SELECT RECORD 1 TO k
PREFERRING
  ?style ~= :House
AND
  ?style ~= :Techno
AND
  ?releaseYear ~= 2002
```

Algorithm 4 Select Items with High Value of Information

```
$getItemsWithHVI()$
  $partialPreorder \leftarrow collectPreferences()$
  $totalPreorder \leftarrow computePreferences(partialPreorder)$
  $value \leftarrow null$
  $i = 0$
  **while** $value = null$ **do**
    **if** $totalPreorder[i] \notin partialPreorder$ **then**
      $value \leftarrow totalPreorder[i]$
    **end if**
    $i{+}{+}$
  **end while**
  **return** $getItems(value)$
```

We will then randomly select an item from this result set and present it to the user as the next elicitation query. We can assume that this elicitation query item will have similar attribute values to the previous one. For example, music styles House *and* Trip Hop *and with the release year 2003.*

3.2.2 Querying Diverse Items

Let us consider now the case where we have elicited a sufficient amount of information about a certain branch in the ontology. We would like to change our preference elicitation queries so that we can elicit user preferences about other parts of the ontology. If the reaction of the user was sufficiently positive in the previous phase, in order to make the transition between items smoother and offer a better user experience we select items within a limited distance from the previous items. In this case $establishQueryType$ will return query type $ControlledDissimilarity$ which will modify the call to $getItemsRelated$ by selecting elements from the middle of the result set (controlled by the parameters k and l predetermined as a function of the size of the item-space).

EXAMPLE 5. *Consider the item ranked in Example 4 and suppose we have now reached the stage where we would like to query items with certain diversity to this item. We will execute the same P-SPARQL query as before with the parameters k and l. We can assume that the next elicitation query item will have attribute values which appear within a certain limited distance to the previous one. For example, music styles* Dance *and* Ambient *and the release year 2000.*

However, if the user's response to the previous phase was not positive, then eliciting preferences from that particular area in the ontology may no longer be desirable and we can try moving further afield. Since we are dealing with hierarchical structures, it is relatively easy to control the selection of preference elicitation queries and increase the level of dissimilarity our P-SPARQL query returns. In that case we will execute an *exploration query* which will select items from the bottom of the similarity query's result set and will return an item with greater distance than the last window. Note that once we have selected an item via these query types, we will go back to querying with similarity (which means we will reset the window) until the threshold criteria will entail querying with certain diversity again.

3.2.3 High Value of Information

In many domains, users may have preferences over more than one type of individual. In ontological terminologies, they may hold preferences over different branches in the hierarchy. Therefore, once we have gathered enough confidence in the preference elicitation for a particular part of the hierarchy, it is important to be able to explore different areas as well. In our work, preferential information with a *high value of information* are those preferences the user holds which we have not yet revealed. In order to discover these preferences we need to be able to explore individuals classified to attribute values the system is uncertain about. Since we are dealing with hierarchical structures, here again we can control the selection of preference elicitation queries. We will choose to select new attribute values about which we are uncertain of the user preferences and which are highly similar to known preferences. The way we achieve this is by computing the preferences we have obtained from the user up to this point as a partial pre-order and then searching for an attribute value which does not appear in this pre-order but is similar to an attribute value which is highly preferred in the order. The obtaining of a partial pre-order over a certain attribute from ranked individuals is briefly discussed in Section 3.4. Once we obtain the partial pre-order, we expand it to a total pre-order (as described in Section 2.3) and select an attribute value which appears highly ranked in the total pre-order but does not appear in the partial pre-order. Since this value did not appear in the partial pre-order we can deduce that the user has not yet ranked sufficiently enough items of this type. And since it appears highly ranked in the ordering, it satisfies our High Value of Information criterion. Algorithm 4 shows a simplified version of these ideas where the $getItems(value)$ procedure is assumed to execute a P-SPARQL query selecting items with a certain attribute value which equals a given $value$. In terms of user interaction, we will use this query type in order to re-start the elicitation process when repeatedly receiving negative feedback from the user (line 3 in Algorithm 3). Note that this preference query type is also used at the beginning of every user session.

EXAMPLE 6. *Suppose we have calculated the user preferences as a partial pre-order in terms of music style and the resulting ordering is:*

{AlternativeRock, ProgRock} THEN *{Electronic}*

Calculating the total pre-order, the music style IndieRock *will appear in a high position in the total pre-order and will be used to select the next preference query item (see Figure 1).*

3.2.4 Complexity and Performance Issues

The main difficulty when it comes to database querying with preferences is the complexity of queries. Even though the basic ideas behind the Pareto and Cascade operators are simple, the nature of comparison queries makes them quadratic in the number of items. Since we are dealing with large item spaces and since the execution of such queries could be very frequent, it is crucially important to make these queries more efficient. Furthermore, in our case, when we wish to explore diverse items w.r.t a ranked item, it is unnecessarily expensive to compare all particular individuals to each other in order to get this desired effect. In order to avoid that, it would be better to focus our attention on part of the item-space. We do so by grouping our individuals into some high-level clusters and reason over these clusters before we dive into the actual selection of particular preference query items. We discuss this enhancement in the next section.

3.3 Making Preference Elicitation Faster with Clustered Ontologies

There are many ways to cluster categorical and numerical information. In our work, we make use of a particular structure, namely *explicit semantic relations*, which is knowledge described in the ontology by way of direct semantic associations between individuals. This is usually done via *roles* which describe a direct semantic relation between individuals of the same type. For example, $similarTo : Artist \mapsto Artist$ to describe that one artist is similar to another. These structures can be viewed as weighted or unweighted graphs in which elements of the item-space are nodes of the graph and the semantic relation determines the edges (e.g. $similarTo$). We find that in many domains there exist classifications which form this kind of structure and induce a semantic network and in some cases a similarity network. In our example, there exists a similarity network between artists, available in music libraries etc., which can be viewed as a graph (in our case, an unweighted graph) and allows us to generate a high level hierarchical clustering structure.

3.3.1 Clustering Semantic Relatedness

The main idea behind clustering similarity graphs is to look for highly connected sub-graphs. The way this is done is through methods which are based on *minimum cut trees* within the graph. Given a graph $G(V, E)$, a cut is a set of edges whose removal will disconnect the graph. A minimum cut is a cut with a minimal number of edges. [15] presents a clustering algorithm which computes minimal cuts iteratively and looks for sub-graphs with a high level of connectivity. At the end of this process, the graph will be partitioned into j clusters where j is the number of clusters and is unknown at the beginning of the process. In our work we adopt this technique since it has a simple generalisation into a hierarchical clustering method. This is a very desirable effect since we are dealing with ontologies which treat hierarchical structures straightforwardly. The resulting cluster structure gives a high-level classification of individuals with some similar characteristics.

3.3.2 Querying the Clustered Item Space

The basic idea behind our enhanced preference elicitation technique is similar to what we have seen before. The main difference, now that we have our items clustered, is that on every elicitation query, we will first determine which cluster we would like to select from and then execute the P-SPARQL query limiting the search to that cluster (and thus limiting the complexity to the size of the cluster). Algorithm 5 (which now replaces Algorithm 2) shows the revised preference query item selection procedure with clustering. The selection of the particular cluster will depend on the preference query type and the cluster of the item previously ranked according to the same principles we discussed in 3.2.

3.4 Building the Preference Model

During the preference elicitation process, users encounter a variety of different items whose attributes are described in terms of concept classes. These concept classes are used to represent their preferences. We now build a partial pre-order preference relation over those classes on behalf of the user. The way we approach this is through standard statistical reasoning where we look at the mean score of individuals classified to each class taking into account the confidence of this score. For each concept class, we compute its score and confidence. The score of a class is measured in terms of the probability that if we draw an individual classified to this class, it will be ranked positively, negatively or neutrally. A requirement for a class to be included in the partial pre-order is that its confidence measure is at most equal to some predetermined constant which determines how sparsely ranked a concept might be to be included in the partial pre-order. We will order the classes in the partial pre-order according to their score and confidence where two classes will be ordered at the same level if they do not have a significant statistical difference. This is computed in relation to their overlapping confidence interval ratio. This partial pre-order pref-

Algorithm 5 Next Item Selection (revised)

```
$nextPreferenceQuery()$
  $possibleQItems \leftarrow \{\}$

  $queryType \leftarrow establishQueryType()$

  **if** $queryType = Similar$ **then**
    $c \leftarrow lastQItem.cluster$
    $possibleQItems \leftarrow getItemsRelated(lastQItem, c)$
  **else if** $queryType = ControlledDissimilarity$ **then**
    $c \leftarrow getSimilarCluster(lastQItem.cluster)$
    $possibleQItems \leftarrow getItemsRelated(lastQItem, c)$
  **else if** $queryType = Explore$ **then**
    $c \leftarrow getDiverseCluster(lastQItem.cluster)$
    $possibleQItems \leftarrow getItemsRelated(lastQItem, c)$
  **else if** $queryType = ExploitPreferences$ **then**
    $c \leftarrow getClusterWithHVI(lastQItem.cluster)$
    $possibleQItems \leftarrow getItemsWithHVI(preference, c)$
  **end if**
  **return** $rand(possibleQItems)$
```

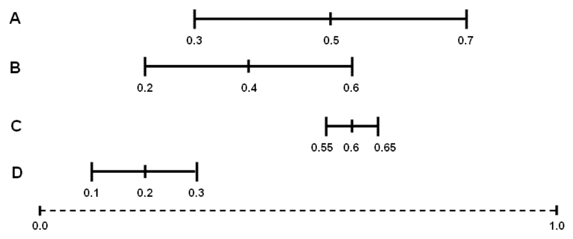


Figure 3: Overlapping Confidence Intervals Example

erence model will be then expanded to a total pre-ordering using (Section 2.3). Due to lack of space, we do not present the mathematical details of this method. However, this can be easily derived using standard statistical inference.

EXAMPLE 7. *Consider the scoring statistics of four classes as gathered by the system according to user responses in Figure 3 given in terms of confidence intervals* $[0, 1]$. *Computing the overlapping confidence interval ratio of classes* A *and* B, *we can infer that* A *and* B *should be considered at the same preference level. On the other hand, class* C *will be strictly preferred over class* A. *Similarly, we can compute the overlapping confidence interval ratio of classes* B *and* C. *The resulting partial preference pre-order in this example will be* $C \succeq \{A, B\} \succeq D$.

4. EXPERIMENTAL RESULTS

We evaluate our preference elicitation methods through a series of experiments in the domain of music. We have implemented a music preference elicitation system we call *The Music Preference Explorer* which selects and suggests music to users according to the preference elicitation methods described above. We ran an experiment with the system and several users.

4.1 Data Model

The data model we use is an ontology we collected from information available on the web. The ontology consists of 134 artists and around 1200 tracks from 338 albums of popular rock, pop and electronic music. We created a hierarchy of 137 styles which is arranged as a DAG and classifies each album to at least one style (and around four on average). We also collected similarity information between artists and clustered our data set into 26 different clusters arranged hierarchically according to the method described in 3.3. The cornerstone for this dataset is the *MusicBrainz* library. However, since this is a new service, some classifications needed to be collected manually from other resources on the web.

4.2 User Trials

When a user logs into the system, they are presented with a piece of popular music and can specify their ranking as one of the following options: they can either say they like the music they are listening to, they do not like it or that they are not sure about their preferences. The system then collects those rankings and gradually builds a preference model on behalf of the user in terms of the music style classified. The system executes the preference elicitation cycle we described in 3.2. If the user responds positively to certain types of music, the system will suggest music which is similar according to its model. However, it will gradually select music with a different style and at some point explore styles which are quite different (and notify the user that it has done so). If a user gives a negative response to a certain type of music, this process will occur faster. We ran the trial on 22 users and present the results in the following section.

4.3 Results

We measure the performance of our methods in terms of the prediction accuracy of the elicited preference model as well as the coverage of different preference levels.

4.4 Preference Model Accuracy

Our first test measures the accuracy of the elicited model by trying to predict the user response on random tracks according to the created preference model. Figure 4 shows the accuracy of correctly predicting whether a user will say they like a piece of music and when they say they will dislike it. We compute the mean rank of a track (according to its classified information) and compute a threshold (depending on the number of preference levels the user holds) to differentiate between predicting whether the track will be liked or otherwise. In order to measure the growth in accuracy, we recreated the user ranking cycles by first building their preference model using the first 20 tracks (in order) and gradually adding tracks until we used all the tracks they had ranked. We have clearly shown a growth in prediction accuracy and reached over 80 percent accuracy on average from around 160 examples. Note that these results were achieved by basically looking at a single (probably very central) attribute and performing well on average.

4.5 Preference Coverage

Finally, since we are dealing with a dynamic process where our methods are designed to cover different user preferences over time by allowing the user to explore new preferences, we measure preference coverage in terms of preference ordering and clusters covered. Figure 5 shows the growth in the number of different preference ordering levels we elicit over time. This includes the partial pre-order computed through the statistical analysis described in 3.4 as well as the number of total pre-orders realised after augmenting this pre-order with further information from the ontology. This gives some idea about the dynamics of the system and shows the growth in elicited information over time. In addition to the number of preference levels, we also measure the number of clusters the user has visited since this is a very central part of the elicitation cycle. Note that when reaching over 180 ranked items, the number of preference levels starts to drop. This is due to the overlapping confidence interval matching we described in Section 3.4: some ordering levels are merged when gathering more statistical confidence. Note also that when reaching this number of rankings, our model prediction accuracy reaches around 85% (Figure 4). This shows that the preference ordering matching was indeed justified and better reflected the user's true preference.

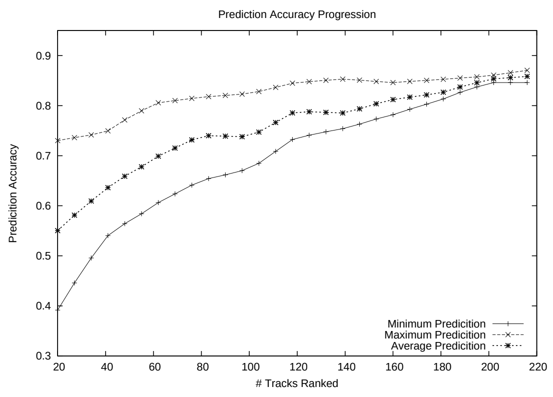


Figure 4: Prediction accuracy progression: shows the accuracy of predicting whether the user will like/dislike a track according to the elicited preference ordering

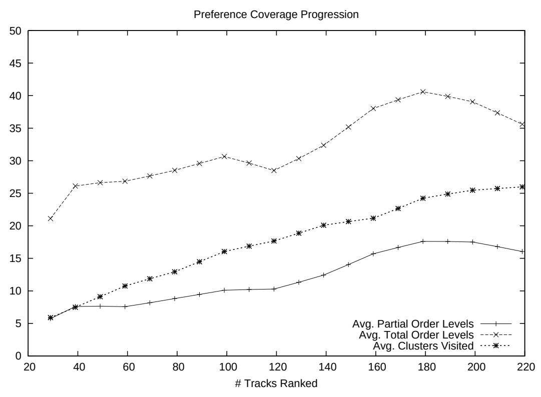


Figure 5: Preference coverage progression: shows the progression of preference ordering levels (partial and total) as well as the number of clusters visited. Note that when reaching over 180 ranked items, the number of preference levels starts to drop due to overlapping confidence interval matching as described above, as the confidence grows

5. CONCLUSIONS

In this paper we have developed a preference elicitation method which caters for a large class of problems in preference modelling. In particular, we would like to elicit a preference ordering that can be used as the basis of a formal preference query to a database of items. However, on the one hand user's are not adept at making explicit their preferences and on the other, even those preferences that could be made explicit, may be difficult for the non-expert user to specify in a formal query language. The technique we developed here provides a formal solution to this problem while at the same time hiding the technical detail from the user. In our method, we rely on a complex model of the domain at hand; namely a domain ontology created by experts. We presented an interactive elicitation process which is dynamic and is able to elicit many preferences by covering large portions of the ontology. During this process, users are presented with examples which they are asked to rank and a system collects this information and builds a preference model on their behalf. The dynamics of the process relies mainly on the interaction between exploiting previously elicited preferences thus suggesting similar examples and exploring new preferences. We evaluated our methods through experiments run with several users in the domain of music and show significant results in terms of the preference model prediction accuracy as well as the coverage of different preferences elicited during this process. For future work, we intend to investigate the elicitation of more complex preference models, i.e., dependency structures over multiple attributes as well as a more advanced use of ontologies and semantic web features. Another avenue for future work would be in the area of collaborative filtering. We can aggregate the preference profiles of similar users to provide a more accurate 'collaborative' preference.

6. REFERENCES

[1] Anand, S.S., Kearney, P., Shapcott, M.: Generating semantically enriched user profiles for web personalization. ACM Trans. Inter. Tech. **7**(4) (October 2007)

[2] Rashid, A., Albert, I., Cosley, D., Lam, S., Mcnee, S., Konstan, J., Riedl, J.: Getting to know you: learning new user preferences in recommender systems. In: IUI, New York, NY, USA (2002) 127–134

[3] Bradley, K., Rafter, R., Smyth, B.: Case-based user profiling for content personalisation. LNCS **1892** (2000) 62–72

[4] Boutilier, C., Brafman, R.I., Hoos, H.H., Poole, D.: CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. JAIR **21** (2003)

[5] Boutilier, C.: A POMDP formulation of preference elicitation problems. In: AAAI. (2002) 239–246

[6] U. Chajewska, D.K., Parr, R.: Making rational decisions using adaptive utility elicitation. In: AAAI. (2000) 363–369

[7] Middleton, S.E., Shadbolt, N.R., De Roure, D.C.: Ontological user profiling in recommender systems. ACM Trans. Inf. Syst. **22**(1) (2004) 54–88

[8] Schickel-Zuber, V., Faltings, B.: Inferring User's Preferences using Ontologies. In: AAAI 2006. (2006) 1413–1418

[9] Ziegler, C.N., Lausen, G., Schmidt-Thieme, L.: Taxonomy-driven computation of product recommendations. In: CIKM '04. (2004) 406–415

[10] Kießling, W.: Foundations of preferences in database systems. In: VLDB. (2002) 311–322

[11] Siberski, W., Pan, J.Z., Thaden, U.: Querying the semantic web with preferences. In: ISWC. (2006) 612–624

[12] Chamiel, G., Pagnucco, M.: Exploiting ontological information for reasoning with preferences. In: Multidisciplinary Workshop on Advances in Preference Handling. (2008)

[13] Wu, Z., Palmer, M.: Verb semantics and lexical selection. In: ACL. (1994) 133–138

[14] Chamiel, G., Pagnucco, M.: Exploiting ontological structure for complex preference assembly. In: Australian Joint Conference on Artificial Intelligence. (2008)

[15] Flake, G.W., Tarjan, R.E., Tsioutsiouliklis, K.: Graph clustering and minimum cut trees. Internet Mathematics **1**(4) (2004) 385–408

Augmenting Collaborative Recommender by Fusing Explicit Social Relationships

Quan Yuan, Shiwan Zhao
IBM China Research Laboratory
Beijing, 100193, China
{quanyuan, zhaosw}@cn.ibm.com

Li Chen
Department of Computer Science, Hong Kong Baptist University
Hong Kong
lichen@comp.hkbu.edu.hk

Yan Liu
IBM T.J. Watson Research Center
Yorktown, NY 10598
liuya@us.ibm.com

Shengchao Ding
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, 100190, China
dingshengchao@ict.ac.cn

Xiatian Zhang
IBM China Research Laboratory
Beijing, 100193, China
xiatianz@cn.ibm.com

Wentao Zheng
IBM China Research Laboratory
Beijing, 100193, China
zhengwt@cn.ibm.com

ABSTRACT

Nowadays social websites have become a major trend in the Web 2.0 environment, enabling abundant social data available. In this paper, we explore the role of two types of social relationships: membership and friendship, while being fused with traditional CF (Collaborative Filtering) recommender methods in order to more accurately predict users' interests and produce recommendations to them. Through an exploratory evaluation with real-life dataset from Last.fm, we have revealed respective effects of the two explicit relationships and furthermore their combinative impacts. In addition, the fusion is conducted via random walk graph model in comparison with via weighted neighborhood similarity matrix, so as to identify the best performance platform. In-depth analysis on the experimental data particularly shows the significant improvement by up to 8% on recommendation accuracy, by embedding social relationships in CF via graph model.

Categories and Subject Descriptors

H.5.3 [**Group and Organization Interfaces**]: [Collaborative Filtering, Computer-supported cooperative work, Evaluation/methodology]; H.3.3 [**Information Storage and Retrieval**]: [Information Filtering]

General Terms

Algorithms, Experimentation, Human Factors

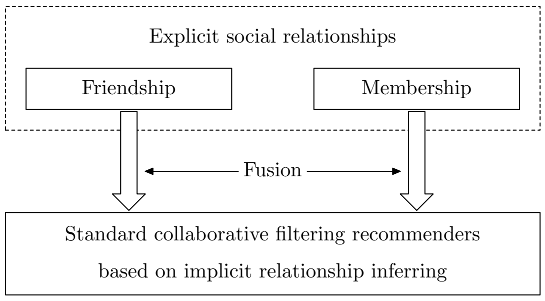
Keywords

Recommender Systems, Collaborative Filtering, Social Relationship, Random Walk

1. INTRODUCTION

In recent years, collaborative-filtering (CF) based recommender systems have been widely developed in order to effectively support users' decision-making process especially when they are confronted with overwhelming information (e.g. a large amount of product options that popularly appear in the current Web environment). There are two basic entities considered by the recommender: the user and the item. The user provides his rates on items (e.g. movies, music, books, etc.) that he has experienced, based on which the system can connect him with persons who have similar interests and then recommend to him items that are preferred by these like-minded neighbors. In some cases which don't have rating values available, the user's interaction with items can also be considered. That is, if we can only get the information that a user watched a movie, the "rating" is 1; otherwise it is 0. The recommendation method based on this binary rating matrix is also named as log-based CF [24]. The traditional approaches can be hence regarded as implicit ways to infer the social relationship between users. However, it inevitably brings the limitation when few users rated or viewed few items (i.e. the sparsity problem), to make it hard to infer such preference relationship. With the increasing emergence of social network services, many websites support online user communities, such as *Youtube*, *Last.fm*, *del.icio.us*, and e-commerce sites including *Amazon.com* and *eBay.com.* Community facilities are provided so that users can create and access to their community information and communicate with friends or members. For example, on Last.fm (a popular music recommender website), the user can establish friendship with others by "finding people" and/or join a group of users having similar music tastes (e.g. through "finding groups").

We term this kind of community relationship as explicit social relationship, since it is directly defined by users, rather than inferred by the system. As a matter of fact, two types



Explicit social relationships
Friendship
Membership
Fusion
Standard collaborative filtering recommenders based on implicit relationship inferring

Figure 1: Fusion of Friendship and Membership into Standard CF

of explicit relationships are commonly available on the current websites (as in the example of Last.fm): *friendship* and *membership*. For instance, users A and B are friends given that A adds B in his friend list (or conversely), but it does not refer that they must have similar music tastes. On the other hand, if A and B join in the same group, it indicates that they have the membership relation and are likely with the similar interest (e.g. on "Beatles" provided it is the group's title).

It is believed that explicit social relationship can be likely applied to compensate the limitation of implicit relationship inference approach and improve the accuracy of recommender systems [1]. Some researchers have recently been engaged in fusing such kind of information into traditional CF methods [1, 15, 19]. However, to our knowledge, most works purely concentrate on friendship data [4, 16]. Their tentative studies unfortunately show that the fusion sometimes does not work well due to the friendship's inherent ambiguity as a relational descriptor [5]. Moreover, existing fusion approaches have been mainly based on the rating-matrix. It is in essence lack of exploration of other possible ways that may be potentially more effectively able to fuse the explicit social relationship with CF recommenders. As the social data is inherently in a graph structure, fusion via graph may be a good approach.

Given that the membership contains more information implying the user's preferences, such as his interest on the music genre or singer as indicated by the group's property and his like-minded people involved in the same group, we are interested in understanding whether this additional information could produce any practical benefits. Thus, in this paper, our objective is to study whether and how to best fuse both of membership and friendship, by means of comparing their respective effects and potential combinative impacts via different fusion platforms. We believe that our study will shed light on the role and applicability of social information in boosting collaborative intelligence of current recommender systems. More specifically, our contributions can be summarized as follows:

- We demonstrate the exact value of fusing explicit social relationships into recommender systems. Particularly, in some cases, the neighborhood of users learnt from membership has been found more accurate than from purely embracing friendship with traditional CF.
- We develop a framework to fuse and evaluate multiple types of social relationships in a systematical approach through weighted-similarity calculation. Moreover, we propose a novel graph model to fuse the social relationship with rating matrix, and adopt random walk algorithm to produce neighborhood similarities for recommendations. With a real-life dataset, we compared it with the weighted-similarity approach and identify the superior performance of graph fusion in improving recommendation accuracy.

In the next section, we first provide a brief review of related work, and then propose a systematical approach to study the use of explicit social relations and incorporate them in collaborative recommenders via graph model in addition to via weighted-similarity. Next, we show the experiment design and results analysis, followed by the final section of conclusion and future work.

2. RELATED WORK

We review the related literature of fusing heterogeneous data sources with rating matrix to improve standard CF from two perspectives: *Fusion of Social Relationship*, and *Graph-based Recommender Algorithm.*

2.1 Fusion of Social Relationship

Since popular user-based and item-based CF algorithms that only rely on user-item rating matrix always suffer from sparse and imbalance of rating data, researchers have started to incorporate other data sources to improve standard CF. Balabanovic et al. [6] were among those who first investigate *content-based* systems that make use of the descriptive data about an item. Melville et al. [7] enhanced CF by using content of a movie, e.g., movie genre. Pazzani [8] investigated hybrid methods using both of user data (demographic information) and item data (content) for improving recommendation accuracy.

Recently, with the increasing development of social websites and appearance of social data, researchers have begun to pay attention to the social data and explored its usage in recommender systems. Konstas [21] adopted Random Walk with Restart to model the friendship and social annotation (tagging) in a music track recommendation system. Golbeck [11] used trust relationship in social network to improve movie recommendations. [15] used social network data for neighborhood generation. In a Munich-based German community, friends are compared to neighbors of collaborative filtering for rating prediction. Their results showed that the social friendship can benefit the traditional recommender system. [19] proposed an online social recommender system attempting to use more social information for recommendation generation. The social data they introduced are the friendship of users (from GeekBuddy), which was used to refine the description of each user. [10] proposed a factor analysis approach based on probabilistic matrix factorization to solve the data sparsity and poor prediction accuracy problems, by employing both of users' social network information and rating records. This work also concentrated on using friendship to improve recommendations. However, it has been shown that online friendship sometimes does not work well due to its inherent ambiguity as a relational descriptor [4, 16]. Compared to online friendship, online community membership contains more information about users' preferences. [2] used membership for recommending online communities to members of the Orkut social network. However, their recommendations were on a per-community, rather than on a per-user basis. I. Guy et al. [22] [23] built a people recommendation system named SONAR by leveraging data from multiple channels including membership in project wiki.

2.2 Graph-based Recommender Algorithm

The computation of user/item similarity plays a key role in user/item-based collaborative recommenders. Popular measurements of user similarity are Cosine similarity and Pearson's correlation coefficient (see [9] for examples) based on the user-item rating matrix. The limitation is that they only use the local pairwise user information for neighborhood searching.

Recent years, graph-based methods have been introduced to model relations between users and items from a global perspective, and been used to seamlessly incorporate heterogeneous data sources into the traditional user-item rating matrix. Huang proposed a two-level graph model for products [18], in which the two layers of nodes represent products and customers respectively, and three types of links between nodes are: the product-product, the user-user, and the user-product link. The recommendation is generated based on the association strengths between a customer and products.

Random walks on graph have been extensively discussed [12] and shown a rather good performance in the recommendation area. M. Gori and A. Pucci proposed a random-walk based scoring algorithm, ItemRank [14], which can be used to rank products according to expected user preferences, so as to recommend top ranked items to potentially interested users. Similarly, Baluja et al. [3] made video recommendations for *YouTube* through random walk on the view graph, which is a bipartite graph containing users and videos where links are visiting logs of users on videos. F. Fouss et al. [13] presented a new perspective on characterizing the similarity between elements of a database or, more generally, nodes of a weighted and undirected graph. This similarity called L^+, the pseudoinverse of the Laplacian matrix of the graph. Their experimental results on the MovieLens database showed that the Laplacian-based similarity computation performed well in comparison with other methods.

However, the limitation of related work in the "fusion of social relationship" (the first subsection) is that few have considered the potential positive role of membership. Moreover, in the related work on "graph-based recommender algorithm", no work on its performance as a fusion platform has been conducted. In this paper, we therefore aim at investigating how to effectively fuse both of friendship and membership into CF algorithm via random walk approach in order to improve the performance and solve the sparsity problem. To the best of our knowledge, our work is one of the first attempts to use both friendship and membership to enhance recommender systems.

3. FUSING SOCIAL RELATIONSHIPS INTO RECOMMENDERS

In this section, we mainly take into account of two types of explicit social relationships: friendship and membership, and propose a generic framework to fuse them with the user-item rating matrix. Given that user-user similarity computation is crucial to collaborative recommenders, a more accurate user-user similarity always leads to better recommendation results. Our fusion framework aims at leveraging the two social relationships to strengthen the user similarity calculation process by two means: one is combining the user-similarity from friendship and/or membership with similarity from rating matrix in a weighted approach; the other is modeling the social relations and rating matrix all in a graph, and then applying random walk on this graph to compute the user similarity.

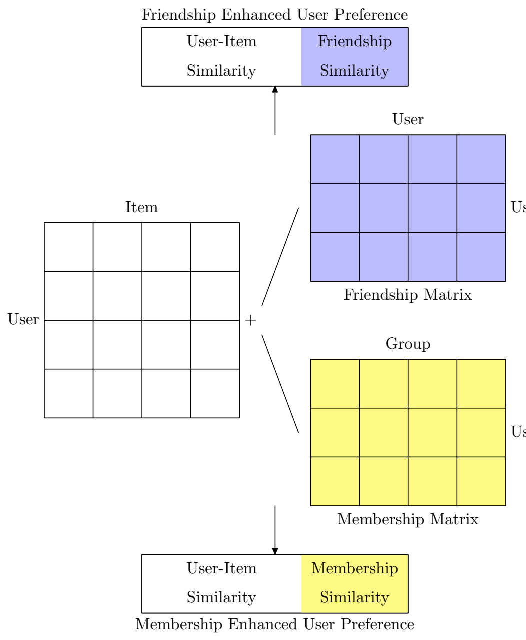


Figure 2: Fuse Friendship and Membership into Collaborative Recommender via Weighted-Similarity

3.1 Fusing via weighted-similarity

In the rating matrix, we can view the preferences of users as feature vectors. Every user vector consists of n feature slots, one for each available item. The values used to fill those slots can be either the rating r_{ak} that a user u_a provides to the corresponding item i_k, or 0 if no such rating exists. Now, we can compute the proximity between two users u_a and u_b, by calculating the similarity between their vectors. For example, we can use the Cosine similarity for this calculation as follows:

$$Similarity(u_a, u_b) = \frac{R_{u_a} \cdot R_{u_b}}{||R_{u_a}||||R_{u_b}||} \quad (1)$$

where R_{u_a} and R_{u_b} are two vectors of ratings from users u_a and u_b respectively.

According to the user similarity calculated from the rating matrix, we give a detailed description of fusing social relationships via weighted user similarity based on Fig 2.

When fusing friendship with user similarity from rating matrix, we need to get a user similarity based on the friendship firstly. We represent the friendship in the form of a user-user matrix. If two users u_i and u_j are friends, then the value of cell u_{ij} is set to 1, otherwise 0. Based on this user-user matrix, we calculate a friendship similarity by adopting Cosine Correlation, named Sim_{fri}. Next, when calculating the final user similarity between u_a and u_b, we combine the Sim_{fri} with Sim_{ui}(user similarity calculated from user-item matrix) in a weighted approach as follows:

$$Sim_{ui+fri}(u_a, u_b) = \lambda Sim_{ui}(u_a, u_b) \\ +(1-\lambda)Sim_{fri}(u_a, u_b) \quad (2)$$

The parameter λ is used to adjust the weight of Sim_{fri} and Sim_{ui}, the bigger the λ is, the rating matrix plays a more important role in the combined similarity. Finally, we use this combined similarity Sim_{ui+fri} in finding neighbors for each user.

When fusing the membership, firstly we also need to get a user-user similarity based on the membership data. Since membership is the relationship between the user and the communities/groups he/she joined, a new type of entity was introduced besides the two entities (user and item). We concretely represent the membership in the form of a user-group matrix, where the rows indicate users and the columns indicate the groups joined by users. If a user u_i joins group g_j, the value of cell u_{ij} is set to 1, otherwise 0. Based on this user-group matrix, we can get a membership similarity by using Cosine Correlation too, named Sim_{mem}. As for the generation of final user-user similarity, a weighted formula is applied where λ plays the same role as before.

$$Sim_{ui+mem}(u_a, u_b) = \lambda Sim_{ui}(u_a, u_b) \\ +(1-\lambda)Sim_{mem}(u_a, u_b) \quad (3)$$

Furthermore, we are interested in seeing what will happen if two types of social relations are fused together with the rating matrix. In this condition, we first calculate the user-user similarity Sim_{fri} from friendship and Sim_{mem} from membership independently, and then introduce two parameters: λ and β to adjust the weights of three data sources as shown in the equation 4.

$$Sim_{ui+fri+mem}(u_a, u_b) = \lambda Sim_{ui}(u_a, u_b) + (1-\lambda) \\ (\beta Sim_{mem}(u_a, u_b) + (1-\beta)Sim_{fri}(u_a, u_b)) \quad (4)$$

At the first level, λ is used to adjust the weight between rating matrix and the other two social relationships; and then β is used to adjust the remaining weight between friendship and membership. The bigger the λ is, the rating matrix plays a more important role; the bigger the β is, the membership plays a more dominant role in the combined user-user similarity.

After the computation of user-user similarity for finding neighbors, the next step is to recommend items to users by predicting each item's ratings. The predicted rating $r_{i,m}$ of a test item m for the user i is hence computed as:

$$r_{i,m} = \frac{\sum_{j=1}^{N} sim(u_i, u_j) * r_{j,m}}{\sum_{j=1}^{N} sim(u_i, u_j)} \quad (5)$$

where $r_{j,m}$ is the rating of user u_j on the item i_m, and $sim(u_i, u_j)$ is the similarity between the current user u_i and the neighbor u_j. In fusing via weighted-similarity approach, $sim(u_i, u_j)$ can be similarity measures in equation 2, 3, and 4, depends on the fusing strategy and data used each time.

3.2 Fusing via Graph

In the “fusing via weighted-similarity” method, we calculated the user-user similarity based on the static “local” pairwise user information. As we know, social network is inherently in a graph structure with the transitivity characteristics as a key feature of social relations, therefore we have been motivated to further use graph-based random walk algorithm for modeling these social data (i.e. friendship and membership). We think that the transitivity of the graph will improve the computation of the similarity between two users.

3.2.1 Graph Construction for Social Community

The first key issue is to construct a meaningful graph so that the resulting similarity can truly reflects the preference similarity between users. Considering a graph $G = (V, E, W)$, where V is the set of nodes (users, items or communities, etc.), E is the set of edges which represent relationships between all types of nodes, and W is the set of weights for all edges.

The relationship data in a social community can be interactive relationship, such that user watched a movie or listened to a song; or be social relationship like membership or friendship. Let us use Last.fm which contains all types of these data for example. It can be modeled by a graph G in the following way: there are three types of nodes in Last.fm, the user, artist (item) and group, and each element of the user, artist and group corresponds to a node of the graph; and the interactive relationship like a user *listens_to* an artist, social relationship like user is a *member_of* a group, and user is a *friend_of* another user is expressed as an edge. When a random walker walks on the graph, the difference of node types were ignored, and we only care about how many short paths existed between two nodes which have directly impact on their similarity computation, so special treatment is not needed for any type of nodes in our graph.

The weight w_{ij} of the edge connecting node i and node j should have a meaningful value. Traditionally, we usually deal with interactive data in the following convention: the more frequent the interaction between node i and node j, the larger the value of w_{ij}, and consequently, the easier the communication through the edge. However, in the case of social community, besides interactive data we also have social relationship, and they usually are not associated with a frequent number, e.g. most users join a group only once, and so is the same for adding friends. For the consistency of the two types of edges and simplicity, we set the weight of *member_of* edge and *friend_of* edge to be 1, and treat *listens_to* edge as follows: if a user listened to the songs of an artist more than 3 times, then the edge connecting the user and the artist was weighted 1. We require the weights to be both positive $w_{ij} > 0$ and symmetric $w_{ij} = w_{ji}$, so the graph we built is an undirected graph.

When handling friendship between users, there are several approaches to transforming the pairwise similarity between nodes of the same type (e.g. users) into a graph [17], such as the ϵ-neighborhood graph, k-nearest neighbor graph, and the fully connected graph. We choose the ϵ-neighborhood graph to fuse the friendship on the graph, not only because it has computational advantage by using a sparse representation of the data, but also because it can filter out the noisy data so that we can create a concrete friend set for each user. When constructing the ϵ-neighborhood graph, we connect user-node pairs whose friendship similarities are greater than ϵ. Given that the range of friendship similarity is [0,1], the range of ϵ is also [0,1]. We enumerate ϵ in this range with step 0.01, and finally we get the optimal ϵ.

When the graph was built, for the corresponding symmetric adjacency matrix A of graph G, the element a_{ij} was defined as: $a_{ij} = w_{ij}$ if node i is connected to node j and $a_{ij} = 0$ otherwise. Thus, people who listen to the same artists and join same groups, will be connected by a comparatively larger number of short paths.

3.2.2 Random Walk and Similarity Measures

Random walk is a mathematical formalization of a trajectory that consists of taking successive random steps. At each step, the next node in the walk is selected randomly from the neighbors of the last node in the walk. The sequence of visited nodes is a Markov Chain [20], with the transition probability:

$$p_{ij} = \begin{cases} \frac{1}{d(i)}, \text{ if } (i,j) \in E \\ 0 \quad , \text{otherwise} \end{cases} \tag{6}$$

where $d(i)$ is the degree of node i.

From the viewpoint of collaborative recommender systems, finding accurate neighbor set for each user is the cornerstone, so a good similarity measure on the graph is a crucial step.

Fouss et al. [13] has discussed several approaches to computing similarities between nodes of graph and their application to collaborative recommendations, that mainly included distance-based measure like ECTD, and inner-product based measure like L^+.

ECTD is the abbreviation for Euclidean Commute-Time Distance. Average commute-time is the average number of steps that a random walker, starting in node i ($i \neq j$), will take to enter node j for the first time and go back to i, represented as $n(i,j)$. Average commute-time is symmetric by definition, and it is a distance measure. $n(i,j)^{1/2}$ is also a distance in Euclidean space, and it is named as Euclidean Commute-Time Distance.

L^+ is the *Moore-Penrose pseudoinverse* of the Laplacian matrix L. The Laplacian matrix $\mathbf{L}$ of the graph is defined as, $\mathbf{L} = \mathbf{D}$ **-** $\mathbf{A}$, where $D = Diag(a_{i.})$, with $d_{ii} = [D]_{ii} = a_{i.} = \sum_{j=1}^{n} a_{ij}$, and A is the adjacent matrix of graph G. Let e be a column vector with 1(i.e., $e = [1, 1, \dots, 1]^T$, where T denotes the matrix transpose), then L^+ can be computed with the formula:

$$L^+ = (L - ee^T/n)^{-1} + ee^T/n, \tag{7}$$

where n is the number of nodes. If we define e_i as the ith column of $\mathbf{I}$, $e_i = [0, \dots, 0, 1, 0, \dots, 0]^T$(1 is in the ith column), then we can explain the relation between average commute-time and L^+ in the form

$$n(i,j) = V_G(e_i - e_j)^T L^+ (e_i - e_j), \tag{8}$$

where each node i is represented by a unit vector e_i in the node space spanned by $\{e_i\}$. It can be proved that the node vector e_i can be transformed into a new Euclidean space, and the elements of $L^+(l_{ij})$ are the inner products between these transformed node vectors. Therefore L^+ is a kernel matrix(a Gram matrix) and can be used as a similarity matrix for the nodes.

According to Fouss' study, the inner-product based similarity measure L^+ provides better and more stable results for collaborative recommenders, so we adopt L^+ metric to measure the similarity between users, which means in equation 5, $sim(u_i, u_j)$ equals to l_{ij} in this case.

4. EXPERIMENTS

4.1 Data Sets

Traditional data sets used in the evaluation of collaborative filtering systems, such as MovieLens, do not include explicit social relationship, while in *Last.fm*, a popular social music site, community information is available so that an entity-relation model can be generated which includes the relationship between users and items.

For our purpose, we extracted two typical social relationships: the friendship between users and the membership which describes the user's participation in groups, by accessing its Web Service APIs [1]. Besides, we think it is more meaningful to recommend artists instead of individual music since music is variable while preference on artists is more constant, so we use artist as the "item" in our recommendations. A user and an item is linked if the user listened to song(s) of the artist, and a user and a community is linked if the user joined the group. The relationship between an artist and a community is formed if the artist's songs were frequently listened by users in this group. There are also links among users describing their friendship.

We concretely established an active data set consisting of 943 users, 1,001 artists and 676 groups. There are 36,424 records in the user-artist matrix which sparsity degree (percentage of zero values in the matrix) is 96.14%, and 7,038 records in the user-group matrix which sparsity is 98.89%. The total number of friendship of 943 selected users is 33776, which means each user on average has 35.8 friends. Please note that the rating matrix here is the user-artist matrix, and if a user listened to song(s) of an artist, there is "1" in the corresponding cell.

By means of 5 fold cross-validation [2], each row (represent a user) of the user-artist matrix is randomly split into five different sets. For each time of experiment, four-fifths of of the data is included in the training set and the other is used as the testing data.

4.2 Evaluation Metrics

We adopted standard metrics in the area of information retrieval to evaluate our recommenders. During each round of cross-validation, we recommend and rank a set of potential artists for each user. We then compare the predicted recommendation list with true preferences on artists in the test set, and compute precision, recall, and F-measure scores.

1. **Recall.** The score measures the average (on all users) of the proportion (in percentages) of artists from the testing sets that appear among the top n ranked list from the training sets, for some given n. It should be as high as possible for good performance. We computed the recall from Top-1 to Top-20 artists(for a total of 1001 artists).

2. **Precision.** This metric measures the proportion of recommended items that are ground truth items. Note that the items in the profiles of the testing data represent only a fraction of the items that the user truly accessed.

[1]The web page is `http://www.last.fm/api`
[2]The number of fold is the number of tested sets.

3. **F-measure.** F-measure is the weighted harmonic mean of precision and recall. The equation is as follows:

$$F = \frac{2 * precision * recall}{(precision + recall)} \tag{9}$$

In the following, we report the results of recommending the top 1, 2, 5, 10 and 20 artists. At each pass, 50 users are taken as neighbors based on different similarity measures for recommendation.

4.3 Experiment Design and Results

At first, we evaluated the fusion approach via weighted-similarity. A user-based collaborative filtering recommender was first run on the user-artist rating matrix, resulting in the baseline represented by CF_{UI} (see Tables 1 to 3 respectively showing precision, recall and F-measure scores).

Table 1: Precision of Fusion via Weighted-Similarity Approaches

Precision	Top 1	Top 2	Top 5	Top 10	Top 20
CF_{UI}	**29.93**	25.12	19.09	15.23	11.33
WS_{fri+UI}	29.67	25.33	**19.64**	**15.51**	**11.48**
WS_{mem+UI}	29.44	**25.49**	19.58	15.33	11.40
$WS_{fri+mem+UI}$	29.37	25.45	19.58	15.34	11.42

Table 2: Recall of Fusion via Weighted-Similarity Approaches

Recall	Top 1	Top 2	Top 5	Top 10	Top 20
CF_{UI}	**3.87**	6.50	12.36	19.72	29.34
WS_{fri+UI}	3.84	6.56	**12.71**	**20.07**	**29.73**
WS_{mem+UI}	3.81	**6.60**	12.67	19.84	29.52
$WS_{fri+mem+UI}$	3.80	6.59	12.68	19.86	29.58

Table 3: F-measure of Fusion via Weighted-Similarity Approaches

F-measure	Top 1	Top 2	Top 5	Top 10	Top 20
CF_{UI}	**6.85**	10.33	15.01	17.19	16.35
WS_{fri+UI}	6.80	10.42	**15.43**	**17.50**	**16.56**
WS_{mem+UI}	6.75	**10.49**	15.38	17.29	16.45
$WS_{fri+mem+UI}$	6.73	10.47	15.39	17.31	16.48

Then, we tried to fuse friendship with the rating matrix and used λ to adjust the weight of rating matrix and friendship while computing user similarity. Figure 3 shows how F-measure changes with the changing of λ. It achieves the peak when $\lambda = 0.7$, which means that the rating matrix contributes to 70% percent of the weight while friendship contributes to 30% in calculating the user-user similarity. Specifically, results in the second rows of Tables 1 to 3, represented by WS_{fri+UI}, give the precision, recall and F-measure scores when $\lambda = 0.7$. It can be seen that it is better than the baseline, especially when returning top 5 recommendations (the improvement of F-measure achieved up to 2.79%). In the process of tuning λ and β in the following, we considered the average score of Top-1 to Top-20 recommendations.

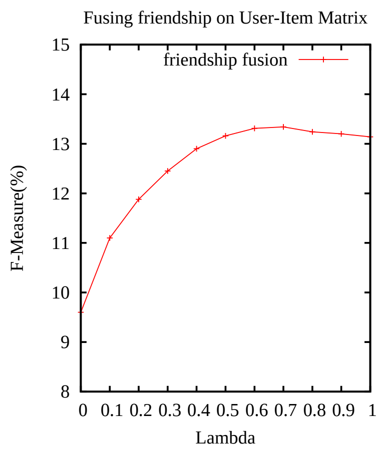


Figure 3: F-measure changes as lambda change in friendship fusion

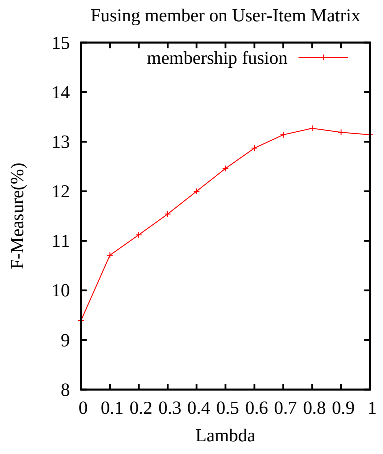


Figure 4: F-measure changes as lambda change in membership fusion

Next, we fused membership with rating matrix and also used the parameter λ to adjust the weight of rating matrix and membership in the similarity calculation.

From the figure 4, it can be seen that when $\lambda = 0.8$, we can get best results overall. The third rows of Tables 1 to 3, represented by WS_{mem+UI}, respectively show precision, recall and F-measure values when $\lambda = 0.8$. It shows that the results slightly improve on the baseline, but are a little weaker than the fusion of friendship on the rating matrix.

We further fused two relationships together on the rating matrix and adopted two parameters: λ and β to adjust the weights for the three data sources as shown in Formula 4. Experimental results indicate that the hybrid fusion performs best when lambda = 0.8 and beta = 0.5 (which means rating matrix contributes 80%, friendship and membership respectively contributes 10% and 20% in the user-user similarity calculation). These results are illustrated in the last rows of Tables 1 to 3. To our surprise, compared to fusion of friendship and membership separately, it did not have dis-

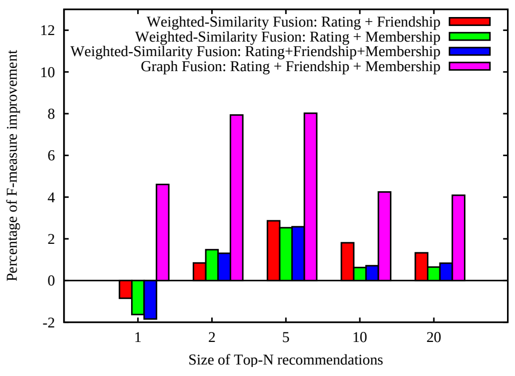


Figure 5: Improvements on F-Measure of All the Fusion Approaches

tinct difference and was actually even a little weaker than the pure fusing of friendship.

In order to identify whether the fusion via graph model would perform better than via weighted similarity approach given that social data are inherently in the graph structure, we finally did the experiment of fusing the two social relationships on the graph and applied random walk algorithm to calculate neighborhood similarity.

Based on the graph construction method described before, we firstly run a series of simulations to learn the optimal ϵ. After running 100 times, the optimal ϵ that we got is 0.05. The comparative results under threshold 0.05 are then computed and listed in Table 5. It shows that precision, recall and F-measure scores are all highly improved compared to the fusion via weighted-similarity approach. In particular, when returning top 2 and top 5 recommendations, the improvements significantly reached 7.94% and 7.99% respectively.

Table 4: Fusing friendship and membership via Graph

$G_{fri+mem+UI}$	Top 1	Top 2	Top 5	Top 10	Top 20
Precision	31.30	27.12	20.62	15.88	11.79
Recall	4.05	7.02	13.35	20.55	30.54
F-Measure	7.18	11.15	16.21	17.92	17.01

Figure 5 further shows that while returning Top-1 recommendation, the fusion via graph can achieve an improvement at 4.82%, and others are however all slightly weaker than the baseline. When returning top N recommendations from 2 to 10, all of the fusion approaches enhance the baseline CF method, which positively proves the usefulness of social relationship data especially when multiple recommendations are computed. It also indicates that the fusion via graph can boost the baseline significantly by up to 8%, demonstrating that the graph model is a more proper way to fuse explicit social relationships.

5. CONCLUSION AND FUTURE WORK

In this paper, we presented two principal methods to integrate explicit social relationships into traditional CF methods: the weighted-similarity fusion and the graph fusion. We demonstrated the effectiveness of social relationships in augmenting recommendations, and particularly that the graph-based fusion is more effective in bringing into play of the power of social data. To the best of our knowledge, the work is one of the first attempts to explore the effect of membership in addition to friendship, and to fuse both of them based on random walk graph model with collaborative filtering (CF) systems.

Table 5: Table of improvements on F-Measure by Comparing with the Baseline

Improvements	Top 1	Top 2	Top 5	Top 10	Top 20
WS_{fri+UI}	-0.73	0.87	2.80	1.80	1.28
WS_{mem+UI}	-1.46	1.55	2.47	0.58	0.61
$WS_{fri+mem+UI}$	-1.75	1.36	2.53	0.70	0.80
$G_{fri+mem+UI}$	4.82	7.94	7.99	4.25	4.04

For the next step, we are interested in further exploring the impact of social relationships on recommender systems from three aspects: one is to explore other potential relationships, such as the relation between items and associated groups, other social relationships besides friendship and membership, such as the reporting chain in a company, to see how to model and utilize these data in order to make better recommendations; another direction is to explore how to enhance Random Walk model so as to handle heterogeneous data in a more fine-grained way, based on the method proposed in [25]; finally, as the explosion of the size of social websites, we need to pay more attention to the algorithm's scalability and efficiency, when the social graph grows with millions of nodes.

6. ACKNOWLEDGMENTS

We thank Lawrence Bergman of IBM T.J. Watson Research Center for his generous help and valuable comments on this work.

7. REFERENCES

[1] C. Lam. Snack: incorporating social network information in automated collaborative filtering. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 254–255, New York, NY, USA, 2004. ACM.

[2] E. Spertus, M. Sahami, and O. Buyukkokten. Evaluating similarity measures: a large-scale study in the orkut social network. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 678–684, New York, NY, USA, 2005. ACM.

[3] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. In J. Huai, R. Chen, H.-W. Hon, Y. Liu, W.-Y. Ma, A. Tomkins, and X. Zhang, editors, *WWW*, pages 895–904. ACM, 2008.

[4] D. Boyd. Friends, friendsters, and myspace top 8: Writing community into being on social network sites. http://www.firstmonday.org/issues/issue11_12/boyd/index.html, December 2006.

[5] N. Baym. How Good A Friend Is A Last.fm Friend? http://www.last.fm/user/popgurl/journal/2008/04/28/3d9l_how_good_a_friend_is_a_last.fm_friend.

[6] M. Balabanović and Y. Shoham. Fab: content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997.

[7] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Eighteenth national conference on Artificial intelligence*, pages 187–192, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[8] M. J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artif. Intell. Rev.*, 13(5-6):393–408, 1999.

[9] J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In G. F. Cooper and S. Moral, editors, *Proceedings of the 14^{th} Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[10] H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 931–940, New York, NY, USA, 2008. ACM.

[11] J. Golbeck. Generating predictive movie recommendations from trust in social networks. pages 93–104. 2006.

[12] P. G. Doyle and J. L. Snell. Random walks and electric networks, 2000.

[13] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *Knowledge and Data Engineering, IEEE Transactions on*, 19(3):355–369, March 2007.

[14] M. Gori and A. Pucci. Itemrank: A random-walk based scoring algorithm for recommender engines. In M. M. Veloso, editor, *IJCAI*, pages 2766–2771, 2007.

[15] G. Groh and C. Ehmig. Recommendations in taste related domains: collaborative filtering vs. social filtering. In *GROUP '07: Proceedings of the 2007 international ACM conference on Supporting group work*, pages 127–136, New York, NY, USA, 2007. ACM.

[16] R. Gross, A. Acquisti, and J. H. Heinz. Information revelation and privacy in online social networks. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80, New York, NY, USA, 2005. ACM Press.

[17] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007.

[18] Z. Huang. *Graph-based analysis for e-commerce recommendation.* PhD thesis, Tucson, AZ, USA, 2005. Adviser-Hsinchun Chen and Adviser-Daniel D. Zeng.

[19] H. G. Hummel, B. van den Berg, A. J. Berlanga, H. Drachsler, J. Janssen, R. Nadolski, and R. Koper. Combining social-based and information-based approaches for personalised recommendation on sequencing learning activities. *International Journal of Learning Technology*, 3:152–168(17), 12 August 2007.

[20] J. G. Kemeny and J. L. Snell. *Finite Markov Chains.* Springer-Verlag, 1976.

[21] I. Konstas, V. Stathopoulos, and J. M. Jose. On social networks and collaborative recommendation. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 195–202, New York, NY, USA, 2009. ACM.

[22] I. Guy, I. Ronen, and E. Wilcox. Do you know?: recommending people to invite into your social network. In *IUI '09: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 77–86, New York, NY, USA, 2009. ACM.

[23] J. Chen, W. Geyer, C. Dugan, M. Muller, and I. Guy. Make new friends, but keep the old: recommending people on social networking sites. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 201–210, New York, NY, USA, 2009. ACM.

[24] J. Wang, A. P. de Vries, and M. J. T. Reinders. A user-item relevance model for log-based collaborative filtering. In M. Lalmas, A. MacFarlane, S. M. Rüger, A. Tombros, T. Tsikrika, and A. Yavlinsky, editors, *ECIR*, volume 3936 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 2006.

[25] J. Zhang, J. Tang, B. Liang, Z. Yang, S. Wang, J. Zuo, and J. Li. Recommendation over a heterogeneous social network. *Web-Age Information Management, International Conference on*, 0:309–316, 2008.

Spreading Activation Approach to Tag-aware Recommenders: Modeling Similarity on Multidimensional Networks

Alexander Troussov
IBM, Ireland
IBM Software Lab, Bld. 6,
Mulhuddart, Dublin 15, Ireland
+353-1-815 1906
atrousso@ie.ibm.com

Denis Parra
University of Pittsburgh and
CNGL at Trinity College Dublin
135 North Bellefield Ave., Pittsburgh,
PA 15260, USA
+1 (412) 624 9403
dap89@pitt.edu

Peter Brusilovsky
University of Pittsburgh
135 North Bellefield Ave.,
Pittsburgh, PA 15260, USA
+1 (412) 624 9404
peterb@mail.sis.pitt.edu

ABSTRACT

Social tagging systems present a new challenge to the researchers working on recommender systems. The presence of tags, which uncover the reasons of user interests to tagged items, opens a way to increase the quality of recommendations. Yet, there is no common agreement of how the power of tags can be harnessed for recommendation. In this paper we argue for the use of spreading activation approach for building tag-aware recommender systems and suggest a specific version of this approach adapted to the multidimensional nature of social tagging networks. We introduce the asymmetric measure of relevancy (proximity) of two nodes on a multidimensional network as a cumulative strength of (weighted) multiple connections between two nodes, which includes paths and graph-structures connecting the nodes. This metric is also applicable to measure relevancy of two sub-graphs. Spreading activation methods (SAM), which usually employ breadth first search, are an efficient way to define and compute such measure taking into account not only links constituent a path, but the properties of nodes in the path such as node's types and outdegree.

We apply this notion of relevancy to measure similarity of collaborative tagging systems users and present the results of numerical simulation showing that spreading activation methods allow us to discriminate between diverse graph-structures connecting users via resources and tags. We show that the results of simulation are stable w.r.t. the variation of parameters of spreading activation algorithm used in our experiment.

Categories and Subject Descriptors
H.3.4 [**Information Storage and Retrieval**]: Systems and Software – *information networks*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *data sharing.*

General Terms
Algorithms, Measurement, Performance, Experimentation.

Keywords
Tagging, relevancy propagation, spreading activation, graph-based mining, structural cohesion, CiteULike.

1. INTRODUCTION

Social tagging systems introduced new challenges to the well-established area of recommender systems. While the majority of content based, collaborative, and hybrid recommender approaches were created for a bi-modal world of items and users (connected by rating incidents), social tagging systems present a more complicated world of users, items, and tags (connected by tagging incidents, also known as tagging instances). While some early works attempted to treat the problem of recommendation in social tagging systems in an "old way", basically ignoring the tags, the majority of researchers in this new area argued that tags are vital for successful recommendation in this new domain and called for tag-aware recommenders. They argued that on one hand, tags can compensate the loss of ratings (which are not available in most social tagging systems), while on the other hand, tags can make recommendation more precise because they provide not only the information of what items are of interest to a user, but also why they are of interest [8,14,20,26].

Despite the common agreement that tags should be used as a successful recommender component of a social tagging system, there is no agreement on how it should be done. As a result, a multitude of approaches emerged just over the last three years. Roughly, these approaches can be classified as an extension of either content-based or collaborative filtering approaches. The former group emphasizes connections between items and tags treating tags as an alternative (or additional) way to describe items and establish a profile of user interests [9, 15]. The latter group emphasizes connections between users and tags to establish a better similarity between users in a social tagging system [25, 26].

We argue than inherently networked nature of social tagging systems calls for some alternative recommender approaches, which are not just simple extension of either content-based or collaborative technologies. A successful recommender approach for this new context should fully employ the complex network structure of a typical social tagging system and use all kinds of links: user-tag, item-tag, user-item. We think that the most promising in this context is the spreading activation approach. This approach has been originally developed in the field of cognitive psychology [3] to model human brain and later explored in the context of information retrieval [7].

The power of spreading activation approach was recognized in the area of recommenders and other personalized systems as well; however, so far these approaches form just a small minority. The problem is that the traditional user-item universe does not provide a sufficiently rich network for spreading activation technology. Thus most of known recommenders based on spreading activation were built for context where an additional network can be formed such as a hypertext network for Web page recommendation [17] or a network of entities and concepts in semantically enriched recommenders [4, 12, 18].

We believe that social tagging systems will provide a new promising context as well as new challenges for recommenders based on spreading activation. What we consider as the main challenge is the multidimensional nature of a typical social tagging network. Almost all existing applications of spreading activation for personalization and recommendation operated in relatively homogeneous kinds of networks with 1-2 kinds of nodes and one kind of links. In contrast, even a simplified social tagging network, where each tagging event is represented by a group of three links (user-tag, item-tag, and user-item) includes three types of nodes and three types of asymmetric links. This organization requires some more sophisticated spreading activation approaches.

Our paper attempts to address this challenge by introducing the asymmetric measure of relevancy (proximity) of two nodes on a multidimensional network as a cumulative strength of (weighted) multiple connections between two nodes which includes paths and graph-structures connecting the nodes. This metric is also applicable to measure relevancy of two sub-graphs. Spreading activation methods, as breadth first search, is an efficient way to define and compute such measure taking into account not only links constituent a path, but the properties of nodes in the path such as node's types and outdegree.

We apply this notion of relevancy to build a tag-aware approach to measure similarity between users in collaborative tagging systems. The paper presents the results of a numerical simulation showing that spreading activation algorithms allow discriminating the degree of connectivity of users between certain graph-structures connecting users via resources and tags. We demonstrate that the results of the simulation are stable w.r.t. the variation of parameters of the spreading activation algorithm used in our experiment.

The rest of the paper is organized as follows. In section 2 we first provide a short overview of related work focusing on the use of spreading activation methods (SAM) to propagating and redistributing relevancy. We also theorize about desired properties of relevancy propagation on multidimensional network models of Web. 2.0 data needed to create efficient and scalable recommender systems.

In section 3 we render a formal model of folksonomies (tripartite hypergraph) as a multidimensional network with four types of nodes corresponding to users, resources, tags and instances of tagging. In section 4 we present the results of numerical simulation. Finally, section 5 describes the conclusions and future work

2. RELATED WORK

2.1 Overview of Relevancy Propagation Using Spreading Activation Methods

In neurophysiology interactions between neurons are modeled by way of activation which propagates from one neuron to another via connections called synapses to transmit information using chemical signals. The first spreading activation models were used in cognitive psychology to model these processes of memory retrieval [5, 3]. This framework was later exploited in Artificial Intelligence as a processing framework for semantic networks and ontologies, and applied to Information Retrieval [2, 7, 19] as the result of direct transfer of information retrieval ideas from cognitive sciences to AI. In other domain, [27] created spreading activation models for trust propagation on the Web.

In [21] and [23] authors work with the notion of the relevancy of ontological concepts to a free text. They propagate relevancy of the concepts explicitly mentioned in a document to other ontological concepts using a spreading activation algorithm. Their algorithm works in such a way, that after short number of iteration the topical foci of a cohesive coherent text become the most activated concepts (even if they were not explicitly mentioned in the text).

In [22] authors summarize their experience in creating graph-based related item recommender for activity centric environment on a Nepomuk Social Semantic Desktop [24]: relevancy of a "pile" of nodes representing resources and concepts is propagated to other nodes. Authors in [22] conclude that as a graph-mining technique, spreading activation combines fuzzy clustering and soft inferencing, and therefore might be suitable for relevancy propagation. Propagation should lead to discovery of new nodes which have short length paths to many (if not all) nodes from the initial set. In other words, newly discovered nodes should minimize the "distance" to the initial set of nodes, i.e., nodes which might be considered as potential centroids of strong clusters induced by the initial conditions. Since partitioning of the nodes according to these clusters is not needed, processing of polycentric queries [22] for related item recommendation could be done using soft clustering methods. On the other hand, relevancy propagates through links. an alternative view on the related item recommendation is that newly discovered nodes must be connected to the initial conditions by particular types of directed links. Therefore, propagation of relevancy might be interpreted as fuzzy inference.

In [23], the authors go further in analyzing SAM as a very general class of iterative algorithms for relevancy propagation, local search, relationship/association search, and computing of dynamic local ranking. Authors indicate that the same iterative algorithms were used long before in numerical simulation in physics, mechanics, chemistry, and engineering sciences. Hence, the algorithm is quite polymorphic: "Using the same iterative algorithm, with one set of parameters one can emulate heat transfer; with another set of parameters the same algorithm will show us the behavior of oscillating strings".

2.2 Spreading Activation in Recommender Systems

Spreading activation approach as a technology for recommendation in various kinds of networks belongs to a broader group, which is typically referred to as graph-based approaches for recommendation. In addition to several recent papers mentioned in the introduction, which explicitly use spreading activation to build recommender systems, we can a few other examples of using various graph-based approaches. In [1], the authors presented a theoretic approach where users are modeled as nodes in a directed graph and the directed links represent how representative is a user of another user's behavior. In [11], the authors use spreading activation to deal with the sparsity problem in collaborative filtering. They try to tackle the problem finding transitive relationships by comparing three different methods on a bipartite graph which represented consumer-product interactions. Other interesting approach was the one presented in [10], where the authors propose a constrained spreading activation algorithm having good results compared with a traditional memory-based approach over a small subset of the Movie Lens data set. These approaches show the potential of spreading activation to be used on recommender systems, but they don't take into account the nature of multidimensional networks, such as folksonomies derived from collaborative tagging systems, where different types of nodes, links and relationships can have a strong influence in the design of the algorithms.

2.3 Propagating Relevancy on Multidimensional Web 2.0 Networks

We focus on the applications of SAM to measure similarity between the users of collaborative tagging systems modeled as multidimensional networks. Indeed, we treat graph-based "similarity" of users as a particular case of "relevancy" of nodes on multidimensional networks. In this subsection we provide consideration on which properties of a generic class of spreading activation algorithms are suitable methods for modeling relevancy propagation.

The general inspiration behind using graph-based methods to model relevancy (energy, trust, risk, etc.) propagation on networks is probably the same in many domains: the relevancy is treated as a kind of energy which might be "injected" into some nodes, and propagated through links to other nodes: "… the closer node x to the injection source s, and the more paths leading from s to x, the higher the amount of energy flowing into x in general" [27]. Therefore, spreading activation methods (SAM), which usually employ breadth-first search), are an efficient way to propagate relevancy. Since according [23] SAM is a broad class of algorithms, the choice of algorithm's parameters is crucial and can be done taking into account the nature of the target application.

First of all, Web 2.0 data could be accurately modeled only by multidimensional networks. For instance, formal model of a folksonomy as tripartite hypergraph [13] converted to network representation, has four types of nodes: users, resources, tags, instances of tagging. The shortest possible path between two folksonomy users has the length four (for instance, user1- instance of tagging1- tag - instance of tagging2 - user2). As compared to trust propagation in heterogeneous networks, the amount of relevancy flowing from one node to another should depend not only on types of links, but on properties on nodes in paths. Connections via resources might be more important than connections through tags. In our future work we are going to exploit what [23] calls "the importance of nodes", but one property of nodes which should significantly affect the propagation, can be immediately inferred from the local topology of the network, namely from the number of outcoming links from a node. Ambiguous and top popular tags might be linked to big number of tag instances and big number of users. Intuitively, connections via such tags should provide less (if any) contribution to the similarity of users as compared to the connections through less popular tags.

In [27], the authors assume that nodes with the higher shortest path distance from the injection source should be accorded less trust in general. This property of trust propagation is probably not applicable to propagating relevancy to measure similarity of folksonomies users. Moreover, we suggest that for many applications on multidimensional networks the length of the shortest path might have positive correlation with the relevancy, but is probably much less important and is too coarse-grained measurement compared to trust propagation.

A final observation on relevancy propagation on multidimensional networks: we don't assume that all (or many) aspects of such propagation can be properly understood in terms of paths. We assume that there might be structures (like network B on the Fig. 1), which might significantly affect the relevancy propagation.

3. THE ALGORITHM

The algorithm we used in our experiment in general follows [23] and employs iterative steps where activation is propagated between neighbor nodes. To facilitate comparison of activation distributions on the same or different networks and to account for dissipation of activation caused by list purging step in spreading activation, we introduce the step of normalization (calibration).

A multidimensional network can be modeled as a directed graph, which is a pair $G = (V,E)$ where

V – is the set of vertices v_i

E – is the set of arcs e_j

init: $E \rightarrow V$, is the mapping that provides initial nodes for arcs

term: $E \rightarrow V$, is the mapping that provides terminal nodes for arcs

imp – is importance value of arcs and nodes.

w – "weights"

$F(E)$ – is the "activation" real valued function

The algorithm has the following steps

Initialization

Sets the parameters of the algorithm, network, and initial $F(E)$ as a list of non-zero valued nodes V_n

Iterations

a. List Expansion.

b. Recomputation: The value at each node in the list is recomputed based on the values of the function on nodes which have links to the given node and types of connections.

c. List Purging: We exclude the nodes with the values less than a threshold.

d. Conditions Check To Break Iterations.

Normalization

Linear scaling up or down the numerical values of the activation level of all nodes in the list of activated nodes to satisfy some conditions of activation conservation

Output

The list of nodes (value of the function after spread of activation) ranked according F values.

Recomputation step is as follows:

- We have the list of nodes Vn.
- **Input/Output Through Links Computation.**

- For each node v we compute the input signal to each arc e, such that init(e)=v. This computation can be based on the value F(v), the outdegree of a node etc. For instance, if the node v has n outgoing arcs of the same type, each arc e might get input signal:

  $$I(e) = F(\text{init}(e)) \cdot (1 / \text{outdegree}(v) \text{^beta})$$

  where beta might be equal to 1. It could be also less than one, in which case the node v will propagate more activation to its neighbors than it has. (This might be fine for some applications).

- When the signal ("activation") passes through a link e, the activation usually experiences decay by a factor $w(e)$:

  $$O(e) = I(e) \cdot w(e)$$

▪ **Input/Output Of Node Activation**

- Before the pulse, the node v has the activation level $F(v)$.
- Through incoming links v get more activation:

  $$\text{Input}(v) = \Sigma\ O(e)$$

  for all links e such that init(e) $\in Vn$, term(e) = v.

- By dissipating the activation through outgoing links, the node v might lose activation:

  $$\text{Output}(v) = \Sigma\ I(e)$$

  for all links e such that init(e) = v, term(e) $\in Vn$

▪ **Computation Of New Level Of Activation**

$$\text{Fnew}(v) = F(v) + \text{Input}(v)$$

To apply spreading activation to measure "similarity" of two nodes on a network, we put the initial activation 1.0 at the first node, and measure the activation at the second node after certain number of iterations.

4. EXPERIMENTS

To apply graph-based mining on web 2.0 data we model the data by a multidimensional network (where nodes and links are typed, and links are "weighted").

In our experiments we use three networks representing instantiations of collaborative tagging systems. Each of these networks has two actors (A1 and A2), two resources (R1 and R2), and four instances of tagging (I1, I2, I3 and I4). For instance, the network A on the figure 1 has the instance of tagging *I1* with links to the actor *A1*, the resource *R1*, and the tag *T*; this sub-network shows that the actor *A1* used the tag *T* for the resource *R1*. Correspondingly, the links from the instance *l2* show that the actor *A1* used the tag *T* for the resource *R2*. The instances *I3* and *I4* show tagging for the user *A2*. The network *A* represents the situation where both actors used the same tag for both resources.

In the implementation of our algorithm, each of these networks is modeled by a directed graph, where for each link we create two reciprocal arcs. In each experiment we set initial activation at the node corresponding to the actor A1 and after several iterations of the algorithm we compute the "similarity" of actors A1 and A2 using the method described in 3.

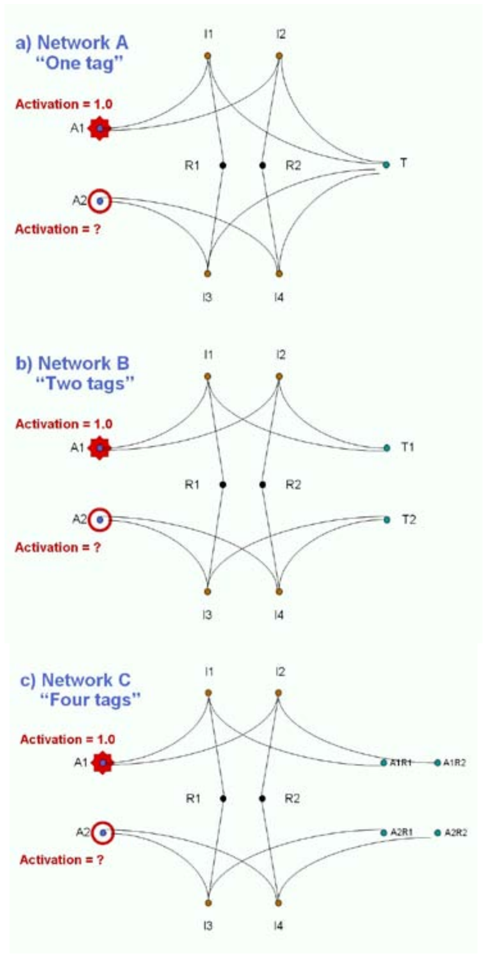


Figure 1. Three networks modeling instantiations of collaborative tagging systems.

In [23], the authors view SAM in terms of graph-mining algorithms as a technique for soft clustering. The major parameters of SAM affecting "the scale" of the phenomena to be discovered are signal decay and number of iterations (larger number of iterations and low decay are needed to discover "bigger" clusters). Since Web 2.0 applications are at the focus of this paper, we run the experiments varying these two parameters. Our target was to find regions of the parameters which allow us consistently to capture structures like that on the Fig.1.

In this paper, we use SAM as a link analysis algorithm for local ranking, in the same way as PageRank algorithm is used for global ranking [28]. The major difference between them is that PageRank iteratively redistributes the relevancy measure which is initially set to each node of the network, while we use SAM to iteratively redistribute the relevancy measure (the activation) from one (or more) nodes sometimes referred to as "seeds".

Diameter of graphs B, and C is 6, with the number of iterations less than 6 the activation from a node on a network will not necessarily reach all the nodes. The limit distribution (distribution of the activation after a number of iterations big enough), produced by SAM, in general does not depend on the choice of the initial seed. This behavior gives us the estimate that local ranking, which is highly sensitive to sub-graphs with the diameter

6, could be achieved when the activation will be redistributed on such sub-graphs several times which amounts roughly to 12-48 iterations.

Our underlying common-sense assumption is that connectivity of A1 and A2 is bigger in the network A than in B and C; and that the connectivity of A1 and A2 in the network B is bigger than in the network C. In other words, if we denote the final activation of the node *v* in the network configuration *X* as *x(v)*, we would expect that sensible local ranking results should satisfy inequality:

$$a(A2) > b(A2) > c(A2) \qquad (1)$$

The shortest path between the nodes A1 and A2 equals to 4 in the network A, and to 6 in networks B and C. So the first part of the inequality is easily achieved with any parameters of the algorithm (provided that the number of iterations is not less than 3). To investigate how the algorithm can discriminate between configurations B and C we introduce the *network discrimination factor* as

$$NDF = \frac{b(A2) - c(A2)}{a(A2) - c(A2)} \qquad (2)$$

We computed the *NDF* ranging the number of iterations from 1 to 50, and the decay factor from 0 to 1. Figure 2 shows the results, where the X axis represents number of iterations, the Y axis the decay factor, and the Z axis the network discrimination factor.

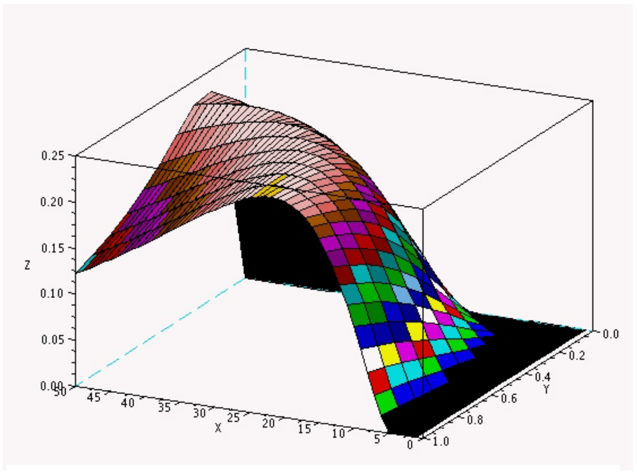


Figure 2. Results of the NDF experiment. Axis *X* shows iterations, axis *Y* decay values, and axis *Z* the NDF.

The results in figure 2 show that we maximize the NDF when running our spreading activation algorithm with a decay factor between 0.8 and 0.9, and 24 iterations. Additionally, the plot shows stable results for our algorithm, which suggests that selecting values in close ranges will not return unexpected or random activation values.

We have shown that on small networks SAM might be used to measure similarity between users. It is part of our future plans to show that on big multidimensional networks representing Web 2.0 data activation initiated at one of the nodes could be kept flowing within strong clusters induced by the initial set of activated nodes (because of high degree of clustering); and therefore the results could be generalized to real-world data.

5. CONCLUSIONS AND FUTURE WORK

Our paper argued for the use of spreading activation as a recommendation mechanism in multidimensional networks produced by collaborative tagging systems. We introduced the new network-based asymmetric measure of relevancy of two nodes on a multidimensional network and applied it to build a tag-aware approach to measure similarity between users in collaborative tagging systems. While it is just one of several possible ways to use spreading activation in collaborative tagging context, we consider it as the best way to start. As demonstrated by the stream of recent works, calculating similarity between users is a component of the recommendation process where the use of tags can provide a most valuable impact [25, 26].

The results of our experiments show that our metrics can be used to differentiate activation levels on different network configurations and they also show a stable behavior when input parameters are changed. These results lead us to pass to the next step on our research on this bottom-up approach, which is to prove that our results are repeatable in large scale networks. We are currently running our experiments on real social network data that we have collected from the social bookmarking service CiteUlike.

In this paper we presented applications of spreading activation methods to local ranking on small networks. We didn't prove yet that the same "good" properties hold true when the algorithm runs on massive networks. However, multidimensional networks which model web 2.0 data and processes usually exhibit small world phenomena properties, which include small average distance and clustering effect. According to [23] spreading activation might be considered as a method for soft clustering. Intuitive justification of the use of spreading activation for ranking is the same as for the PageRank algorithm [28]: a node can have a high rank if there are many nodes that point to it, or if there are some nodes that point to it and have a high rank. On each iteration strongly activated nodes continue to support the high level of activation of nodes to which they have outcoming links, while nodes which have little connection with strongly activated nodes eventually lose their activation. Therefore, even if constrained spread of activation from one node might in several iterations reach significant portion of the network (small average distance), strong level of activation will be supported mainly in strong clusters induced by the node.

6. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 0840597. We also want to thank Dr. Vincent Wade from the CNGL of the Trinity College Dublin for his support to work on this collaborative research.

Dr. Alexander Troussov's work was done in collaboration with CNGL, which is funded under Science Foundation Ireland's CSET programme: Grant# 07/CE2/I1142.

7. REFERENCES

[1] Aggarwal, C. C., Wolf, J. L., Wu, K., and Yu, P. S. 1999. Horting hatches an egg: a new graph-theoretic approach to collaborative filtering. In *Proceedings of the Fifth ACM SIGKDD international Conference on Knowledge Discovery and Data Mining* (San Diego, California, United States, August 15 - 18, 1999). KDD '99. ACM, New York, NY, 201-212. DOI= http://doi.acm.org/10.1145/312129.312230

[2] Aleman-Meza, B., Halaschek, C., Arpinar, I., & Sheth, A. (2003). Context-Aware Semantic Association Ranking. Proceedings of SWDB'03, Berlin, Germany, 33-50.

[3] Anderson, J., 1983. A Spreading Activation Theory of Memory. Journal of Verbal learning and Verbal Behavior 1983, (22), 261-295.

[4] Bier, E. A., S. K. Card, et al. 2008. Entity-Based Collaboration Tools for Intelligence Analysis. IEEE Symposium on Visual Analytics Science and Technology, VAST 2008, Columbus, Ohio, IEEE.

[5] Collins, A.M. & Loftus, E.F. 1975. A spreading-activation theory of semantic processing. Psychological Review. 1975 Nov Vol 82(6), 407-428.

[6] Contractor, N. 2007. From Disasters to WoW: Using a Multi-theoretical, Multilevel Network Framework to Understand and Enable Communities. Retrieved March 8, 2009, from http://www.friemel.com/asna/keynotes.php

[7] Crestani, F. 1997. Application of Spreading Activation Techniques in Information Retrieval. Artificial Intelligence Review, 11(6), 453-482.

[8] Dattolo, A., F. Ferrara, et al. 2009. Supporting Personalized User Concept Spaces and Recommendations for a Publication Sharing System. 17th International Conference on User Modeling, Adaptation, and Personalization (UMAP 2009), Trento, Italy, Springer.

[9] de Gemmis, M., P. Lops, et al. 2008. Integrating tags in a semantic content-based recommender. the 2008 ACM conference on Recommender systems, RecSys '08 Lausanne, Switzerland, ACM.

[10] Griffith, J., O'riordan, C., and Sorensen, H. 2006. A constrained spreading activation approach to collaborative filtering. pp. 766-773.

[11] Huang, Z., Chen, H., and Zeng, D. 2004. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Trans. Inf. Syst.* 22, 1 (Jan. 2004), 116-142.

[12] Hussein, T. and J. Ziegler 2008. Adapting web sites by spreading activation in ontologies. ReColl '08: Int. Workshop on Recommendation and Collaboration (in conjunction with IUI 2008).

[13] Mika, P 2007. Ontologies are us: A unified model of social networks and semantics. J. Web Sem. 5(1): 5-1

[14] Nauerz, A., S. Pietschmann, et al. 2008. Using Collective Intelligence for Adaptive Navigation in Web Portals. 3rd International Workshop on Adaptation and Evolution in Web Systems Engineering at 8th International Conference on Web Engineering 2008, Yorktown Heights, New York, USA.

[15] Niwa, S., T. Doi, et al. 2006. Web Page Recommender System based on Folksonomy Mining for ITNG'06 Submissions. Third International Conference on Information Technology: New Generations, ITNG 2006.

[16] Rocha, C, Schwabe, D., & Poggi de Aragao, M. 2004. A Hybrid Approach for Searching in the Semantic Web. Proceedings of the 13th international conference on WWW, May 17-20, 2004, New York, NY, USA, 374-383.

[17] Olston, C. and E. H. Chi 2003. "ScentTrails: Integrating browsing and searching on the Web." ACM Transactions on Computer-Human Interaction 10(3): 177-197.

[18] Sarini, M. and C. Strapparava 1998. Building a User Model for a Museum Exploration and Information-Providing Adaptive System. Second Adaptive Hypertext and Hypermedia Workshop at the Ninth ACM International Hypertext Conference Hypertext'98, Pittsburgh, PA.

[19] Schumacher, K., Sintek, M., Leo Sauermann 2008 Combining Fact and Document Retrieval with Spreading Activation for Semantic Desktop Search. The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Spain, June 1-5, 2008 LNCS, Springer Verlag, Volume 5021/2008, 569-583

[20] Shepitsen, A., J. Gemmell, et al. 2008. Personalized recommendation in social tagging systems using hierarchical clustering. the 2008 ACM conference on Recommender systems, RecSys '08 Lausanne, Switzerland, ACM.

[21] Troussov, A., Judge, J., & Sogrin, M. 2007, December 13). IBM LanguageWare Miner for Multidimensional Socio-Semantic Networks. Retrieved March 8, 2009, from http://www.alphaworks.ibm.com/tech/galaxy

[22] Troussov, A., Judge, J., Sogrin, M., Bogdan, C., Edlund, H., & Sundblad, Y. 2008b. Navigating Networked Data using Polycentric Fuzzy Queries and the Pile UI Metaphor Navigation. Proceedings of the International SoNet Workshop, 5-12.

[23] Troussov, A., Levner, E., Bogdan, C., Judge, J., Botvich, D. "Spread of Activation Methods", in Dynamic and Advanced Data Mining for Progressing Technological Development, Y. Xiang and S. Ali (eds) IGI (to appear 2009).

[24] Sauermann, L., Kiesel, M., Schumacher, K., & Bernardi, A. 2009. Semantic Desktop. Social Semantic Web 2009: 337-362

[25] Tso-Sutter, K., L. Marinho, et al. 2008. Tag-aware recommender systems by fusion of collaborative filtering algorithms. the 2008 ACM symposium on Applied computing, SAC '08, Fortaleza, Ceara, Brazil, ACM.

[26] Zhao, S., N. Du, et al. 2008). Improved recommendation based on collaborative tagging behaviors. the 13th international conference on Intelligent user interfaces, IUI '08, Gran Canaria, Spain, ACM.

[27] Ziegler, C.-N. and G. Lausen, 2004. Spreading activation models for trust propagation. IEEE International Conference on e-Technology, e-Commerce, and e-Service, IEEE Computer Society Press.

[28] Brin, S. and Page, L., 1998. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In: Seventh International World-Wide Web Conference (WWW 1998), April 14-18, 1998, Brisbane, Australia.

The Copied Item Injection Attack

Nathan Oostendorp
School of Information
University of Michigan
Ann Arbor, MI 48104
oostendo@umich.edu

Rahul Sami
School of Information
University of Michigan
Ann Arbor, MI 48104
rsami@umich.edu

ABSTRACT
In many web communities, users are assigned a reputation based on ratings on their past contributions, and this reputation in turn influences the recommendation level of their future contributions. In this type of system, there is potentially an incentive for authors to copy highly-rated content in order to boost their reputation and influence within the system. We describe this strategy as a copied-item injection attack. We conduct an empirical study of this attack on the online news discussion forum Slashdot. We find evidence of its use and demonstrate its effectiveness in eliciting high ratings. We explore variants of this attack in other domains and discuss potential countermeasures..

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**]: Information Filtering

General Terms
Reliability, Security.

Keywords
Manipulation, Recommender System, Online Discussion, User-Contributed Content

1. INTRODUCTION
Numerous online communities offer the ability to post and view user-contributed content, but participants can suffer from information overload in high-traffic environments. Often, rating and filtering systems are used to promote content that has been created or rated highly by leading users in the community. With these systems, an item's initial prominence is often based on the reputation of the content creator. This reputation is based (at least in part) on feedback on items that user has created in the past, and serves as a signal of quality as well as an incentive to improve quality. For example, reviews by 'Top Reviewers' on ePinions [5] and 'Elite Members' on Yelp [22] are prominently displayed, and comments by Slashdot [21] members with higher 'Karma' start at a higher level than other comments.

For these systems, as with other recommender systems, there is increasing concern about manipulation by users with a vested interest in promoting or burying certain target items. There is a growing literature on addressing the threat posed by attackers who create multiple shill or sybil accounts, and then use them to rate items in patterns (perhaps randomized) that will lead to collaborative filtering algorithms boosting or burying the target items. Defense techniques that have been developed include detecting and removing anomalous user profiles [2, 17, 12, 20,13], limiting the influence of user profiles until they have made contributions [19], and providing monetary incentives for honest rating [14, 1]. In this paper, we identify a new class of attacks that user-contributed content recommenders may be vulnerable to: the injection of duplicated or plagiarized items. We study the prevalence and effectiveness of this attack using a corpus of over 20 million comments from the technology news website Slashdot, and propose countermeasures against this attack.

Execution of a copied-item injection attack involves a two-step process for the attacker: First, she must find old items (comments, on Slashdot) that have been rated very highly by the community. She can then duplicate the entire item, or a portion of the item, and post this to the site as a new item (i.e., a new comment on a different story), claiming to be the creator. Site moderators do not always recognize this as a recycled item, and so rate it highly based on the quality of the original item. In turn, this leads to the reputation of the attacker being increased, as she is the purported author of high-quality content. Subsequently, she can exploit this higher reputation, and the improved visibility it brings, to attract attention to subsequent original (and possibly inferior) items she creates.

The first question raised by attacks of this form is: Are they harmful to the site or the rest of the community? This is not obvious, because in some contexts it may be a useful contribution to redirect the community's attention to valuable information that was known in the past, but has been forgotten. For any given domain, this will need to be weighed in comparison to the harm caused by the attack. In section 5, we argue that, for the Slashdot domain, the potential damage caused by this attack outweighs the potential benefit.

Existing techniques to prevent or limit manipulation in recommenders do not protect against copied-item injection attacks. This attack does not require the attacker to change her rating profile, so procedures that detect and filter anomalous ratings would not work. The influence-limiting approach also is not effective: Creating a good duplicate, or rating it highly, will be counted as a contribution by the attacker, but in this context, the attacker is merely reusing earlier information from raters on the original item to infer that the copy will be well-liked, but is not actually contributing new information. Existing mechanisms that prescribe monetary incentives to rate honestly do not address this problem, as the raters who rate the copies highly are being honest about their perceptions of its quality. In section 6 we discuss some techniques that could be used to combat copied-item injection attacks.

The rest of this paper is structured as follows. In section 2, we review the related literature. In section 3, we formalize our definition of copied-item injection attacks. Section 4 describes our empirical analysis of this attack on the Slashdot dataset, and our measurements of the current prevalence and effectiveness of this attack in the Slashdot domain; we discuss the consequences of these results in Section 5. In section 6, we discuss countermeasures against this threat. We conclude and identify directions for future work in section 7.

2. RELATED WORK

Recently, there has been a rich literature centering on the vulnerability of collaborative filtering recommender systems to attack, as well as defenses against those attacks. This was initially observed by Lam and Riedl [8] and O'Mahony et al. [16]. This literature has focused on a particular class of threats: attackers can create "shill" or "sybil" user profiles, and use these to promote or bury items they have a vested interest in. A number of authors have studied variants of this attack, as well as defenses against them; we refer readers to recent surveys by Mobasher et al [15] and Mehta and Nejdl [13]. Techniques to defend against this attack include methods to detect and remove anomalous user profiles [2,17, 12, 20, 13], limiting the influence of user profiles until they have made contributions [19], and providing monetary incentives for honest rating [14, 1]. The chief difference with our current work is that we consider a different class of attack: we study settings in which, in addition to potentially injecting shill user profiles, the attacker can inject items with known quality (derived by copying existing items).

There has also been prior research on the Slashdot moderation system. Lampe and Resnick [11] analyze the performance of the moderation system in identifying high-quality comments, and show that it is largely effective. Lampe and Johnston [9] report that new users of the site use the moderation feedback they receive as a cue to learn the norms of the community. Lampe et al. [10] propose to use a second level of collaborative filtering to adapt users' interface views of the moderated comments. Poor [18] argues that Slashdot is an archetypical public sphere on the Internet, and describes the role of the Slashdot moderation system in fulfilling this function.

David and Pinch [3] conducted a qualitative study of strategic reviewing on Amazon.com. They document several cases of plagiarized reviews; one of the motives they identify for plagiarizing is to build up a long profile of ratings with low effort. This is similar to the modus operandi of our copied-item injection attack, except that the community's ratings on the content are more important than the raw number of comments in our setting.

3. MODEL AND TERMINOLOGY

In this section, we introduce terminology to clarify our discussion of the copied-item injection attack.

There is a set U of users; we use h to denote an honest contributor, and a to denote the attacker. A set of items I; each item $i \in I$ has two characteristic features: *creator*(i) denotes the user who is listed as the creator of the item, and *content*(i) is a description of its content (text, image features, etc.). The attacker has some target content T that she would like to promote. At any point in time, an item has a recommendation level *rec*(i). For simplicity, we assume that the recommendation level is not personalized; for personalized recommenders, *rec*(i) could denote the average recommendation level among the target community, or another summary statistic.

Each user u has a reputation $R(u)$. Item recommendation level *rec*(i) is assumed to depend on its creator's current reputation R(*creator*(i)) as well as the corpus of ratings on the item set I. The user reputation $R(u)$ is assumed to be computed based on the corpus of ratings; we assume that, other things being equal, $R(u)$ is higher if a particular item i with *creator*(i)=u has higher recommendation level *rec*(i). We assume that two items i,j with *content*(i)=*content*(j) have positively correlated recommendation levels, because the raters cannot consistently identify the later item as having duplicated content. This is realistic in a system with a large number of items and users.

A **copied-item injection attack** involves the attacker copying a genuine item i, with a high *rec*(i), to create a new item c, with *content*(c)=*content*(i), but *creator*(c)=a while *creator*(i)=h. The attacker then waits for c to collect a sufficient number of ratings, so that *rec*(c) increases towards the high level of *rec*(i). Finally, attacker a creates a new item t with *creator*(t)=a and *content*(t)=T.

The simplest measure of the success of an attack is the difference between *rec*(t) after this attack then it would have been if item c was not created. A slightly more nuanced measure, which is natural is the context of analyzing a's incentives, is the increase in a's *net benefit*, accounting for the cost of creating the copy c and the opportunity cost of not creating an original posting instead. We explore this idea further in section 6.

4. ANALYSIS OF SLASHDOT

Slashdot is a high traffic online news site and an active forum that receives several thousand user-contributed comments and over a million pageviews every day [20]. To help the users navigate among the large amount of user-contributed material, it uses a rating/moderation system that lets them filter comments based on a score from -1 to 5. This system has elaborate controls to detect and discourage abuse, including rules on who can moderate, how often they can moderate, and how much they influence the score[11].

4.1 Slashdot's Moderation System

The system revolves around two scores assigned to user accounts: **karma**, which is accrued by contributing comments and receiving positive moderations on those comments, and **mod points**, which allows users to rate other users comments up or down. In this system, the users and items are linked by authorship, so that each item's rating is aggregated into karma for the user. A user's karma then determines both the probability of acquiring mod points and the starting score for their posted comments. Because positive ratings on an authors comments gives the author additional influence within the system, there is clear incentive to manipulate the system if a users goal is to gain influence or prominence in these discussions.

Additionally, users can **meta-moderate** and judge whether a users mod points have been spent appropriately. In this process, users can view comment moderation pairs and give a up/down feedback on if each moderation was appropriate. Users who frequently are evaluated as having rated inappropriately become less likely to receive mod points. This was designed to defend against simple manipulations where mod points were traded or spent on inferior comments for the express purpose of improving another users karma.

While this system has some algorithmic checks for basic profile-injection strategies such as detection of high-traffic cyclical moderation patterns between users, there are some manipulation strategies that can be used to gain undue influence within the system. The online comic WellingtonGrey has humorously documented a few of these in flowchart form [6]. This chart identifies tactics for accruing karma including profile-injection ("a second account with mod points"), strategically expressing popular sentiments in comment text ("Is it about Microsoft? Say they suck. Is it about Apple? Say they rule."). It also advises recycling of old material. ("Do you have any old +5 posts on this topic? Quick, post one!") This third tactic describes copying an item to gain positive ratings, and therefore karma.

The Slashdot environment is likely to be an ideal environment for this type of attack, due to several factors. Its longevity as a news source (it celebrated its 10th Anniversary in 2008), and high volume of traffic gives it a large library of existing comments that could be recycled. Since so many comments are posted every day, it is also reasonable to assume readers will be unable to recognize an older comment out of the millions authored on the site. Additionally, the nature of "news cycles" means that certain topics recur frequently: a subject line search shows that Slashdot has over 200 stories on Windows Vista, which has been in the news for 2-3 years.

Based on these factors we can make a few generalizations about where a copied-item attack might be used. Certainly it must have an environment where the cost of item creation is low and also the cost of copying an item is similarly low. The incentive to use the attack must come from when the author receives some indirect benefit from positive ratings on the items they create. The Copied Item attack will also be easier where there are extremely large numbers of items so that the probability of duplication detection by recognition from readers is low. Finally, it will be easier to deploy the attack when items have simple data structures, such as a comment with a block of text, a subject line, and an authorship reference, as opposed to items that might be indexed on many different attributes and therefore may have too many similar attributes to the original.

4.2 Description of Slashdot Data

We used a snapshot of Slashdot's database from January 28, 2009, which contained 20,830,313 comments contributed by 307,158 users across 158,867 news story discussions. Each comment record contained a short subject line, a longer message body, a timestamp of publication, the final rating for the comment, and numerical ids referencing for the story and author.

The rating distribution for comments, shown in Figure 1, is roughly a right-skewed normal distribution centered on the mean of 1.158 with a standard deviation of 1.149. 1.30 million comments have a rating of 4 or 5, or about 6.2% of the entire population.

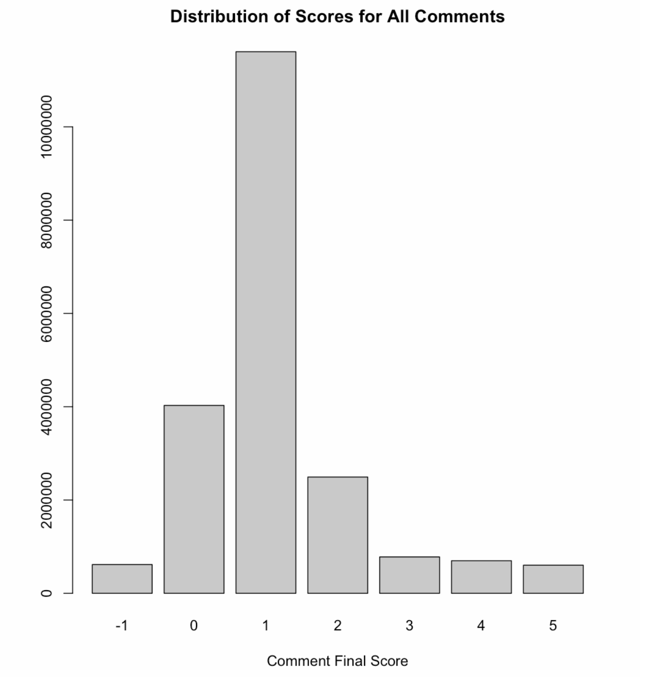


Figure 1: Score Distribution for All Comments on Slashdot

The comment text length distribution is shown in Figure 2 and follows a lognormal distribution. After a logarithmic transformation, the mean comment length is 5.68 (293 characters) with a standard deviation of 1.11. The entire body of text from all of these comments is roughly 11.0 billion characters.

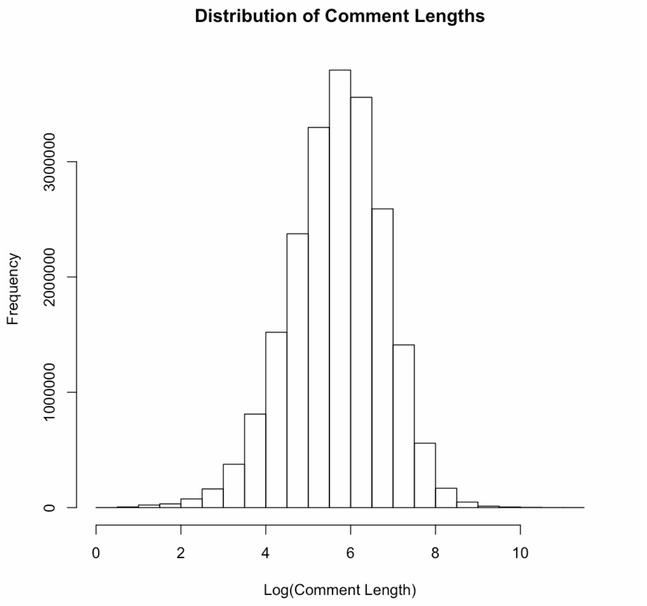


Figure 2: Histogram of Log-Transformed Comment Lengths

4.3 Detection of Copied Items

In this study, our goal was to detect plagiarized comments in this large Slashdot comment corpus. The core of this process was finding comments that shared large substrings. However, there are several conflating factors which could legitimately lead non-attackers to reuse large substrings within their comments: users quote from earlier comments or quote the same source; there is a form of political activism that involves posting the same text repeatedly such as the DeCSS decryption codes; and some users attempt to disrupt a forum by posting as many junk comments as possible. We processed the comments conservatively, so that we would identify a comment as plagiarized only if none of the conflating factors is a plausible explanation for the duplicated text.

In order to detect plagiarisms our first step was to detect comments that had significant duplicate text. We implemented a Rabin-Karp search [7] with a window of 255 characters. Using this method we converted each 255-character substring of a comment message body into a hash value, and searched for co-occurrences of hash values across multiple comments. The entire corpus generated about 6.4 billion (hash,comment_id) pairs. Any comment found to have more than 3 hash collisions with any single previously posted comment was logged. We then went through the logged comment pairs and confirmed that there was significant duplicated text using a longest common substring algorithm. This process resulted in 196,349 pairs of potentially plagiarized comments among the 20-million comment corpus.

In order to narrow this set of comment pairs to distinguish comments that may have been directly plagiarized with intent to boost ratings, we applied a sequence of filtering steps to the original set of copied items. These included:

1. We removed any pairs where the original comment had a final rating score of 3 or less. This was eliminate comment copies that had little reason to expect a high rating.
2. We removed any pairs where the longest common substring was less than 90% of the copied comment length. This was to avoid comments that had significant original material as well as copied content.
3. We eliminated comment pairs where the copied comment did not begin with the longest common substring. This rule was used to weed out quotations since attributions or quotation marks would typically prefix a quote.
4. We removed any comment pairs that appeared in the same story. This was to avoid implicit quoting within replies.
5. We eliminated comment pairs where the copied comment was posted anonymously, rather than by a logged in user, as anonymous users see no direct benefit from having their post rated highly.
6. We eliminated comment pairs where the original comment was copied more than once; this was used to control for overt reposting, DeCSS code posts, or other forms of habitual reposting.

With these conservative restrictions in place, the set of probable plagiarisms was 735 comment pairs, where 423 users had posted the copied comments. We visually inspected about two dozen pairs manually to confirm that there was no other apparent reason for duplication.

4.4 Hypotheses and Results

Intuitively, we expect that copies of highly-rated comments will also garner high ratings and be useful to potential attackers for the purpose of acquiring karma. In this section we formulate three hypotheses that test this conjecture.

Hypothesis 1: Copying a comment with a high rating is profitable for attackers, in that it produces a comment which is more likely on average to be highly rated.

If the copying of comments were profitable for an attacker, we would expect the copies of these high scoring comments to garner higher ratings than the population at large. We found in the target population of likely plagiarized comments the rating distribution of the copied comments was substantially changed versus the distribution of the global population, as illustrated in Figure 3. Indeed population of copied comments had a mean of 2.15 vs the global mean of 1.16, nearly a full standard deviation higher than the global mean, a difference of 0.987 points. Additionally, 30.4% of items in the copied set had a rating of 4 or 5 as opposed to 6.2% of the global comment population. A two-sample t-test confirmed significance of both results ($p < 0.001$). This discrepancy confirms Hypothesis 1.

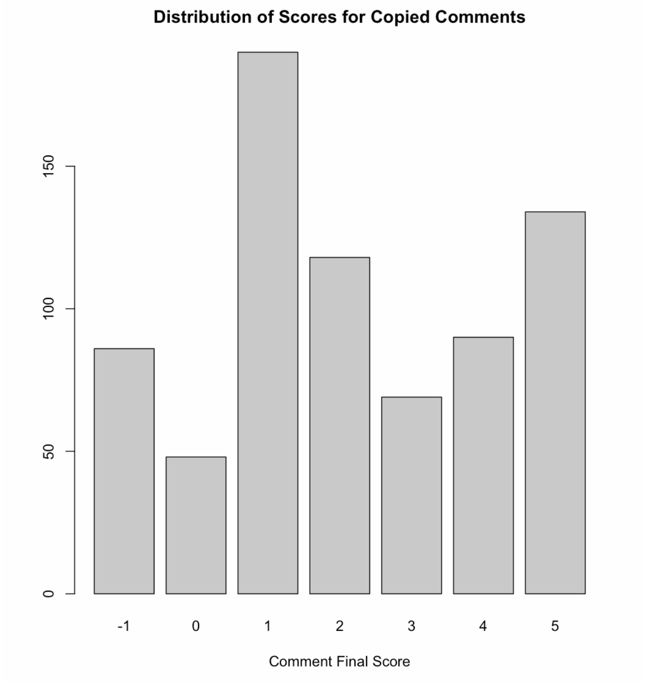


Figure 3: Distribution of Scores for Copied Comments

Hypothesis 2: Copying a comment with a high rating is more profitable than contribution of other content by the attacker.

To see if this strategy is incentive compatible for the attacker, we looked at our set of copied comments compared with the mean rating for the copied item authors other items. By comparing each of the copied comments scores with the users mean post rating in a pair-wise t-test, we found the copied item had a mean improvement of 0.730 points ($p < 0.001$). This confirms Hypothesis 2.

Hypotheses 1 and 2 confirm that copies of highly-rated comments tend to be rated highly even when taken out of their original context. It is conceivable that these comments add value to the readers of multiple topics, and that little damage is done by rewarding the copiers for reposting them. We will discuss the harm caused by the copied-item attack in more detail in section 5.1. Here, we provide evidence that the copies damage the signaling quality of the Slashdot rating system:

Hypothesis 3: The average rating of comments, other than the copied comment, by the copier is lower than the average rating of other comments by the original poster.

In order to test this hypothesis, we first excluded all instances in which the original comment was posted by an anonymous user. (If the original comment was posted by an anonymous user, we could not identify other comments posted by the same user; further, it is clear to the readers that a comment is anonymous, and hence it is unlikely that they would improve their expectation of other anonymous comments). For each of the 683 surviving instances, we measured the average rating of all comments (other than the copied comment) posted by the original poster, and the average rating of all comments (other than the copied comment) posted by the copier. We find that the average rating for the original poster is 1.70, vs 1.38 for the copiers; a two-sample t-test confirms significance (p <0.001). This suggests that the copiers actually had lower quality than the original posters, and thus, the high rating they receive for the copied content reduces the ability of readers to distinguish them from the higher-quality posters who posted the original comments.

Hypothesis 4: Copied comments are much more likely to be topic starters (comments starting a discussion thread) than other comments, since it would be more difficult to have a copied response seem appropriate as a reply to multiple comments.

We looked at the location of our copied comment population in Slashdot discussions and found that of the 734 copied comments 573 were topic starters. If you contrast this with the entire comment population of 20.8 million, 6.28 million comments started topics. A two-sample t-test indicates that the copied comments are 47.8% (78.0% vs 30.2%) more likely to be topic starting than a comment in general (p < 0.001). Hypothesis 3 is therefore confirmed. The consequence of this hypothesis is that copying can distort the pattern of interaction on the site, skewing it towards breadth rather than depth of exchange.

5. DISCUSSION

With H1, H2, and H3 confirmed, it seems evident that item copying has been successfully used on Slashdot to systematically garner high ratings for comments and therefore improve the users karma score. We expect that this type of item injection attack has potential to be a widespread problem both in the realm of Slashdot and other moderation-based comment systems as well as other collaborative filtering spaces. In any forum where inserting copies of highly rated content is incentive compatible and technically possible there is a strong likelihood of abuse. At the core of this incentive problem on Slashdot is the transitive property of item scores to users, where a user stands to directly gain influence in the system by receiving positive feedback on their items. However, systems containing low-cost item creation may present different incentives for this type of attack, and it may create variations in overall impact.

Simple manipulations to try and disrupt this type of behavior may add only marginal costs to the effort required to copy comments. On March 20, 2001 Slashdot deployed a code update that attempted to curtail comment "re-posting" by logging an MD5 hash encoding of the entire comment text. Subsequent comments that were posted with the same MD5 sum as a previous comment were rejected from the discussion. We looked at our copied comments sample set and found 28 comments posted before this feature was deployed, 26 of which were exact copies. After this change it was not possible to post the *identical* comment again; however, it was possible to make a trivial change to a comment, such as addition of whitespace, and repost. Of the 707 copies detected dated after the March 20, 2001, 618 were identical to the original except for the insertion or deletion of punctuation and/or whitespace. After controlling for whitespace and non-alphanumeric characters we found no significant difference between entire/partial match ratio between the two populations using a binomial test.

We suspect that this may possibly be due to the extreme ease with which a duplicated post could be altered by adding even a single whitespace character anywhere in the text. It also may be that our conservative heuristics used to detect likely plagiarisms select primarily towards exact matches in this data set.

5.1 Is Slashdot comment copying really harmful behavior?

From a certain perspective, it may be reasonable to point out that the copied comments on Slashdot do add value to the system. In a sense, the positive ratings that the duplicated comments receive are signals from the raters that the comment has value, and this may add insights that otherwise wouldn't be seen in this discussion environment. While it may take a certain moral flexibility to ignore the taboo of plagiarism, the copied item posters could be thought of as agents of conversational arbitrage, seeking out and shining up old gems from previous discussions. However, simply looking at the reposted comments as harmless injections ignores other externalities of having unattributed reposting in a discussion system. Although the user ratings reflect the immediate visceral reaction of the raters to the content, this may not capture the entire value of a piece of content to the system.

For the Slashdot domain, we believe that the potential damage outweighs the potential benefit: Users can always jog the community's memory by quoting earlier comments with attribution instead of resorting to plagiarizing comments, and quoting is fairly widespread, so there is little additional benefit accrued through these attacks. In fact, given the availability of quoting as an alternative which meets community norms and requires negligible additional effort by the copier, the fact that a user would choose to not credit the original author is illuminating: it indicates that they expect to gain a better reception (and better ratings) by suppressing the fact that the content was duplicated. This in itself suggests that the ratings are not perfectly aligned with the community's perception of the long-term value of a contribution.

It is likely any systemic method to gain karma would have an undesirable effect on the Slashdot system, and become increasingly widespread if the technique was communicated between users. One problem is this tactic distorts karma as a signal of someone who has contributed good fresh content. For instance, in the Slashdot system, karma has a direct impact on the starting score of a users post. Therefore a user with high karma user may start their post at 2, rather than 0 or 1. This means that the comment ratings lose their effectiveness as a signal of quality as well in this particular situation. This loss of signaling quality was borne out in our confirmation of hypothesis 3.

The other potential impact if this tactic of copying comments was widespread is that it would have a negative impact on the dynamic actual conversations that occur within Slashdot. Hypothesis 4 confirms that these comments tend to be discussion-topic starters, but any replies to these copied comments would be very likely to be disregarded by the attacker. They are, after all talking to a different person than the user who originally generated the comment text. This means in as copied comments became more frequent within the system, the harder it would be for users to find genuinely interactive experiences.

Ultimately, we believe the threat is significant enough that defenses against it merit careful consideration. This phenomenon potentially weakens both incentive and signaling function of the site's reputation system: Users may be incentivized to copy items as a lower-cost way of building reputation than creating original content, even though the latter is a more valuable contribution; and, future original contributions by the attacker may start as a misleadingly high recommendation level, because they reflect the quality of the author of the original item, rather than the attacker's inherent quality. Additionally, it may create an incentive for copying content without attributing the original author, which can disrupt the norms of the online community.

5.2 Variants in other domains

It is possible that a copied item injection attack could potentially appear in other types of recommender spaces where items can be inserted into the system with relatively minor barriers, just as profile injection attacks are potentially problematic in spaces where a user creation in a system is extremely low-cost. In particular, any systems where ratings on items transitively score the users who create the items will provide incentive for this type of attack.

Although the Slashdot recommender system uses a simple voting method of collaborative filtering, it is sophisticated in tracking reputations for users and using these reputations to allocate visibility and influence. Reputation tracking is a powerful method of identifying high-quality contributors over time, so we expect that many recommenders for social web applications will adopt some it in some form. Then, copied-item injection attacks, perhaps in conjunction with other attacks, will become a potential threat.

In particular, it is the combination of an item and profile attack that could be extremely problematic. A sophisticated attacker could use the copied items to establish validity for shill items posted by shill accounts, and likewise rate other comments similarly with shill accounts. This would potentially create a system where scores could be quickly increased on both shill users and items.

In a movie recommender system (or other traditional item recommenders) a combination of an item and user injection could potentially distort recommender predictions if site maintainers were not vigilant about repairing duplication. A copied item, whether legitimately cataloged as a variant of an original film (ie a "directors cut") or sorted under a different name, could be used as a target item in a manipulative attack in order to "push" or "nuke" according to an agents agenda.

Another application in which copying items can increase the power of an attacker is in search engine website rankings. Here, the `ratings' are expressed in the form of other sites linking to a particular site. By copying some content from a high-quality site, an unscrupulous site operator can increase the chances of other genuine sites linking to his site. This will drive up the ranking of his site on search engine results pages; some of these pages can be used to damage readers through unrelated advertisements or fraudulent content.

There are several other domains that could potentially see item-injection attacks. In the news website space, gaming a collaboratively filtered news aggregator such as Digg [3] could be profitable by increasing traffic and therefore ad revenue.

6. POTENTIAL COUNTERMEASURES

In this section, we describe a framework for reasoning about countermeasures to the copied-item injection attack, and identify several possible techniques that could effectively combat this threat. There are two core factors behind the copied-item attack: (1) Users have an incentive to increase their reputation, and incur effort costs when they attempt to do so, either by copying items or by creating fresh contributions. (2) Copied items are likely to garner ratings that are similar to those of the original item. We frame our discussion of countermeasures with these two aspects of the problem in mind.

For a given domain, it is helpful to visualize a space *A* of possible pieces of content, coupled with a distance metric that captures the similarity between two pieces of content: The smaller the distance between *x* and *y*, the more similar the pieces of content. For example, *A* could be the space of all text strings, and the distance measure could be based on edit distance, or keyword frequencies. For a movie domain, *A* could be defined by a set of features (title, actors, director, etc.), with a distance metric based on this feature similarity. Modeling the content space in this way allows us to reason about near-copies as well as exact copies. The cost and benefit to an attacker *a* in executing an item-copy injection attack can then be described in terms of this reference model. When *a* copies an item *i* to generate a near-copy item *c*, her cost is presumably increasing in the distance between *content*(*i*) and *content*(*c*), reflecting the effort of obfuscating the fact that the item was copied; for example, it takes some effort to reword a comment or change the whitespace and punctuation. The benefit accruing to the attacker depends on the ratings that *c* garners; given that *i* was a very highly-rated item, the benefit might be highest for an exact copy but drop off as the distance between *content*(*i*) and *content*(*c*) increases.

Techniques to combat item-copy injection attacks can work by raising the cost of carrying out the attack, imposing a penalty if the attack is detected, or reducing the benefit of creating the copy item *c*.

- One natural technique is to detect copies, and either prohibit them outright, or impose a reputation penalty when they are injected. This is the approach that Slashdot implemented when they prohibited exact copies of comments. In practice, however, this imposes an insignificant cost on attackers, as they only have to make trivial changes to a previous comment. Instead of merely identifying exact copies, a slightly more sophisticated approach might detect an item within a certain distance of a pre-existing piece of content, using a distance metric appropriate for the domain. This has a two-fold advantage: it forces attackers to put in more effort in modifying the original content, and in doing so, the copy is less similar to the original item, leading to a lower expected benefit. Another variation would be to not prohibit near copies, but rather, to merge similar items into a single logical 'item-cluster'.

There are two drawbacks to this approach, however. First, it is only as good as the distance metric used. This might spark an arms race between attackers and site managers, in which attackers continually find clever ways to retain the quality of the original item while appearing to be distant under the current metric, and site managers continuously update the metrics to plug these gaps. Second, as the distance threshold increases, there is a growing threat of false positives: genuine items that get mistaken for copies. This could hamper the contribution of honest users.

- Alternatively, the defense can focus on reducing the benefit to users of copying items, relative to more socially valuable activities such as the creation of original content. The attacker derives benefit because of the increase in her reputation and the privileges that accompany a better reputation. This suggests that a more sophisticated reputation update may be effective: When a user *a* creates an item *i*, rather than increase her reputation based merely on the average rating of *i*, we should account for the average rating of similar items as well. For example, the creator's contribution might be calculated as the difference between the average rating of item *i* and the average rating of the nearest (in terms of content distance) pre-existing item *j*; or, perhaps, use a similarity-weighted average of all pre-existing items. This reduces the benefit of copying high-quality items, hopefully to the point that users choose more valuable ways of building their reputation. Genuine posting of similar items would still be possible, but there would be a reduced incentive to do so.

  The same approach can be extended to tailor the incentives of *raters* as well as creators. The Influence Limiter [18] scores raters based on the amount they improve predictions for future raters. Loosely, a rater who is the first to rate a high-quality item high will gain the highest score, while subsequent raters will be measured as having diminishing contributions. A rater's accumulated score is then used to limit their influence on others' predictions. In the case of a profile injection attack, the effectiveness of each shill is stunted – as it adds no information, it will not earn a high reputation score, and hence have limited influence. As described in [18], the Influence Limiter might be susceptible to copied-item injection attacks: The attacker expects the copy *c* to have similar ratings to the original *i*, and thus, attacker shills can be the first to put in high ratings where relevant. This can be countered by scoring the early raters on items relative to a benchmark prediction that is the average of pre-existing items with similar content.

- A third technique might be to rely on targeted moderation that flags items as 'legitimate' or 'plagiarized'. Human moderators could be shown nearest content items, and might be more skilled at distinguishing genuine forms of copying from reputation-boosting plagiarism. The tradeoff, of course, is that this requires additional human effort that might be better spent in creating or rating items. In addition, as with rating systems, there would need to be a system to prevent attacker shills from controlling this moderation process, perhaps necessitating a level of "meta-moderation" as well.

One constraint on all of these techniques is that calculating distances between pieces of content in a large database can be very computationally intensive. This might preclude the use of these techniques in a online mode. Instead, the automated techniques could be used offline to periodically filter items or adjust reputations. Human moderators trying to locate similar pieces of content online would have to rely on simple distance metrics.

It is not possible to meaningfully evaluate the performance of these techniques on our existing dataset, as the attackers are likely to adapt the detailed form of attack once a specific countermeasure has been deployed. This is borne out by the way in which users sidestepped Slashdot's check for identical comments, as described in section 5. The evaluation of the relative effectiveness of these countermeasures is therefore left as a subject for future work.

7. FUTURE WORK

In this paper, we have identified a class of attacks, copied-item injection attacks, that user-generated content recommenders on the web may be vulnerable to. We have studied this attack in a single domain, but the attack pattern is relevant to many different settings; likewise, countermeasures developed in one setting will be helpful in others as well. There are several important directions for future work. The development and implementation of practical countermeasures should be a priority for applications where the copied item injection attack is a feasible strategy. For some domains where duplicate detection of content is impractical, one direction of research may be to use patterns of user ratings to identify similarity between items.

Additionally, it would be useful to conduct empirical or experimental measurement of the prevalence of this attack in other domains. This would give confirmation as well as a broader understanding of attack patterns and the motivations of attackers.

Once countermeasures have been implemented and deployed, and users have had a chance to adapt to them, it will be important to experimentally determine their effectiveness by comparing the frequency and impact of attacks with and without defenses.

8. ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under award IIS-0812042. We would also like to thank the Slashdot Engineering team at SourceForge Inc, specifically Rob Malda, Jamie McCarthy, and Uriah Welcome for their help in accessing and interpreting Slashdot comment data. We are also grateful to Paul Resnick at the University of Michigan for his helpful feedback and suggestions on this project.

9. REFERENCES

[1] R. Bhattacharjee and A. Goel. Algorithms and incentives for robust ranking. In Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA '07), 2007.

[2] P.-A. Chirita, W. Nejdl, and C. Zamfir. Preventing shilling attacks in online recommender systems. In WIDM 05, pages 67–74, 2005.

[3] S. David and T. Pinch. Six degrees of reputation: The use and abuse of online review and recommendation systems. First Monday, 6, 2006.

[4] Digg, 2009. http://www.digg.com.

[5] Epinions, 2009. http://www.epinions.com.

[6] W. Grey. The slashdot flowchart, 2007. *http://miscellanea.wellingtongrey.net/2007/04/28/slashdotflowchart/*

[7] R. M. Karp and M. Rabin. Efficient randomized pattern-matching algorithms. IBM J. Res. Dev., 31(2):249–260, 1987.

[8] S. K. Lam and J. Riedl. Shilling recommender systems for fun and profit. In Proceedings of WWW '04., pages 393–402, 2004.

[9] C. Lampe and E. Johnston. Follow the (slash) dot: effects of feedback on new members in an online community. In Proceedings of the 2005 international ACM SIGGROUP conference on supporting group work, 2005.

[10] C. Lampe, E. Johnston, and P. Resnick. Follow the reader: Filtering comments on slashdot. In Proceedings of CHI 07 Conference on Human Factors in Computing Systems, pages 1253–1262, 2007.

[11] C. Lampe and P. Resnick. Slash(dot) and burn: Distributed moderation in a large online conversation space. In Proceedings of ACM CHI 2004 Conference on Human Factors in Computing Systems, 2004.

[12] B. Mehta, T. Hoffman, and P. Fankhauser. Lies and propaganda:detecting spam users in collaborative filtering. In Proceedings of IUI'07, 2007.

[13] B. Mehta and W. Nejdl. Attack resistant collaborative filtering. In Proceedings of ACM SIGIR '08, 2008.

[14] N. Miller, P. Resnick, and R. Zeckhauser. Eliciting honest feedback: The peer-prediction method. Management Science, 51(9):1359–1373, 2005.

[15] 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, 7(2):1–40, 2007.

[16] M. O'Mahony, N. Hurley, and G. Silvestre. Promoting recommendations: An attack on collaborative filtering. In Proceedings of the 13th International Conference on Database and Expert System Applications, pages 494–503. Springer-Verlag, 2002.

[17] M. P. O'Mahony, N. J. Hurley, and G. C. M. Silvestre. Detecting noise in recommender system databases. In Proceedings of the 2006 International Conference on Intelligent User Interfaces, pages 109–115, 2006.

[18] N. Poor. Mechanisms of an online public sphere: The website slashdot. Journal of Computer-Mediated Communication, 10(2), 2005.

[19] P. Resnick and R. Sami. The influence limiter: Provably manipulation-resistant recommender systems. In Proceedings of the ACM Recommender Systems Conference (RecSys07), 2007.

[20] 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, 2007.

[21] Slashdot, 2009. http://www.slashdot.com.

[22] Yelp, 2009. http://www.yelp.com.

Does Trust Influence Information Similarity?

Danielle H. Lee & Peter Brusilovsky
School of Information Sciences
University of Pittsburgh
135 N. Bellefield Ave. Pittsburgh, PA, USA
+1-412-624-9437
{hyl12, peterb}@pitt.edu

ABSTRACT

In collaborative filtering recommender systems, users cannot get involved in the choice of their peer group. It leaves users defenseless against various spamming or “shilling” attacks. Other social Web-based systems, however, allow users to self-select trustworthy peers and build a network of trust. We argue that users self-defined networks of trust could be valuable to increase the quality of recommendation in CF systems. To prove the feasibility of this idea we examined how similar are interests of users connected by a self-defined relationship in a social Web system, *CiteuLike*. Interest similarity was measured by similarity of items and meta-data they share. Our study shows that users connected by a network of trust exhibit significantly higher similarity on items and meta-data than non-connected users. This similarity is highest for directly connected users and decreases with the increase of distance between users.

Categories and Subject Descriptors

H.1.2 [**User/Machine Systems**]: Human Factors; Software Psychology; J.4 [**Social and Behavioral Sciences**]: Sociology

General Terms

Measurement, Human Factors

Keywords

User Similarity, Trust, Human Network

1. INTRODUCTION

Recommender systems powered by collaborative filtering (CF) technologies become a feature of our life. Such popular systems as Amazon.com, Netflix, Last.fm, and Google News use Collaborative filtering to recommend us products to buy, movies to watch, music to listen and news to read. The power of this technology is based on a relatively simple idea: starting with a target user’s rating, find a peer cohort (neighborhood) of users who have similar interests and recommend items favored by this cohort to the target user. As such, the choice of cohort is an essential part in CF recommendations and is usually determined by automatically calculating rating similarities between the target user and other users. In a typical CF system, this peer cohort (a group of users selected as the basis for CF) is unknown to target users. Moreover, the target users cannot add trustworthy users to their cohort group nor exclude suspicious users from the group.

The success of social linking and bookmaking systems that allow users to build their networks of trust, stresses a fact forgotten by modern CF systems: the source of the recommendation is an important criterion for judging the quality of recommendations [2]. A range of Web 2.0 systems such as *LlinkedIn*, *Flickr*, *Delicious*, *CiteuLike* etc., provide various kind of social linking, enabling their user to pick known and trusted users and add them to their list of connections. These self-defined links between users establish a rich network of trust, which is, in turn, used to propagate various kinds of information. Given that, it is natural to expect some kind of merger between social linking and CF technology: a new generation of *trust-based recommender systems*, which will use self-defined social networks of trust to improve the quality of CF systems and the satisfaction of their users. Some pioneer works in this direction already appeared [4, 5, 6, 9, 13]

To prove that trust-based recommenders are more than a speculation, some important assumptions have to be checked. Is it true that connected users in the networks of trust share not only trust, but also some common interests? Is it true that information can flow along these networks, i.e., the choices made by users are affected by the choices of users they trust? The goal of this paper is to test these assumptions. Using real life data collected from a social Web system, *CiteuLike*, we examined several important properties of self-defined trust networks. We investigated how similar are users’ interests in these networks, the extent to which amount of similar information collected by users depends of the strength of their connection, and ultimately, how feasible it may be to use a network of trust for personalized recommendation.

The term ‘trust’ as used in this paper may not be an exact match with the general use of ‘trust’ as defined in the sociology. The social relationship used in this paper is defined unilaterally, simply indicating user trust in the usefulness of information provided by connected individual. It is not trust through personal interaction or emotional support (for instance, connected with an expectation of obligation, morality or responsibility [7]). Since referred users are deemed “trustworthy” by the target user in terms of information collection, however, the term ‘trust’ was selected. Furthermore, the term ‘trust’ as defined in the Webster’s Third New International Dictionary meets our interpretation of ‘trust.’ Its definitions for the term are “a confident dependence on the character, ability, strength, or truth of someone or something,” “confident anticipation,” and “a charge or duty imposed in faith and confidence or as a condition of some relationship” [7]. To date, a better or more precise term for this relationship has not been found; hence, *trust* is used hereafter.

2. RELATED WORK

The popularity of CF technology, revealed some problems. CF appeared to be not well-protected against malicious users who try to harm the system or to make a profit by gamming the system. For example, by copying the whole user profile, a malicious user is perceived by the system to be a perfect peer user and the products added by him are therefore recommended to the target user [3, 5, 8]. Even without malicious users the quality of recommendation can be affected by peculiar users with unusual interests [10]. Moreover, since CF systems have to compare all other users in order to find the peer group, the computation requires substantial off-line process [4]. Finally, users who do not have sufficient ratings are not able to receive reliable recommendations [10]. These CF-related problems occur in part because the recommender systems make a choice of peer group purely by similarity computation, and do not allow the target users to affect this part of the recommendation process.

Several research teams attempted to exploit trust between users to resolve some of the cited problems of CF technology. Massa and Avesani's study [4] showed that a user's trust network can solve the ad-hoc user problem, improve recommendation prediction and attenuate the computational complexity. Another study indicated that a trusted network decreases the recommendation error and increases the accuracy as well [9]. For users with a unique taste, their own trusted network could increase the satisfaction of recommendation, since they are able to know where the information came from [12]. The recommendations made by friends were known to be frequently better and more useful than the recommendation made by systems [11].

To prove the feasibility of trust network as a source of information for reliable recommendation, several research teams started with checking the main assumption: do users linked by self-defined networks of trust have similar interests.

Singla and Richardson (2008) found the positive correlation of frequency and time of instant messaging between users with search interests [11]. Another trust-related research suggested that two users who are friends tend to share similar vocabularies, in-links and out-links on their personal homepages [1]. Ziegler and Golbeck [13] compared interest similarity between people in a trusted network. They used information regarding users and the user's trust ratings in the book recommendation. Rather than using each information item, they grouped the items by topics, using an existing taxonomy. Then, they built topic-based user profiles and the closeness of the user profiles in the trusted network was assessed. As the conclusion, they found that topic-based user profiles became more similar as the trust values between two users increased and reduced the data sparsity problem existing on the comparison of individual item [13]. Our work presented below was motivated the same goal: to assess interest similarity between users connected by relationship of trust.

3. DATA COLLECTION

3.1 Data Sets

As a source of data for our study we selected *CiteuLike*, a social Web system for sharing bibliographic references. To pick up initial set of users, we visited this site randomly in September and October of 2008. Users who posted new articles at the time of visit were picked. The information collected for each user included the bibliography (article title, list of authors, journal name, publication year, etc.) and the *watchlists* (connected users). After collecting a group of initial users, we collected data of their trusted connections. Table 1 shows the descriptive statistics.

In collaborative tagging systems explicit connections between users are of special nature. In some sense, they bear more "trust" than the connections between friends in social networking systems. In *CiteuLike*, users can directly connect to other users who have interesting bibliography by adding them '*watch list*.' Then the system displays the whole bibliography of watched users.

Table 1. Data Summary of *CiteuLike*

Total no. of users	21076
Total no. of distinct items (papers)	449824
Average no. of items per user	28.69
Total no. of unidirectional relation	11295
Total no. of reciprocal relation	93

3.2 The Networks of Trust

In this paper, we interpret user's act of connecting to other users (by adding this user to the watch list) as a sign that she likes the focus and trust the quality of the added user's references and wants to have direct access to them continuously in future. Thus, watching in *CiteuLike* could be considered as evidence that connected users are trustworthy to the original user in terms of information collection.

We distinguish two kinds of trusted connections – unidirectional and reciprocal. The act of adding another user to the '*watch list*' is unidirectional (which is different from social networking systems). If user A added user B to her network, it does not imply that user B will be added to A's network necessarily. The users in A's network decide independently whether to add A to their networks. For example, user B may not have A in his network and we call the relationship between A and B as 'unidirectional'. Another user C in A's network may add A to his network as well. We call this relationship as 'reciprocal' (Figure 1).

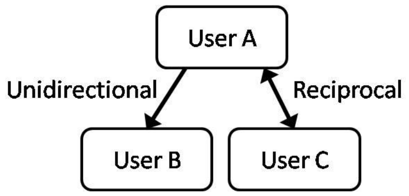


Figure 1. Directions of relation in the center of user A

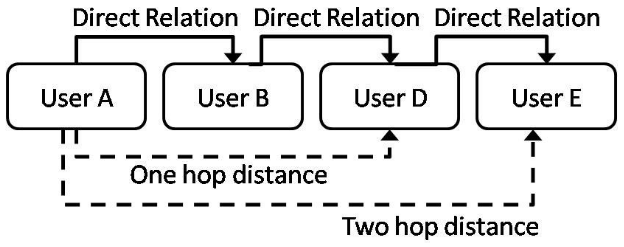


Figure 2. Relation distance in the center of user A

We also distinguish distances of connections to investigate the transitivity of common interests in the networks of trust. Three distances between users in trust networks were explored: *direct*, *one hop* and *two hops*. In the above example, user A and user B are in 'direct' relationship. If user B is trusting user D, user A and user D are in 'one hop distance' unidirectional relationship (Figure 2). If user E belongs to the watch list of user D, users A and E are in 'two hop distance' unidirectional relationship. These distances can be applied to the reciprocal relationship as well.

4. DATA ANALYSIS

In this study we tested how similar the information shared by people in trusted network is. Specifically, we counted the number of shared information *items* (academic papers) and *meta-data*. In *CiteuLike* context, authors and journals (or conferences) is a good example of meta-data.. In our study, however, we considered authors only since it is more reliable and easy to track. Following Ziegler and Golbeck [13] experience with topics (which is another kind of metadata), we expected that the users who share the same interests may not necessarily agree about specific items, but demonstrate higher agreement on the level of meta-data (authors).

Since sizes of item collections varied dramatically from user to user, we had to examine both absolute and relative similarity measures. I.e., in order to measure between-users' information similarity, we not only used absolute numbers (i.e., number of common items), but we also compared *relative* (normalized) Jaccard similarity: proportion of shared items in respect to the whole collections of connected users. We used three meaningful relative similarity measures as dependent variables. Figure 3 and the following equations explain the meaning of these measures.

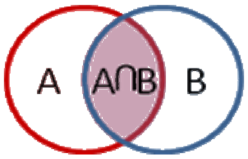


Figure 3. Information Overlap

$$\text{Inlink Power} = (A \cap B)/A \qquad \text{eq. (1)}$$

$$\text{Outlink Power} = (A \cap B)/B \qquad \text{eq. (2)}$$

$$\text{Overall Power} = (A \cap B)/(A \cup B) \qquad \text{eq. (3)}$$

If user A added user B to her trusted network (i.e, A points to B), the inlink power (impact) of the user B for the user A represents how much the information of user A is influenced by the information of user B. The outlink power of User B is how much the information of user B affects the user A. The overall power measures the fraction of overlapped information in the joint information space of both users.

For the information similarity in trusted network, the following hypotheses were assessed: **H1.** Users connected by direct or indirect relationships of trust have more similar information items and meta-data than a non-connected pairs. **H2.** Users in reciprocal relations have more similar information item and meta-data than users in unidirectional relations.

5. THE RESULTS

5.1 Information sharing in trusted network

To test whether users connected by direct or distant links of trust share more information than non-connected pairs (H1), we compared both absolute numbers of shared information items and their normalized numbers (inlink, outlink, and overall powers) using one-way ANOVA test.

First, we explored the number of shared items and meta-data. Table 2 shows mean numbers of shared items and meta-data for direct and distant relationship on contrast to a non-related pair of users (which we can interpret as infinite distance). At average, direct pairs share the largest number of items and meta-data. The numbers are decreasing with the increase of distance in the network of trust achieving its minimum for non-connected pairs. This is the evidence that users connected in a network of trust do have significantly more similar interests than non-connected users. We can also consider it as an evidence of information propagation along a network of trust, although impressive similarity on the meta-data level (which are hard to propagate!) hints that interest similarity may play a more important role than propagation in the observed phenomenon. As Table 2 shows, reciprocal relationships exhibit the same pattern, also with significant differences between columns in the number of shared information items and meta-data.

Table 2. The Average Number of the Common Information

		Direct	1hop	2hops	No Rel.
Unidirect-ional	Items	**.82**	**.20**	**.14**	**.00**
		F (3, 412315) = 6961.18, *p* < .001			
	Meta-data	**22.65**	**18.85**	**20.04**	**.02**
		F (3, 412315) = 618.37, *p* < .001			
Reciprocal	Items	**8.35**	**1.50**	**.72**	
		F (2, 1368) = 137.40, *p* < .001			
	Meta-data	**93.02**	**67.77**	**33.59**	
		F (2, 1368) = 9.16, *p* < .001			

Second, we explored differences between relative similarity measures – fractions of shared items and meta-data for unidirectional relationship (Table 3) and reciprocal relationships (Table 4). In both cases, same pattern can be observed for relative similarity measures: directly related users have the largest fraction of shared items and meta-data and the fractions decrease with the increase of the distance between users and reach the minimal level for not connected users (infinite distance).

Table 3. The Average Similarity Powers of Common Information (Unidirectional Relations)

		Direct	1hop	2hop	No Rel.
Items	Inlink	**2.01%**	**0.55%**	**0.41%**	**0.03%**
		F (3, 412315) = 2841.92, *p* < .001			
	Outlink	**0.85%**	**0.17%**	**0.10%**	**0.00%**
		F (3, 401164) = 5643.51, *p* < .001			
	Overall	**0.35%**	**0.07%**	**0.04%**	**0.00%**
		F (3, 412315) = 7696.06, *p* < .001			
Meta-Data	Inlink	**5.33%**	**1.64%**	**1.54%**	**0.02%**
		F (3, 412315) = 1969.01, *p* < .001			
	Outlink	**2.87%**	**2.88%**	**2.74%**	**0.03%**
		F (3, 401164) = 1383.66, *p* < .001			
	Overall	**1.25%**	**0.77%**	**0.76%**	**0.01%**
		F (3, 412315) = 908.51, *p* < .001			

Table 4. The Average Similarity Powers of Common Information (Reciprocal Relations)

		Direct	1hop	2hop
Items	Inlink & Outlink	**6.79%**	**1.05%**	**0.37%**
		F (2, 1368) = 160.70, *p* < .001		
	Overall	**2.45%**	**0.32%**	**0.10%**
		F (2, 1368) = 258.52, *p* < .001		
Meta-data	Inlink & Outlink	**13.01%**	**6.48%**	**3.20%**
		F (2, 1457) = 36.08, *p* < .001		
	Overall	**4.79%**	**2.47%**	**1.26%**
		F (2, 1456) = 23.92, *p* < .001		

In addition to demonstrating a clear connection between item and meta-data level similarity and user closeness in a network of trust, the data shown above allows to make interesting observations. First, as we expected, between-user similarity on the level of *meta-data* is much larger than similarity on the level of *items* for both systems. For example, the inlink power similarity of items in direct relation is 2.01% while inlink power similarity of meta-data in the same direct relation was 5.33%. Second, both absolute and relative similarities are pair-wise larger for reciprocal than for unidirectional connections for all distance levels. This difference is most pronounced in relative form reaching its highest level for direct reciprocal relations (6.79% for items and 13.01% for metadata). Next section examines the difference between reciprocal and unidirectional connections in details and checks its significance.

5.2 Unidirectional vs. Reciprocal Relations

To compare the differences of information sharing pattern between unidirectional and reciprocal relations, we started with comparing the number of shared information items and meta-data, doing it now separately for several distances of relations. In all three distances but meta-data of 2-hop connection, the numbers of shared information items and meta-data in reciprocal relations were significantly larger than in unidirectional relations. In case of meta-data of 2-hop relation, there was no significant difference.

Secondly, we checked the significance of observed differences in relative information item similarity between reciprocal and unidirectional relations (Table 6). For direct and 1-hop relationship, the differences appeared to be significant, i.e., users connected by a direct or 1-hop distanced reciprocal relation shared significantly larger fractions of information items than users connected by unidirectional relation. For 2-hops relations the observed difference appeared to be non-significant for one out of three relative similarity measures.

Table 5. Results for Powers of Information Items

		df	*t*-value	Sig.
Inlink Power	Direct	11478	-7.39*	p < .001
	1Hop	17787	-2.57*	p = .010
	2Hop	30568	.321	p = .748
Outlink Power	Direct	11478	-21.35*	p < .001
	1Hop	17787	-14.05*	p < .001
	2Hop	30568	-7.70*	p < .001
Overall Power	Direct	11478	-21.09*	p < .001
	1Hop	17787	-13.08*	p < .001
	2Hop	30568	-5.93*	p < .001

On the final step we compared relative information meta-data similarity for reciprocal and unidirectional relations (Table 7). The relative source similarity was significantly higher for users connected by direct and 1-hop reciprocal relation than for users connected by unidirectional relations of the same distance. Two out of three relative similarities were significantly larger for reciprocal relations.

Table 6. Test Results about Power of Meta-data

		df	*t*-value	Sig.
Inlink Power	Direct	11356	-6.83*	p < .001
	1Hop	17596	-10.66*	p < .001
	2Hop	30597	-4.82*	p < .001
Outlink Power	Direct	11356	-12.79*	p < .001
	1Hop	17596	-6.11*	p < .001
	2Hop	30597	-1.00	p = .318
Overall Power	Direct	11356	-9.71*	p < .001
	1Hop	17596	-7.52*	p < .001
	2Hop	30597	-2.78*	p = .005

6. CONCLUSION AND DISCUSSION

To prove the feasibility of users' self-defined relations of trust as the bases of recommendation, we examined how similar interests of users connected by a self-defined relation of trust are. Using *CiteuLike* datasets, we found that user connected by a self-defined relation of trust have more common information items and meta-data than user pairs with no connection. The similarity was largest for direct connections and decreased with the increase of distance between users in the network of trust. Users involved in a reciprocal relationship exhibited significantly larger similarity than users in a unidirectional relationship on all levels. Moreover, similarity on the level of meta-data (authors) was larger than similarity on the level of individual items (references).

While the results of our study support the idea of using networks of trust in CF systems, they still do not answer the question how to use this information to improve the quality of recommendation. In out future studies we plan to address this issue. As the first step, we will investigate the impact of trusted networks on recommendation quality using our *CiteuLike* data set. We will also explore how information propagates within trusted networks and investigate the influence of information authorities who play a leading role in disseminating the information. In later studies, we plan to expand our target domains by adding different data sets.

7. REFERENCES

[1] Adamic, L. A. & Adar, E. (2003) Friends and neighbors on the Web, *Social Networks*, 25 (3), pp. 211 ~ 230.

[2] Bonhard P. & Sasse M. A. (2006) Knowing me, Knowing you - Using Profiles and Social Networking to Improve Recommender Systems, *BT Technology Journal*, Vol. 25 (3)

[3] Lam, S. K. & Riejl, J. (2004) Shilling Recommender Systems for Fun and Profit, In: *Proceedings of World Wide Web 2004*, New York, NY, USA, pp. 393 ~ 402

[4] Massa, P. & Avesani, P. (2004) Trust-aware Collaborative Filtering for Recommender Systems, In: *Proceedings of Federated International Conference On The Move to Meaningful Internet*, Agia Napa, Cyprus, pp. 492 ~ 508

[5] Massa, P. & Avesani, P. (2007) Trust-aware Recommender System, In: *Proceedings of Recommender System 2007*, Minneapolis, MN, USA, pp. 17 ~ 24

[6] Maltz, D. & Ehrlich, K. (1995) Pointing the Way: Active Collaborative Filtering, In, Proceeding of CHI' 95, Denver, CO, USA, pp. 1 ~ 8

[7] McKnight, D. H. & Chervany, N. L. (1996). The Meanings of Trust. University of Minnesota, http://misrc.umn.edu/wpaper/WorkingPapers/9604.pdf (accessed on July, 2008)

[8] Mehta, B., Hofmann, T., Nejdl, W. (2007) Robust collaborative filtering, In: Proceedings *of Recommender System 2007*, Minneapolis, MN, USA, pp. 49 ~ 56

[9] O'Donovan, J. & Barry, S. (2005) Trust in Recommender Systems, In: *Proceedings of the 10th International Conference on Intelligent User Interfaces*, San Diego, California, USA, pp. 167 ~ 174.

[10] Schafer, J. B., Frankowski, D., Herlocker, J. & Sen, S. (2007) Collaborative Filtering Recommender System, In: Brusilovsky, P., Kobsa, A. & Nejdl, W. (Eds.) *The Adaptive Web: Methods and Strategies of Web Personalization*, pp. 291 ~ 324

[11] Singla, P. & Richardson, M. (2008) Yes, There is a Correlation - From Social Networks to Personal Behavior on the Web, In: *Proceeding of the 17th International World Wide Web Conference*, Beijing, China.

[12] Sinha, R. & Swearingen, K. (2001) Comparing Recommendations Made by Online Systems and Friends, In: *Proceedings of DELOS Workshop on Personalisation and Recommender Systems*

[13] Tintarev, N. & Masthoff, J. (2007) Effective explanations of recommendations: user-centered design. In: *Proceedings of Recommender System 2007*, Minneapolis, MN, USA, pp. 153-156

[14] Ziegler, C. & Golbeck, J. (2007) Investigating interactions of trust and interest similarity, *Decision Support Systems*, 43, pp. 460 ~ 475

A Holistic Approach to Enhance the Doctor-Patient Relationship for Diabetes Using Social Networking, Personalized Alerts, Reminders, and Recommendations

William WL Yip
University of Hawaii, Manoa
1680 East-West Road,
Honolulu, HI 96822, USA
1-808-956-9988
wyip@hawaii.edu

Luz M. Quiroga
University of Hawaii, Manoa
1680 East-West Road, POST 314B,
Honolulu, HI 96822, USA
1-808-956-9988
lquiroga@hawaii.edu

Abstract

This paper describes an ongoing project that proposes the conceptual design of a decision-support system (DSS) based on patient modeling that enhances the communication and relationship among health care providers and patients with diabetes. This project attempts to answer the following two research questions: 1) What are the challenges in the current relationship between a diabetic patient and his/her health care providers? 2) Can a DSS based on providing motivation support through social networking, personalized alerts, reminders, and recommendations improve objective and subjective factors that affect the overall health outcome of a diabetic patient?

Categories and Subject Descriptors

H.4.2 [**Information Systems Applications**]: Types of Systems – *Decision support (e.g. MIS)*

General Terms

Design, Experimentation, Human Factors

Keywords

Compliance, Diabetes, Empowerment, Information Filtering, Personalization, Social Networking

1. Introduction

This paper describes an ongoing project that proposes the conceptual design of a decision-support system (DSS) based on patient modeling that enhances the communication and relationship among health care providers (i.e. physicians and nurses) and patients with diabetes. With more than 23 million Americans suffering from diabetes [3], health care providers and researchers have devised ways to improve diabetic patients' overall health outcome as well as to reduce expensive acute episodes as a result of non-compliant lifestyle activities [14]. In spite of these efforts, there remains a gap in the communication channel among health care providers and diabetic patients. This gap is attributed by the fact that most health care providers resort to the traditional model of compliance and adherence to treat chronic illnesses like diabetes [2]. This model, which was based on a health care system that provided the majority of its treatment for acute illnesses [1], can have potential damaging effects on the provider-patient relationship. Instead, the empowerment approach emerged in the early 1990's as a new model to promote equal partnership among providers and diabetic patients [7, 9, 12, 17]. A DSS based on patient modeling can potentially facilitate this new approach. Effective communication among health care providers and patients can be facilitated by a DSS that:

For patient:

- Provides motivational support through social networking sites (SNSs).
- Provides alerts and reminders to motivate patient to comply with lifestyle-changing activities.
- Provides personalized recommendations of trusted health-related information based on individual patient's situation.

For health care provider:

- Provides personalized alerts and reminders when his/her patient's physiological parameters (e.g. blood glucose level) are out of range.
- Provides personalized recommendations of treatment options based on evidence-based guidelines.

Situation of each patient is unique. In [13], the study showed the importance of context in users' relevance feedback in information filtering (IF) systems for delivery of personalized consumer health information. The study identified non-topical characteristics such as lifestyle, domain expertise, credibility of information sources, and comprehensibility. To increase patients' motivation to change behavior, providing the right information to the right patient at the right moment is crucial. To achieve this, a holistic model of the patients is required. In the health care domain, many opportunities exist for profiling patients. A holistic model may include:

- Electronic Medical Record (EMR)
- Vital signs and physiological parameters collected from outpatients settings (e.g. blood pressure, blood glucose level)
- Quality-of-Life (QOL) issues (e.g. food intake, exercise)

- Web browsing behavior that includes health-related websites, social networking sites (SNSs), and patient support groups

This information can be fed to an agent-based DSS which in turn provides alerts, reminders, and recommendations to both health care providers and patients.

2. Conceptual Framework

Why do diabetic patients remain non-compliant to lifestyle-changing activities in spite of the physical, psychological, and financial burdens that diabetes place on them? The problem lies in the application of the compliance and adherence model in diabetic care. In chronic diseases like diabetes, this model places the patients in a submissive position obeying authoritative care providers [8]. Health care providers often feel frustrated with their patients' non-compliant activities. Vice versa, patients feel frustrated with their lack of knowledge and understanding of the disease as well as blames from their care providers for their non-compliant activities. What is needed is an approach that: 1) redefines the roles and responsibilities of both patients and care providers; 2) create a relationship that promotes collaboration and partnership [5]. Considerable amount of research has been done to facilitate this approach. Since the early 1990's, there has been a push for patient empowerment that gives control to both patients and care providers [7, 9, 12, 17]. In [5], the authors even downplayed the importance of compliance, claiming that compliance becomes irrelevant if patients are "viewed as collaborators who establish their own goals". In a community-based diabetes self-management education program [6], the study encouraged patients to find their own solutions that fit their psychological and physical needs. In a study conducted with 85 type-2 diabetic patients [12], individuality was identified as one of the five issues that are pivotal to effective management of the disease. It is a patient-centered approach where information delivered to patients is based on their individual needs and concerns.

Diabetic patients need to make informed decisions in order to manage their disease effectively. Informed decisions are based on information provided to patients pertaining to their individual needs and circumstances. Personalized recommendations are provided to diabetic patients in a health information tailoring system called Violet Technology (VT) in [9]. VT is a web portal that performs information filtering and prioritization based on patients' profiles in Diabetes Information Profile (DIP). There are 5 components in DIP:

- Diabetes-related situation: current lifestyle, diabetes education exposure, self blood glucose tests, medications.
- History of information browsing
- Patient information preference
- History of quizzing
- History of agenda generation

The presentation of the information is adapted based on a two-step process. First, information is filtered using a series of rules (e.g. removing female issues for male patients). Then, information is ordered by its significance based on priority assigned to each information item. The patient-modeling approach of VT allows diabetic patients to access information relevant to their individual situations more efficiently. Furthermore, the agenda service allows patients to generate a list of 5 questions that they can bring to their health care providers during their office visits. Although this research shows the face validity of such information tailoring system for diabetic patients, it falls short of being a comprehensive approach for both diabetic patients and their health care providers. Such approach can help bridge the communication gap and provide an environment of equal partnership among all stakeholders. Research has shown that a decision-support system can help health care providers follow clinical guidelines, which eventually leads to improved care [15].

Motivation is a key component in successful management of diabetes. Self-determination theory distinguishes between autonomous and controlled behaviors [18, 19]. Patients are autonomously motivated when their desire to change behaviors comes from within themselves; while behaviors are controlled when patients are pressured from external forces to change their daily activities. Two separate studies [18, 19] showed that patients' autonomous motivation is strongly correlated to their perception of their providers' autonomous support. It is autonomous motivation that leads to patients' competence in making lifestyle changes, and is therefore an important factor to be considered in reducing the communication gap among diabetic patients and health care providers.

Social networking sites (SNSs), mostly in terms of support groups around health issues, has a long tradition, starting with first generations of social tools of the 1980's exemplified by the "Well" community in Rheingold's book "The Virtual Community" to activities such as Sermo (http://www.sermo.com/) - social networking for licensed physicians, NurseConnect (http://www.nurseconnect.com/),Nurses' Lounge (http://www.nurseslounge.com), and specific groups (by illness, treatment, therapies, etc). More recently, progress has been made in Second Life (http://secondlife.com) with islands such as Health Info Island, Karuna (AIDS), Virtual Ability Island (disabilities), and Rachelville (parents of terminally-ill kids). Virtual events are held to promote "social engagement" such as the recent Helen Keller Day, organized by EASI: Equal Access to Software and Information (http://easi.cc), as part of its commitment with students and professionals with disabilities to have the same right to access information technology as everyone else.

In a pilot study [11], five participants were interviewed on their perceptions (both positive and negative aspects) on SNSs and which properties of SNSs can facilitate online support and adherence to health-related regimens. The study found that SNSs were most instrumental in providing emotional (e.g. encouragement from a friend) and informational (e.g. a tip to perform a task) support. Furthermore, users of SNSs tend to build and strengthen existing relationships among family members and friends rather than to meet new friends. This preliminary result suggests that SNSs can potentially have a positive effect for diabetic patients who rely on their close ones for motivation. In the following section, a solution incorporating personalized alerts, reminders, and recommendations, as well as social networking features will be proposed.

3. Solution

This section provides a brief description of each component of the DSS that this project proposes. Figure 1 below illustrates a conceptual diagram of an agent-based DSS that provides personalized alerts, reminders, and recommendations, as well as motivations through social networking.

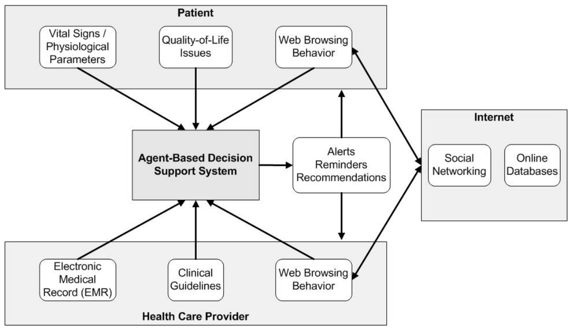


Figure 1 Conceptual Design of a design-support system for diabetic patients and health care providers

Although they provide similar functions, it is important to point out the minor differences between alerts, reminders, and recommendations. The following scenario illustrates the differences:

> *John is a 56-year-old man who was diagnosed with diabetes 5 years ago. He monitors his blood glucose level daily using a blood glucose meter provided by his primary care physician. He receives an* ***alert*** *from the meter when his blood glucose level is 10% above his acceptable range. He has an office visit with his dietician every 6 months. He creates a* ***reminder*** *on Google Calendar to remind himself of the appointments. Lastly, John's brother was recently diagnosed with diabetes so John sent an online article from WebMD about* ***recommendations*** *on how to perform foot care on a periodic basis.*

From the patients' perspective, there are 3 primary data sources where profiling information can be collected: 1) vital signs and physiological parameters; 2) QOL issues; and 3) web browsing behaviors. Vital signs and physiological parameters are measurements that are collected periodically by patients themselves in remote settings. Examples are blood glucose level and blood pressure. QOL issues are qualitative indicators of how well patients are managing their disease. QOL issues may be the amount of exercise that a patient is performing daily, or the lifestyle preferences of the patients (e.g. smoker, preference to alternative treatment, their personal goal, plans, strategies, success and impediments regarding their management of the disease). Together, quantitative measurements of vital signs and physiological parameters and qualitative indicators of QOL issues form a unique model of each patient. In an exploratory study [10], they designed the CHAP (Continuous Health Awareness Program) system that engage patients to reflect on their breakdown activities and to build correlations between these activities and the collected data on the patients' blood glucose values. Based on the collected data on an individual's vital signs, physiological parameters, and QOL issues, a DSS can provide personalized alerts, reminders, and recommendations.

Patients' web browsing behaviors provide a clue on what their information needs are. For instance, a patient who often searches for information about a particular drug indicates that he may be prescribed with the drug and in need for additional information (e.g. recent findings on side effects). A DSS can provide personalized recommendation of health-related information from trusted source (e.g. National Library of Medicine, MedlinePlus).

From the health care provider's perspective, patients' clinical data includes demographic information about the patient, clinical test results, history of drug prescriptions, history of vital signs and physiological parameters, and etc. This, typically in the form of Electronic Medical Record (EMR), is an enormous source of information that provides many opportunities for decision-support services based on evidence-based guidelines. Integrating evidence-based guidelines with EMR can help reduce practice variability and improves the overall quality of care for patients.

The last component of the DSS is an online social network for both diabetic patients and health care providers. Diabetic patients, health care providers, family members, and friends participate in a common medium to provide emotional and information support for each other. Members can write messages in public (e.g. a "wall" on Facebook) as well as private areas. Patients can also post updates on their health status. In addition, games and puzzles can be used to educate members, increase participation, and keep members interested over a longer duration.

4. Research Methodology

This ongoing project proposes the following research methodology, which is broken down into 3 phases.

Phase 1 of the project will focus on the relationship among diabetic patients and their respective care providers. The research question of this phase is:

> *What are the challenges in the current relationship between a diabetic patient and his/her health care providers? Why are diabetic patients non-compliant to lifestyle-changing activities?*

To answer this research question, in-depth interview sessions will be conducted with diabetic patient educators and coordinators from various health institutions. Patient educators act as intermediaries between patients and care providers who can provide their unbiased opinions. In a way, they are "human agents" that perform similar functions that a potential DSS could do. The focus of the interview questions will center on the existing communication means (or lack thereof) among diabetic patients and health care providers.

Phase 2 of this project involves the conceptual design of the social networking component of a DSS. The specifications and design of this component will be based on interview results from Phase 1. In Phase 2, the prototyped social networking component will be integrated with an existing DSS called Comprehensive Disease Management Program (CDMP) (http://www.estenda.com), which is based on the Chronic Care Model [16]. CDMP, currently operational in more than 70 clinics in the Indian Health Service, allows patients to upload their physiological parameters (e.g. blood glucose) and images. This data is then combined with laboratory results and other patient records to provide decision-support services to patients and health care providers. The social networking component implemented in this phase will build upon the existing patient profile in CDMP.

The last phase of this project is to evaluate the effectiveness of the social networking component of a DSS implemented in Phase 2. The research question of this phase is:

> *Can a DSS based on providing motivation support through social networking improve objective and subjective factors that affect the overall health outcome of diabetic patients?*

The independent variable of this experimental study is the existence of a social networking component in health care setting; while the dependent variable is the objective and subjective factors that affect a diabetic patient's overall health outcome. The control group is a group of diabetic patients and health care providers who will not be provided with the social networking component; while the experimental group will evaluate the social networking component implemented in Phase 2. About 50 participants will be chosen from existing users of the CDMP into the control and experimental groups, respectively.

Objective factors are facts that can be measured quantitatively (e.g. patient compliance to measuring blood glucose level, number of times a patient is admitted to a hospital because of complications). On the other hand, subjective factors (e.g. motivation to comply with lifestyle-changing activities, patients' perception of the health care system) are more difficult to measure quantitatively. In this phase, both qualitative and quantitative data will be collected. Objective factors will be measured through a self-reported questionnaire comprised of 6 compliance components: exercise, hypoglycaemia, foot care, diet, home monitoring, and drug [4]. Statistical analysis will be performed to measure the effects of the existence of the social networking component. Subjective factors will be evaluated through an interview session with a subset of the experimental group. The focus of the interview session is to determine whether social networking has an effect on diabetic patients' perception on their health care providers. Aspects of the perception of their health care providers include trust, availability, satisfaction, optimism, and etc. These factors correspond to the autonomous support that self-determination theory identifies and have a major effect on patients' ability to comply to lifestyle-changing activities.

5. References

[1] Anderson, R. M., & Funnell, M. M. (2000). Compliance and adherence are dysfunctional concepts in diabetes care. The Diabetes Educator, 26(4), 597.

[2] Anderson, R. M., & Funnell, M. M. (2005). Patient empowerment: reflections on the challenge of fostering the adoption of a new paradigm. Patient Education and Counseling, 57(2), 153-157.

[3] Centers for Disease Control and Prevention. (2008). National diabetes fact sheet: general information and national estimates on diabetes in the United States, 2007. Atlanta, Georgia: U.S. Department of Health and Human Services, Centers for Disease Control and Prevention.

[4] Chan, Y. M., & Molassiotis, A. (1999). The relationship between diabetes knowledge and compliance among Chinese with non-insulin dependent diabetes mellitus in Hong Kong. Journal of Advanced Nursing, 30(2), 431-438.

[5] Funnell, M. M., & Anderson, R. M. (2000). The problem with compliance in diabetes. The Journal of the American Medical Association, 284(13), 1709.

[6] Funnell, M. M., & Anderson, R. M. (2002). Working toward the next generation of diabetes self-management education. American Journal of Preventive Medicine, 22(4), 3-5.

[7] Funnell, M. M., & Anderson, R. M. (2003). Patient empowerment: a look back, a look ahead. Diabetes Educator, 29(3), 454-464.

[8] Lutfey, K. E., & Wishner, W. J. (1999). Beyond "compliance" is "adherence". Diabetes Care, 22(4), 635-639.

[9] Ma, C., Warren, J., Phillips, P., & Stanek, J. (2006). Empowering patients with essential information and communication support in the context of diabetes. International Journal of Medical Informatics, 75(8), 577-596.

[10] Mamykina, L., & Mynatt, E. D. (2007). Investigating and supporting health management practices of individuals with diabetes. In Proceedings of the 1st ACM SIGMOBILE international workshop on Systems and networking support for healthcare and assisted living environments. San Juan, Puerto Rico: ACM New York.

[11] Olsen, E., & Kraft, P. (2009). ePsychology: a pilot study on how to enhance social support and adherence in digital interventions by characteristics from social networking sites. In Proceedings of the 4th International Conference on Persuasive Technology (pp. 1-6). Claremont, California: ACM New York.

[12] Pooley, C. G., Gerrard, C., Hollis, S., Morton, S., & Astbury, J. (2001). 'Oh it's a wonderful practice... you can talk to them': a qualitative study of patients' and health professionals' views on the management of type 2 diabetes. Health & Social Care in the Community, 9(5), 318-326.

[13] Quiroga, L. M., & Mostafa, J. (2002). An experiment in building profiles in information filtering: the role of context of user relevance feedback. Information Processing and Management, 38(5), 671-694.

[14] Steinbrook, R. (2006). Facing the diabetes epidemic: Mandatory reporting of glycosylated hemoglobin values in New York City. New England Journal of Medicine, 354, 545-548.

[15] Vashitz, G., Meyer, J., Parmet, Y., Peleg, R., Goldfarb, D., Porath, A., et al. (2009). Defining and measuring physicians' responses to clinical reminders. Journal of Biomedical Informatics, 42(2), 317-326.

[16] Wagner, E. H. (1998). Chronic disease management: what will it take to improve care for chronic illness? Effective Clinical Practice, 1, 2-4.

[17] Weiss, M. A. (2006). Empowerment: a patient's perspective. Diabetes Spectrum, 19(2), 116-118.

[18] Williams, G. C., Freedman, Z. R., & Deci, E. L. (1998). Supporting autonomy to motivate patients with diabetes for glucose control. Diabetes Care, 21(10), 1644-1651.

[19] Williams, G. C., Rodin, G. C., Ryan, R. M., Grolnick, W. S., & Deci, E. L. (1998). Autonomous regulation and long-term medication adherence in adult outpatients. Health Psychology, 17, 269-276.

Using Wikipedia Content to Derive an Ontology for Modeling and Recommending Web Pages across Systems

Pei-Chia Chang
Department of Information & Computer Science
1680 East-West Road
Honolulu, HI 96822, USA
1-808-2209701
pcchang@hawaii.edu

Luz M. Quiroga
Department of Information & Computer Science
1680 East-West Road
Honolulu, HI 96822, USA
1-808-9569988
lquiroga@hawaii.edu

ABSTRACT

In this work, we are building a cross-system recommender at the client side that uses the Wikipedia's content to derive an ontology for content and user modeling. We speculate the collaborative content of Wikipedia may cover many of the topical areas that people are generally interested in and the vocabulary may be closer to the general public users and updated sooner. Using the Wikipedia derived ontology as a shared platform to model web pages also addresses the issue of cross system recommendations, which generally requires a unified protocol or a mediator. Preliminary tests of our system may indicate that our derived ontology is a fair content model that maps an unknown webpage to its related topical categories. Once page topics can be identified, user models are formulated through analyzing usage pages. Eventually, we will formally evaluate the topicality-based user model

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval-- Information filtering; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval -- Clustering D.3.3

General Terms

User Modeling, Wikipedia, Management, Measurement, Documentation, Design, Experimentation.

Keywords

Recommender, Agent, User Modeling, Ontology.

1. INTRODUCTION

User modeling through content is one common solution in recommending web pages across systems [3,7,9,14]. In this work, we are interested in using the collaborative content of Wikipedia to derive an ontology as a unified knowledge base for modeling web pages. Wikipedia is one of the world's largest collaborative knowledge bases. Although there are only a few contributors(less than 10% of user population) to the content of Wikipedia[8], it has a huge pool of readers. As Sussan describes, "with Web 2.0 products, it is the user's engagement with the website that literally drives it."[13] Similarly, we speculate Wikipedia's content and its vocabulary may cover recent and popular topical areas that people are generally interested in. The language in Wikipedia may be closer to what the general public use, instead controlled by domain experts. We emphasize the topics, but not content accuracy, from Wikipedia may reflect the dynamic information on the Internet.

Our recommender formulates a user model based on the browsing behavior at a client side and the usage pages mapped to the derived ontology. Given the research potentials of Wikipedia's content, we are interested in the performance of recommending web pages based on the Wikipedia derived ontology. Our research question is "Does the recommender based on the Wikipedia's content model provide topically relevant recommendations?"

2. Related Work

Content-based recommenders include WebWatcher[6], Syskill & Webert[10], WebMate[5], and ifWeb[2]. WebWatcher and WebMate adopt TF-IDF, the vector space model and similarity clustering. Syskill & Webert rely on feature extraction, particularly expected information gain[11], which relies on the co-existence of related keywords, and relevance feedback. The system formulates the profile vector that consists of keywords from pages of positive ratings and against pages of negative ratings. Then, Bayesian classifier is employed to determine a page's topics, and its similarity with the profile vector. ifWeb employs a semantic network and consistency-based user modeling shell[4]. In general, these four systems apply statistical approaches, such as TF-IDF or expected information gain for keywords extraction and a cluster or classifier for similarity identification. Our work borrows Wikipedia's categorization system and augments it with keywords identified by predefined heuristics as topical indices. A full listing of existing categories and indexes in Wikipedia can be found at http://en.wikipedia.org/wiki/Portal:Contents/Categorical_index.
In our study, page classification depends on the frequency of those indexing keywords appeared in a web page. Our difference from the previous systems is the use of Wikipedia's collaborative categorization system to derive an ontology that is augmented with heuristic information extraction from Wikipedia's content.

3. Method Description

3.1 System Architecture

Our recommender uses the Wikipedia's content to derive an ontology for content and user modeling. With the ontology, our system automatically assigns the Wikipedia category(s) to a new page that pass a category's threshold value, which formulates a "categorical" vector space model for the page. The system also captures user interests in the user model through the categories. Pages of similar topics with the profile will be recommended.

Figure 1 depicts the system's architecture, which will be explained in the following paragraphs.

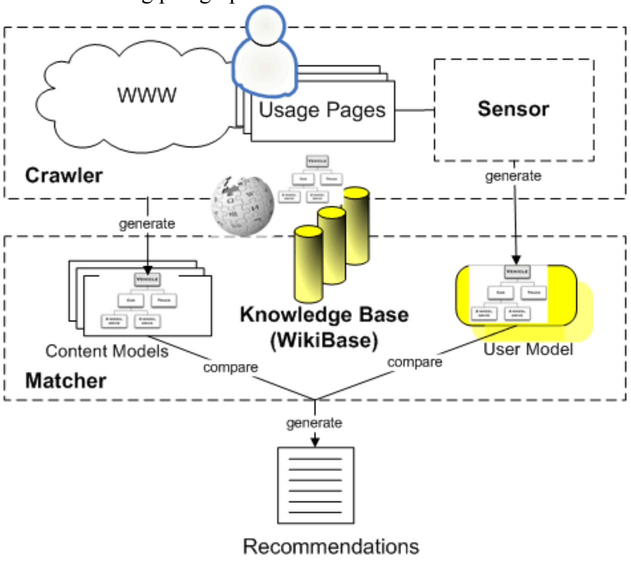


Figure 1 System Architecture

There are four major components in the system -- the crawler, the Wikipedia knowledge base (WikiBase), the sensor, and the matcher. To begin with the top part of the graph, the crawler fetches those hyperlinked pages from the usage pages as well as queries search engines based on the user model managed by the sensor. Utilizing the sensor, the crawler generates a corresponding content model for each newly fetched page.

Every component in the system uses WikiBase, which stores ontologies, keywords, content models and the user model respectively. We construct WikiBase by combining the Wikipedia's categorization system with heuristic information extraction on keywords. Heuristics include page titles, categorical labels, anchor texts, italic, bold, and TF-IDF terms. In order to associate keywords with categories, we extract heuristic keywords form pages labeled as one of the categories by the Wikipedia's editors. Therefore, each category has a list of keywords to be utilized by the sensor.

The sensor manages the user model and maps usage pages into content models. It calculates a page's topical relevance and formulates the corresponding content model according to the WikiBase's keyword weight and the word frequency of the web page. The user model is updated, on a frequency basis, by the sensor whenever it maps a usage page into a content model. Therefore, the user model is constantly evolved. In other words, if a user accesses a specific categorical topic in multiple times or through multiple pages, the user will score higher in the corresponding category of the user model. Keyword weighting and sensing formulas are defined below.

Definitions:
$|K_j|$, the number of keywords extracted for heuristic type j
$|K \in c|$, the number of keywords in category c
|Categories|, the number of categories in the knowledge base
$freq(K_{ij})$ is the frequency of keyword K_{ij} for heuristic type j
$max(K_1, \ldots ,K_m)$ is the maximum value among the m elements

The weight of keyword K_{ij} among m heuristics is:

$$W(K_{ij}) = \sum a_j \left(\frac{freq(K_{ij})}{\max(freq(K_{1j}),\ldots,freq(K_{|Kj|\,j}))} \right)$$

$1 \le i \le |K_j|,\ 1 \le j \le m,$
a_j is a weighting coefficient assigned to heuristic j.

A page's Relevance Score R_c to a category c is:

$$R_c = \sum d\, W(K_i \in c)\ \alpha^{freq(K_i \in c)}$$

$1 \le i \le |K \in c|,\ 0.5 < \alpha < 1,\ 1 \le c \le |Categories|$

$$\begin{cases} 0 < d < 0.5, & \text{partial match} \\ d = 1, & \text{full match} \end{cases}$$

As for the matcher, it compares the cosine similarity of those crawler-retrieved pages with the user model and then generates recommendations. In addition to cosine similarity, the matcher also relies on the ontological structure of WikiBase. With the structure, topical association among web pages can be revealed and it also helps to identify if a user is interested in particular domains or not. We define two indices (diversity and specificity) to represent the coverage of user interests. The following describes the procedure.

At the beginning, construct a minimal spanning tree that traverses all the identified categories in the user model. Identified categories are those categories with a Relevance Score over a predefined-threshold. In order to connect identified categories together, connecting nodes, such as parents or neighbors of the identified categories may be added to the tree. Definitions of the two indices are as follows:

Diversity index: count the number of edges of the minimal spanning tree and normalize it by dividing the number of identified nodes, excluding connecting nodes, in the spanning tree.
Specificity index: sum the minimal distances from the root to all identified categories respectively and normalize it by dividing the number of those identified categories, excluding connecting nodes.

3.2 Evaluation Method

We plan to recruit a few participants (< 10) in the computer science domain where we derive WikiBase. Each of them has to rate a collection (> 300) of web pages based on topical relevance and novelty. They have to provide certain web pages (>30) of their interest in advance as the usage source of formulating the user models. Afterward, they have to rate the collection. The ratings will be divided into a training and validation set. Our system will tune the keyword weight based on the training data. We will compare our system performance with the SMART system[12], which utilizes vector space model as well.

4. Current Status & Discussion

4.1 Current Status

We have built WikiBase in the computer science domain, listed at http://www2.hawaii.edu/~pcchang/ICS699/results.html. We selected the domain due to its rich data. Two preliminary tests were conducted on two computer science professionals. In the first one, w tested the following pages for their topical relevance.

http://www.algosort.com/ (A)
http://tc.eserver.org (B)

Considering only the top two ranking, page A is sensed as "algorithms" and "genetic algorithms" categories; page B is sensed as "human-computer interaction" and "usability" categories. In the evaluation of the classification result, both participants' rankings are the same as the system's ranking, considering only the top two.

In the second test, we selected fourteen pages, listed in the appendix, from four topical areas – algorithm, data mining, human computer interaction (HCI) and computer games. Both participants have to evaluate at least five categorical keywords of each page. They have to provide the degree of agreement from 1 (disagree) to 5(agree) about the following statement. "The given phrase is a topical keyword of the page." The given phrase is a categorical label generated by the sensor for each page. The following table summarizes the ratings.

	Participant 1	Participant 2
Algorithm (3 pages)	3.88	2.83
Data Mining (4 pages)	3.95	3.40
HCI (4 pages)	4.22	4.27
Games (3 pages)	2.67	2.2
Average	3.67	3.2

Table 1 Evaluation of Categorical Keywords

From the result, the ranking of both participants' average scores is: HCI, Data Mining, Algorithm, and Games. Except for the game topic, the agreement score is around 4 for participant 1 and 3.5 for participant 2. We suspect that due the wide coverage of computer games, our system performs worse in that category. Another reason may be because of the nature of computer science category. It reflects the common scientific techniques of theory for producing computer games, which is different from the tested pages that viewing computer games from a player's perspective.

We are still in the process of tuning up the keyword weight by utilizing the computer science pages from Open Directory Project (DOP) [1]. Pages in ODP are manually categorized by its users and we use the classification to evaluate our sensor. As for evaluating the recommendations, we are training the matcher with pages of a different topical coverage. Eventually, we will apply the evaluation method described earlier.

4.2 Discussion

Using the Wikipedia categories as an ontological model yields a simple user profile. This modeling approach benefits significantly in cross-system recommendations. Our recommendation engine works at the client side, which eases the privacy concern of disclosing sensitive information at web servers. Combining categories with heuristic information extractions leaves rooms for the selection of heuristics. Different domains or user groups are able to apply heuristics of interest. Given the above mentioned advantages, we are looking forward to see the results of our evaluation.

5. Future Work

The Wikipedia content and categorization system play an important role in our method to generate recommendations. Our work emphasizes the framework to automate the ontology generation and its performance in recommendations. Nevertheless, the quality of Wikipedia content is controversial. It will be worthwhile to adopt the same framework to another Wikipedia-like platform with a different user group, such as domain experts, to ensure the content quality.

Another interesting area is to study the content statistics, such as volume or the granularity of the categories, with recommendation performance. Not every domain in Wikipedia contains rich categories and articles like computer science. Therefore, the performance of recommendations may be related to some of the statistics.

6. Appendix

Due to limited space, only 1st page of each selected topic displays the categorical keywords.

Algorithms
http://www.algosort.com/
(Algorithms, Genetic algorithms, Root-finding algorithms, Networking algorithms, Disk scheduling algorithms)
http://www.oopweb.com/Algorithms/Files/Algorithms.html
http://cgm.cs.mcgill.ca/~godfried/teaching/algorithms-web.html

Data Mining
http://www.data-mining-guide.net/
(Databases, Algorithms, Knowledge representation, Natural language processing, Knowledge discovery in databases, Machine learning, Data mining)
http://www.thearling.com/
http://databases.about.com/od/datamining/Data_Mining_and_Data_Warehousing.htm
http://www.ccsu.edu/datamining/resources.html

HCI
http://www.pcd-innovations.com/
(Human-computer interaction, Human-computer interaction researchers, Usability, Computer science organizations, Artificial intelligence, Software development)

http://www.nathan.com/
http://nooface.net/
http://www.hcibib.org/

Games
http://www.robinlionheart.com/gamedev/genres.xhtml
(Image processing, Computer programming, Demo effects
Regression analysis, Computer graphics)
http://open-site.org/Games/Video_Games/
http://www.literature-study-online.com/essays/alice_video.html

7. REFERENCES

[1] http://www.dmoz.org/

[2] Asnicar, F., & Tasso, C. 1997. ifWeb: A Prototype of User Model-Based Intelligent Agent for Document Filtering and Navigation in the World Wide Web. In Proceedings of the 6th International Conference on User Modeling.

[3] Billsus, D., & Pazzani, M. 1999. A Personal News Agent that Talks, Learns and Explains. In Proceedings of the 3rd Ann. Conf. Autonomous Agents.

[4] Brajnik, G., & Tasso, C. 1994. A Shell for Developing Non-Monotonic User Modeling Systems. Human-Computer Studies, 40, 31-62.

[5] Chen, L., & Sycara, K. 1998. WebMate: a personal agent for browsing and searching. In Proceedings of the second international conference on Autonomous agents, Minneapolis, Minnesota, United States

[6] Joachims, T., Freitag, D., & Mitchell, T. 1997. WebWatcher: A Tour Guide for the World Wide Web. In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence.

[7] Mooney, R. J., & Roy, L. 2000. Content-based book recommending using learning for text categorization. In Proceedings of the fifth ACM conference on Digital libraries, San Antonio, Texas, United States.

[8] Ortega, F., Gonzalez-Barahona, J. M., & Robles, G. 2008. On the Inequality of Contributions to Wikipedia. In Proceedings of the 41st Annual Hawaii International Conference on System Sciences.

[9] Pazzani, M., & Billsus, D. 1997. Learning and Revising User Profiles: The Identification of Interesting Web Sites. Machine Learning, 27, 313-331.

[10] Pazzani, M., Muramatsu, J., & Billsus, D. 1996. Syskill & Webert: Identifying Interesting Web Sites. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, Portland, Oregon, United States.

[11] Quinlan, J. R. 1986. Induction of Decision Trees. Machine Learning, 1(1), 81-106.

[12] Salton, G., & Lesk, M. E. 1965. The SMART automatic document retrieval systems -- an illustration. Commun. ACM, 8(6), 391-398.

[13] Sussan, G. 2007. Web 2.0 The Academic Library and the Net Gen Student (pp. 35): ALA editions.

[14] Zhang, Y., Callan, J., & Minka, T. 2002. Novelty and Redundancy Detection in Adaptive Filtering. In Proceedings of the 25th Ann. Int'l ACM SIGIR Conf.

Personalised Tag Recommendation

Nikolas Landia
University of Warwick
Coventry CV4 7AL
UK
N.Landia@warwick.ac.uk

Sarabjot Singh Anand
University of Warwick
Coventry CV4 7AL
UK
S.S.Anand@warwick.ac.uk

ABSTRACT

Personalised tag recommenders are becoming increasingly important since they are useful for many document management applications including social bookmarking websites. This paper presents a novel approach to the problem of suggesting personalised tags for a new document to the user. Document similarity in combination with a user similarity measure is used to recommend personalised tags. In case the existing tags in the system do not seem suitable for the user-document pair, new tags are generated from the content of the new document as well as existing documents using document clustering. A first evaluation of the system was carried out on a dataset from the social bookmaking website, Bibsonomy[1]. The results of this initial test indicate that adding personalisation to an unsupervised system through our user similarity measure gives an increase in the precision score of the system.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Indexing Methods*; I.5.3 [**Pattern Recognition**]: Clustering

General Terms

Algorithms

Keywords

tag recommendation, tag extraction, social bookmarking

INTRODUCTION

With the advancement of Web 2.0 and social bookmarking websites, the recommendation of tags for these systems is becoming increasingly important. The major difficulties with tag recommendation for social bookmarking are that tags are related to specific users and the documents, users and tags might be unknown to the recommender system. Suggested tags have to be personalised and the system has to be able to successfully recommend tags for scenarios where the document, user and/or appropriate tag are not in the training set.

Existing approaches to automated tagging include supervised as well as unsupervised tag recommender systems. Classifiers [2] and clustering [5] have been used when a set of predefined tags is known and the task is to assign these tags to new documents. However, these supervised systems do not have a solution to the new tag problem, akin to the new item problem in recommender systems. Clustering can be used to generate tags from the vocabulary of the document set and is thus able to overcome the new tag problem. However, tags from a vocabulary different from that of the document's authors will not be recommended. The supervised and unsupervised techniques on their own ignore information about users that may be available to the system. People differ in their interests (documents/topics), vocabulary (tags) and context. In order to successfully recommend tags to users, vocabulary and tagging habits have to be taken into consideration.

In this paper we suggest a solution to this problem that consists of clustering the existing documents in order to identify sets of similar documents which in turn identifies the set of users whose tags may be propagated to the current target user-document pair. A list of potential tags for the new user-document pair is obtained from the tags of the similar documents. A score is then calculated for each potential tag by taking a weighted combination of the similarity of the document that the tag is assigned to and the similarity of the user who assigned it, averaged over all posts the tag appears in. If the score of a tag is below a set threshold, it means that this tag is unsuitable for the user-document pair. When the number of suitable tags is below a predefined number t, new tags are generated from the content of the document.

DOCUMENT CLUSTERING

There are various techniques that can be used to cluster documents. The clustering approach presented by Song et al. [5] takes into consideration document words and associated tags. Song's approach clusters documents into a predefined number of clusters nc. When finding the cluster for a new document, each of the nc clusters is considered and a cluster membership vector is generated for the new document. The labelling of a document thus has a linear time complexity $O(nc)$. In our unsupervised approach the number of clusters is defined by the data and indirectly by the number of tags w considered in document clustering. When finding the cluster for the new document, the whole tree of clusters is considered, so that a document dealing with a specialised topic is assigned to a leaf node while a document dealing with a more general, broad topic is assigned to a cluster higher up the tree. Nevertheless, the labelling of documents in our system has a time complexity of $O(log\ nc)$ as explained later. The main advantage of our

[1]http://www.bibsonomy.org/

approach is that it can find the level of specialisation of a document in a topic area and the potential set of tags for the new document reflect its level of specialisation. Moreover, the system is able to generate new tags.

The algorithm proposed here is a two part unsupervised document clustering method consisting of a divisive and an agglomerative clustering part. The aim of the algorithm is to organise documents into reasonable clusters and define a predefined number of **unpersonalised** tags, w, for the documents based on the clusters they belong to. Note that these w tags will be generated from words which appear in the word corpus of the document dataset.

In the first stage of clustering (the divisive part), documents are clustered using bisecting K-Means [6], for which different initialisation techniques were examined. Division of clusters is performed until the documents in a cluster satisfy the condition that they share at least s of their w most important words (defined by their tf-idf-score) within the cluster. The second clustering stage (the agglomerative part) takes the final set of clusters generated by the divisive part and merges them hierarchically to create a tree structure of clusters.

The document representation used for clustering is bag-of-words. Stop-words are removed and the dimensionality of the data set is reduced to n using document frequency, which proved to be an effective and cheap technique for dimensionality reduction in the experiments carried out by Yang and Pedersen [7].

Divisive Part

1. define the number of tags w required to be assigned to each document. This parameter, along with the parameter s below, indirectly determines the number of document clusters (leaf nodes in the document hierarchy)
2. for each document, find w words with the highest tf-idf score, $tfidf_d$, as the tags for that document
3. put all documents in one cluster
4. find w words as the tags on the cluster level for this cluster (see section on finding tags at cluster level)
5. check if the cluster has only documents which share at least s tags with the cluster level, $1 <= s <= w$. If yes final cluster for this recursion path was found, stop; if no proceed to split the cluster into two
6. split the cluster by finding two new cluster centroids (as explained in Initialising KMeans below) and assigning each of the documents in the cluster to one of the new centroids using cosine similarity.
7. for each of the two new clusters, perform steps 4-6.

Finding Tags at Cluster Level

Tags at the cluster level can be found by

- taking the t words with highest average tf-idf score across all documents in the cluster
- recalculating the tf-idf on cluster level $tfidf_c$, treating clusters as single documents and the words from all documents within a cluster as members of this single document

The second approach of recalculating the tf-idf values at the cluster level has two advantages. Firstly, it makes sure that the tags found for the given cluster are good for distinguishing it from other clusters. Moreover, it is interesting to observe the change in cluster tags on different levels of the clustering hierarchy. On the lowest level where clusters are smallest, the cluster tags will be very specific to the topic of the documents that are in the cluster. Higher up the tree, when documents of different specialised topics share the same cluster, the words that are common between these documents and at the same time distinguish them from documents in other clusters will be chosen as cluster tags. The tags assigned to clusters on different levels in the tree will thus create a hierarchy of topics from general to specialised.

Initialising K-Means

The standard K-Means algorithm is initialised with random points for centroids [1]. This is the cheapest with regard to cost, however the choice of the location of cluster centroids is not based on any underlying rationale. A better method is to find two points which are far apart from each other as centroids.

Our approach is to select the cluster centroids with the goal of sooner satisfying the end condition of the clustering process. The end condition in our case is that all documents assigned to a cluster share at least s of their top w words with the tags of the cluster. When faced with a cluster that has to be split, we find the mean of the documents in the cluster that do satisfy the end condition and set the centroid of the first cluster to this mean. The candidates for the centroid of the second cluster are all the documents which do not satisfy the end condition. From these, the document that is farthest away from the first centroid by cosine similarity is set as the second cluster centroid. An alternative is to select the document for which the $tfidf_d$ scores for words which are tags of the cluster are lowest.

One problem that arises when clustering is that documents which do not share any attributes with any of the two cluster centroids will have a cosine similarity of zero to both centroids and thus cannot be assigned to one of them. This is overcome by slightly pulling in the cluster centroids towards the mean of all documents (described as “shrinking” in the CURE Algorithm [3]) which eliminates values of zero for attributes.

Agglomerative Part

During the divisive step, documents are partitioned greedily and research has shown that a second pass (using an agglomerative clustering approach) across the resulting clusters (leaf nodes) can help improve the clustering quality by reversing sub-optimal splits occurring in the divisive step [8]. Hence, after the divisive step we use the leaf nodes of the tree and perform agglomerative clustering on them as described below.

1. start with clusters produced by divisive part
2. find the two clusters for which the mean vectors are closest by cosine similarity and merge them
3. repeat step 2 until a desired number of clusters r is reached or until all clusters are merged into one
4. find tags for all clusters by calculating tf-idf on cluster level ($tfidf_c$) based on the documents in the clusters

The result is a hierarchy of cluster merges, with each cluster having w tags which form a topic hierarchy from general to specialised from the root to leaf nodes.

GENERATING NEW TAGS

To generate new tags for a document, the tf-idf score in the document is calculated for each of its words and a user-defined number k of the words with highest scores are proposed as candidate tags. The target document is also assigned to a cluster within the cluster tree generated by the clustering algorithm (see section on finding the document neighbourhood) and the tags associated with the cluster are also added to the candidate list of tags for the target document. Note that some of these words may not appear in the target document and are assigned to the document on the basis of its similarity with other members of the same cluster characterised by these words. The score assigned to a tag is the average between the tf-idf scores on document level ($tfidf_d$) and cluster level ($tfidf_c$).

$$wscore(word) = \frac{tfidf_d(word) + tfidf_c(word)}{2}$$

FINDING PERSONALISED TAGS

The initial set of potential tags for the new document-user pair (u_a, doc_{new}) consists of all tags assigned to documents in the δ-neighbourhood of the new document by any user (see section on finding document neighbourhood below). For each of these tags, a weight is calculated which measures the suitability of the tag based on the similarity scores of the documents that are related to the tag and the similarity scores of the users who assigned that tag. The set of users that we are interested in consist of the users who assigned tags to one or several documents in the neighbourhood of the new document.

The t tags with the highest suitability value are finally recommended to the active user. The formula for calculating the suitability of each tag is as follows.

$$suit(u_a, tag_i) = \frac{1}{|posts_{tag_i}|} \sum_i dsim(doc_{new}, doc_i)^{\beta} \cdot usim(u_a, u_b)^{(1-\beta)}$$

where $|posts_{tag_i}|$ is the number of posts for tag_i, $dsim$ and $usim$ are the document and user similarity measures (described below) and $\beta \leq 1$ is the weight given to document similarity compared to user similarity.

If there are not enough tags in the resulting tag set with a *suitability score* above a user-defined threshold, new tags are generated as described in the section above.

Finding the Document Neighbourhood

As described previously the document clustering algorithm results in a cluster hierarchy where each parent node has two child nodes. To select a set of similar documents for a new document from the cluster hierarchy, the tree is traversed from the top down and the new document is assigned to one of the two clusters at each branch split. Thus a path through the tree is obtained for the new document in *O(log nc)* time. From this path, the cluster with the highest cosine similarity is assigned to the document as its cluster and the documents in the cluster added to the set of similar documents D_{sim}. For each of the documents in D_{sim}, the similarity to the new document $dsim(doc_{new}, doc_i)$ is calculated by cosine similarity. If the document similarity is above a threshold δ for each of the documents in D_{sim}, further documents are added to D_{sim} by travelling up the tree from the cluster and adding all documents in the clusters on the path until a cluster containing a document with $dsim(doc_{new}, doc_i) < \delta$ is encountered.

User Similarity

The similarity between two users can be calculated in three different ways.

The *document set similarity* considers how many document are shared between the two users and is calculated using the Jaccard co-efficient.

$$sim_D(u_a, u_b) = \frac{|D_a \cap D_b|}{|D_a \cup D_b|}$$

where D_a and D_b are the document sets of the two users.

The *vocabulary similarity* measures the overlap in the two users' vocabulary sets and is given by

$$sim_V(u_a, u_b) = \frac{|V_a \cap V_b|}{|V_a \cup V_b|}$$

where V_a and V_b are the sets of tags used by the two users.

The *tagging similarity* considers whether the two users assigned the same tags to the same documents and is calculated as follows.

$$sim_T(u_a, u_b) = \frac{1}{|D_a \cup D_b|} \sum_{i=1}^{|T_{d_i a} \cap T_{d_i b}|} \frac{|T_{d_i a} \cap T_{d_i b}|}{|T_{d_i a} \cup T_{d_i b}|}$$

where $T_{d_i a}$ and $T_{d_i b}$ are the tag sets of the two users for document *i*. The *tagging similarity* measure incorporates *document set similarity* through the division by $|D_a \cup D_b|$. This ensures that the *tagging similarity* reflects not only the documents for which two users have common tags but also takes into account the number of documents for which the two users do not have common tags.

The *tagging similarity* measure is the most valuable since it takes into consideration both, document and tag sets and the relationship between these, while the other two measures focus on only one of these aspects. However, due to sparsity of the data the two users often do not have any shared document-tag pairs which means the *tagging similarity* is zero. Hence, the *tagging similarity* is combined with *vocabulary similarity* which results in zero less frequently. A *tagging similarity* of zero and a non-zero *vocabulary similarity* indicates that the users share some tags but have assigned them to different documents, thus the tags used by user u_a might be suitable for u_b. The combined user similarity score is given by

$$usim(u_a, u_b) = \alpha sim_T(u_a, u_b) + (1 - \alpha) sim_V(u_a, u_b)$$

where $\alpha \leq 1$ specifies the weight given to the *tagging similarity* compared to the *vocabulary similarity.*

EVALUATION

At the time of writing, the implementation of the suggested system is not fully completed and an extensive evaluation to determine the best thresholds and parameters still has to be done. An initial evaluation was carried out on a dataset from the social bookmarking website Bibsonomy. The dataset consists of a variety of websites and academic papers that were bookmarked on Bibsonomy and the tags that were assigned to these documents by the users. The post-core at level 5 was used so each user, document and tag appeared at least five times in the data, resulting in 7538 posts (user, document pairs) containing 244 users, 953 documents and 811 tags. Similarly to [4], leave-one-out cross validation was used to evaluate the algorithm. Each document was represented using a vector space model, where the dimensions were the 1000 words with largest document frequency in the corpus. Five tags were recommended for each test post and the performance measure was standard F1. Since the test dataset contained only users and tags already known to the system no new tags were generated.

The most significant parameters of our recommender system are thresholds on document and user similarity. These thresholds allow us to consider only documents and users with a similarity score higher than the set value when calculating the scores for the potential tags. If we set the threshold for document similarity to 1.0, only tags assigned to the new document in other posts (by other users) are considered. This is referred to as the set of *popular tags* in other systems such as the current del.icio.us tag recommender. While in other systems the score for each tag is given simply by the number of posts it appears in, our system also weights each post's importance by considering the user similarity to the active user, hence personalising the tag recommendation. If we set the threshold for user similarity to 1.0, only the tags from the active user's other posts are considered. This is referred to as the set of *personal tags*. For this scenario each tag's score is influenced by the similarity of the new document to the documents in the user's posts.

When tested on their own, the approach of *popular tags* produced better recommendations (F1 of 0.25) than the *personal tags* (F1 of 0.13). However, the best results were achieved through generating two sets of tags, one from each approach and combining them to give a final recommendation set. The two sets were combined by a weighted sum of the two scores for each tag in order to increase the overall score of tags which appeared in both sets. The final tag scores were then normalised. By recommending tags from the combined set of popular and personal tags, the F1 was increased to 0.35. The results of the combined approach when recommending different sizes of tag sets is shown in Figure 1 below. To additionally evaluate our method of generating new tags, we set the threshold on the tag score to 1.0 in another test run so that all recommended tags were generated from the content of the documents. The resulting F1 was 0.12 when generating five tags for each post.

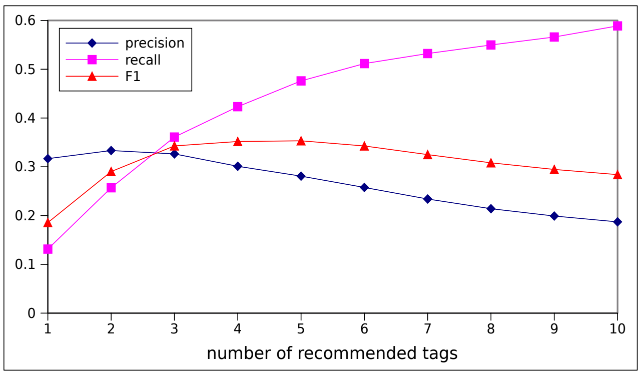


Figure 1: Results on Bibsonomy dataset at p-core 5

CONCLUSION AND FUTURE WORK

In this paper we suggested a novel approach to personalised tag recommendation. By creating a cluster tree of documents we generate a topic hierarchy from general to specialised, and determine the level of specialisation of a new document. The recommended tags reflect the generality of the new document. Through identifying a set of users similar to the active user and measuring their similarity, we are able to suggest personalised tags. If the required number of tags cannot be found in the existing tag set, new tags are generated from the vocabulary of the document corpus.

We plan to perform an extensive evaluation of the suggested system on different datasets in order to find the best values for the many parameters such as the weights given to the different similarity measures and thresholds for tag suitability. We will also evaluate our methods ability to address the new-user/new-document/new-tag instantiation of the new-item problem well known in Recommender literature. To further improve the tag recommender, we plan to implement and evaluate different clustering and cluster representation techniques such as those used by CURE [3].

In the future, we plan to investigate different approaches to tag recommendation, taxonomy generation and dimensionality reduction. Techniques for feature extraction such as finding synonym sets and topic models for documents are very interesting and have the potential increase the performance of any tag recommendation or classification system.

REFERENCES

[1] P. S. Bradley and U. M. Fayyad. Refining initial points for K-Means clustering. In *Proc. 15th International Conf. on Machine Learning*, pages 91–99, 1998.

[2] J. Gemmell, T. Schimoler, M. Ramezani, and B. Mobasher. Adapting k-nearest neighbor for tag recommendation in folksonomies. In *Proc. 7th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems*, 2009.

[3] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *Proc. 1998 ACM SIGMOD International Conference on Management of Data*, pages 73–84, 1998.

[4] R. Jaeschke, L. B. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in folksonomies. In *Proc. 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2007.

[5] Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W.-C. Lee, and C. L. Giles. Real-time automatic tag recommendation. In *Proc. 31st international ACM SIGIR conference on Research and development in information retrieval*, pages 515–522, 2008.

[6] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. Technical Report 00-034, University of Minnesota, 2000.

[7] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proc. ICML-97, 14th International Conference on Machine Learning*, pages 412–420, 1997.

[8] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *Proc. ACM SIGMOD '96: Int. Conf. on Management of Data*, pages 103–114, 1996.

A hybrid PLSA approach for warmer cold start in folksonomy recommendation

Alan Said   Robert Wetzker   Winfried Umbrath   Leonhard Hennig
DAI Labor
Technische Universität Berlin
{alan.said, robert.wetzker, winfried.umbrath, leonhard.hennig}@dai-labor.de

ABSTRACT

We investigate the problem of item recommendation during the first months of the collaborative tagging community *CiteULike*. CiteULike is a so-called folksonomy where users have the possibility to organize publications through annotations - tags. Making reliable recommendations during the initial phase of a folksonomy is a difficult task, since information about user preferences is meager. In order to improve recommendation results during this *cold start* period, we present a probabilistic approach to item recommendation. Our model extends previously proposed models such as probabilistic latent semantic analysis (PLSA) by merging both user-item as well as item-tag observations into a unified representation. We find that bringing tags into play reduces the risk of overfitting and increases overall recommendation quality. Experiments show that our approach outperforms other types of recommenders.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Information Search and Retrieval

Keywords

recommendations, folksonomies, CiteULike, cold start, PLSA

1. INTRODUCTION

Recommender systems have become an integral part of almost every Web 2.0 site, allowing users to easily discover relevant content. Many social tagging communities, such as *CiteULike*[1], *Delicious*[2] and *Bibsonomy*[3], use recommendation techniques as part of their service. These communities, generally referred to as folksonomies, give users the possibility to annotate items with freely chosen keywords (tags) for easier content retrieval at later points in time [14]. Most of these folksonomy systems recommend suitable tags when a user tags a new item.

In this paper we consider a problem all new folksonomy websites and services offering recommendations encounter – the cold start phase, during which recommendations have to be made based on very little historical data. During this phase, the similarities between items are hard to calculate as the user-item graph is sparsely, if at all, connected. However, when turning to tags, we find a higher connectivity. Our objective is to increase the quality of item recommendations during this cold start phase by utilizing item-tag co-occurrences in conjunction with their user-item co-occurrence counterparts. By doing so, we improve standard collaborative filtering models by considering user-given annotations, i.e. tags.

Probabilistic latent semantic analysis (PLSA), as introduced by Hofmann in [8], is known for improving recommendation quality in different settings [1]. PLSA assumes a lower dimensional latent topic distribution of the observed co-occurrences. These latent distributions group similar items together. We use an extended model of the hybrid PLSA recommender described in [15]. Our recommender derives the latent topic distribution from user-item and item-tag co-occurrences in parallel. Furthermore we extend this model to cope with known issues related to overfitting.

We perform our evaluation on a subset of the CiteULike dataset. CiteULike is a service allowing users to share, organize and store scholarly papers by assigning tags. Daily snapshots of the CiteULike dataset are made available through the official website[4]. Since we are only interested in the startup period of recommendation systems we carry out our experiments on the first 12 months of the available data, starting on day one - when the first document was tagged on November 4, 2004. For comparison we also present how our model performs after 24 months, which corresponds to a point in time when CiteULike had become an established service. Our results clearly show that our Hybrid PLSA (HyPLSA) model produces higher quality recommendations during the cold start period compared to other models. Additionally we find that our approach performs well when the dataset has grown significantly in size, although these improvements are not as significant as the ones during the cold start period though.

[1]`http://www.citeulike.org/`
[2]`http://delicious.com/`
[3]`http://www.bibsonomy.org/`

[4]`http://www.citeulike.org/faq/data.adp`

1.1 Related Work

Recommender systems can be divided into three main categories; collaborative filtering-based, content-based, and so-called hybrid systems which combine both. Collaborative filtering approaches base their recommendations solely on co-occurrence observations between users and items. Content-based ones, as the name suggests, derive their similarities based on content, i.e. term distributions etc. Hybrid systems utilize data from both of these models. In folksonomies tags tend to reflect the content of the tagged item [2], thus even if we do not consider the actual item content itself, we group tag-based recommender system together with the content-based ones.

The authors of [15] present a hybrid approach to item recommendation in collaborative tagging communities based on PLSA in which they exploit tags to improve recommendations on very large datasets.

Since the introduction of PLSA by Hoffman in [8] it has shown to perform very well in a wide area of topics, among others it continues to outperform multiple other recommendation and decomposition algorithms [1,16]. A drawback of PLSA is that it does not necessary converge to the global optimum [5]. One way to overcome any effects that may arise from this is presented in [3,6] where the authors show that multiple training cycles for the same test/train splits provide for more robust results.

Past research within the context of recommendation in folksonomies has, until recently, been focused on tag recommendation [7,13]. We apply our extended HyPLSA approach on the task of item recommendation instead.

Another successful approach to recommendation within folksonomies is the FolkRank algorithm introduced by the authors of [10], we use this algorithm as a comparison to our approach.

The remainder of this paper is structured as follows. In Section 2 we present the algorithms utilized in our experiments followed by a description of our tests, dataset and experimental setup in Section 3. We present our results in Section 4 and draw final conclusions in Section 5.

2. ALGORITHMS

In the following section we describe our HyPLSA approach which is an extended version of the one presented by the authors of [15].

2.1 Model fusion using PLSA – HyPLSA

Hotho et al. [9] describe a folksonomy as tripartite graph in which the vertex set is partitioned into three disjoint sets of users $U = \{u_1, ..., u_l\}$, tags $T = \{t_1, ..., t_n\}$ and items $I = \{i_1, ..., i_m\}$. In [15] Wetzker et al. simplify this model into two bi-partite models; the collaborative filtering model IU built from the item user co-occurrence counts $f(i,u)$, and the annotation-based model IT analogously derived from the co-occurrence total between items and tags $f(i,t)$. In the case of social bookmarking IU becomes a binary matrix ($f(i,u) \in \{0,1\}$) since each user can bookmark a given web resource one time only. Given this model, we want to recommend the most interesting new items from I to user u_l given his or her item history.

The PLSA aspect model associates the co-occurrence of observations with a hidden topic variable $Z = \{z_1, \ldots, z_k\}$. In the context of collaborative filtering, an observation corresponds to the bookmarking of an item by a user and all observations are given by the co-occurrence matrix IU. Users and items are assumed independent given the topic variable Z. When applying the aspect model, the probability of an item that has been bookmarked by a given user can be computed by summing over all latent variables Z:

$$P(i_m|u_l) = \sum_k P(i_m|z_k)P(z_k|u_l) \tag{1}$$

For the annotation-based scenario we assume the set of hidden topics to be the same as in the item tag co-occurrence observations given by IT. In compliance with (1), the conditional probability between tags and items can be written as:

$$P(i_m|t_n) = \sum_k P(i_m|z_k)P(z_k|t_n). \tag{2}$$

Following the procedure in [4], we can now merge both models based on the common factor $P(i_m|z_k)$ by maximizing the log-likelihood function:

$$\begin{aligned} L &= \sum_m \Bigg[\alpha \sum_l f(i_m, u_l) \log P(i_m|u_l) \\ &\quad + (1-\alpha) \sum_n f(i_m, t_n) \log P(i_m|t_n) \Bigg], \end{aligned} \tag{3}$$

where α is a predefined weight for the leverage of each two-mode model. Using the expectation-maximization (EM) algorithm [4] we subsequently perform maximum likelihood parameter estimation for the aspect model. During the expectation (E) step we begin with calculating the posterior probabilities:

$$P(z_k|u_l, i_m) = \frac{P(i_m|z_k)P(z_k|u_l)}{P(i_m|u_l)}$$
$$P(z_k|t_n, i_m) = \frac{P(i_m|z_k)P(z_k|t_n)}{P(i_m|t_n)},$$

and then re-estimate parameters in the maximization (M) step according to:

$$P(z_k|u_l) \propto \sum_m f(u_l, i_m)P(z_k|u_l, i_m) \tag{4}$$

$$P(z_k|t_n) \propto \sum_m f(t_n, i_m)P(z_k|t_n, i_m) \tag{5}$$

$$\begin{aligned} P(i_m|z_k) &\propto \alpha \sum_l f(u_l, i_m)P(z_k|u_l, i_m) \\ &\quad + (1-\alpha) \sum_n f(t_n, i_m)P(z_k|t_n, i_m) \end{aligned} \tag{6}$$

Based on the iterative computation of the above E and M steps, the EM algorithm monotonically increases the likelihood of the combined model on the observed data. Using the α parameter, our new model can easily be reduced to a collaborative filtering, or annotation-based model, simply by setting α to 1.0 or 0.0 respectively.

Because of the random initialization of the EM algorithm utilized by PLSA, we employ an averaging approach to reduce any effects possibly caused by local maximum optimizations. Thus, following Equation (1), we repeat Equations (4) to (6) n times for every recommendation and average the probabilities obtained from Equation (1). Our contribution to the model is presented in Equation (7), where the final, averaged, probability is given.

$$\bar{P}(i_m|u_l) = \frac{\sum_n P_n(i_m|u_l)}{n} \quad (7)$$

We can now recommend items to a user u_l weighted by the probability $P(i_m|u_l)$ from Equation (7). For items already bookmarked by the user in the training data we set this weight to 0, thus they are appended to the end of the recommended item list.

3. EXPERIMENTS

We conduct our experiments on the CiteULike dataset, these experiments are described next.

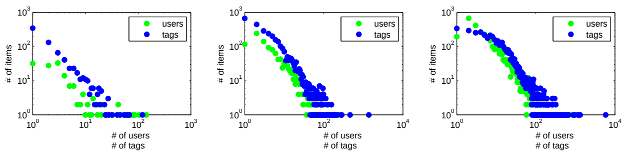
3.1 Dataset

The CiteULike bookmarking service provides daily snapshots of their data for research purposes. At the time of writing the overall dataset consisted of roughly 53 months of data. As noted earlier we are only interested in the initial phase of the service and therefore limit our analysis to the first 24 months, focusing on the first 12.

CiteULike was chosen as the experimental dataset because it is a well known real-world folksonomy and has been experimented on by numerous others [11,12,17].

We started by removing all users who had bookmarked less than 20 items as well as items and tags that occurred only once. We then created monthly snapshots where each of the snapshots accumulated all previous tagging events. By doing this we were able to simulate a growing folksonomy over time.

Figures 1(a), 1(c), 1(b) and 2 show some characteristics of our dataset.



(a) at 3 months. (b) at 12 months. (c) at 24 months.

Figure 1: Users and tags plotted against the number of items they are connected to.

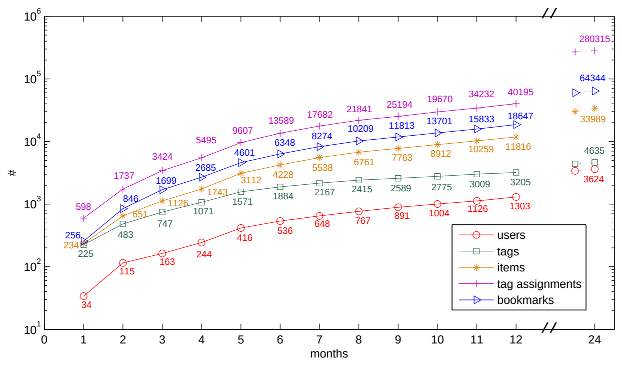


Figure 2: Accumulated number of items, tags, users, tag assignments and bookmarks of our data per month

3.2 Experimental setup

To create test and training sets for our algorithms, we split each monthly snapshot in two. For all users who had bookmarked at least 5 items in the current snapshot, we selected 80% of their items as the training set. The remaining items were consequently used for testing. We then trained all recommender types on the training sets and evaluated their performance on the test sets. The relatively small size of the dataset allowed us to optimize parameters through a brute force approach. Evaluation measures were averaged over all users in 10 independent test runs.

4. RESULTS

We evaluate the performance of each recommender with the well known and widely used precision at 10 measure (Prec@10). Other evaluation measures, such as *mean average precision* (MAP), *area under curve* (AUC)and *F1 score* (F1), showed similar results.

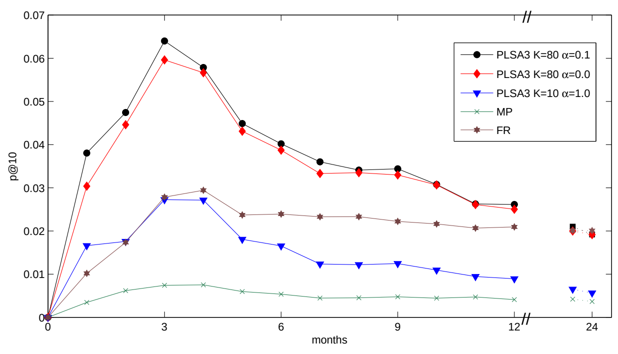


Figure 3: Prec@10 values for the item recommendation task on the *CiteULike* dataset plotted per month. The number of latent topics (k) is set to 80 for the purely annotation-based PLSA recommender ($\alpha = 0.0$) and to 10 for the purely collaborative version ($\alpha = 1.0$). The MP and FR lines represents the performances of a *most-popular* baseline classifier and the *FolkRank* recommender presented by Hotho et al. [10]. The results of the combined HyPLSA approach are seen in the topmost line, the parameters α and k where set to 0.1 and 80 respectively.

Figure 3 shows the Prec@10 values for the HyPLSA recommender obtained with an optimal parameter setting ($alpha = 0.1$, $k = 80$), the purely collaborative filtering-based PLSA recommender ($\alpha = 1$, $k = 10$), the purely annotation-based PLSA recommender ($alpha = 0$, $k = 80$), the baseline most-popular recommender and the FolkRank recommender. The figure shows a significant improvement when using the HyPLSA recommender, especially in the early months of the CiteULike. We also see that as the dataset grows, and the number of possible items to recommend increases, the precision values decrease. Nevertheless, the HyPLSA approach delivers significantly better results than the other evaluated approaches.

Figure 4 shows a figure similar to Figure 3, this time with Prec@10 values plotted against the number of items in the dataset, confirming the observation made earlier.

In Figure 5 we present the relative improvements in precision of the HyPLSA approach plotted against the other ones explored in this paper. The reason for these results

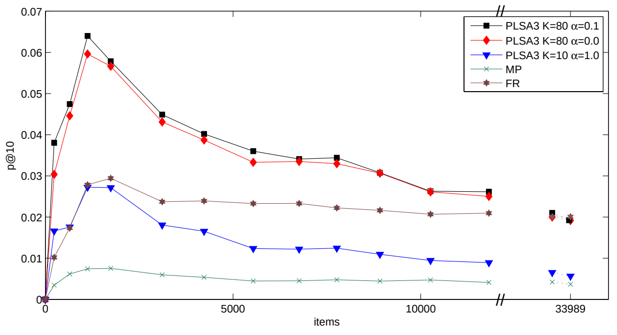


Figure 4: Prec@10 values for the same scenario and value as in Figure 3 plotted against the number of items in the dataset.

can be traced back to Figure 1(a) where we clearly see the differences in density in the user domain when comparing Figure 1(a) to Figures 1(b) and 1(c). At best our approach improves precision values roughly tenfold compared to the baseline MP recommender and twice as well as the FR recommender. As expected the improvement is highest in the first couple of months and slowly decreases (for MP and FR) or stabilizes (for CF) as the dataset grows. Comparing to the tag-based approach, the improvement is not as distinct as in the case with other recommenders. Relative improvements are in the order of $2-25\%$ during first seven months, decreasing in the long run.

These results confirm the notion that, for a small dataset, the number of user-item co-occurrences is too low to allow a collaborative filtering recommender to make good predictions. Tags and tag-item co-occurrences, on the other hand, provide higher item-item similarities as tags are more abundant and contain contextual information about the items. Therefore tags provide for recommendations on a finer level of granularity.

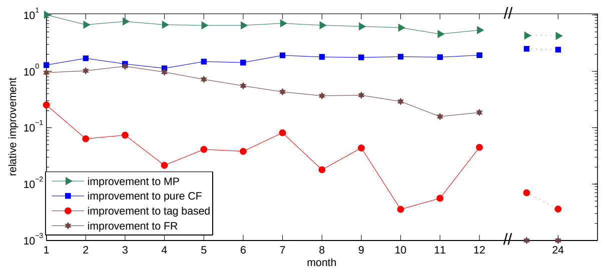


Figure 5: The relative improvement of the proposed HyPLSA recommender compared to the other explored ones. The higher the line, the bigger the improvement.

5. CONCLUSIONS

We have shown that tags improve the quality of item recommendation during the cold start period of a folksonomy. This is due to the fact that tags offer denser, more detailed item information than usage patterns do.

Furthermore we have presented a hybrid probabilistic approach that combines user and tag similarities in order to boost recommendation quality. The recommendation quality improvement created by using this approach peaks during the cold start period, although the approach continues to provide for better recommendations as the size of dataset increases. We believe that the reason for the relative improvement being higher in the beginning can be traced back to the pattern seen in Figures 1(a) to 1(c) where we initially see a very much higher density in tag usage compared to usage patterns. As the tag usage pattern density becomes more and more similar to the tag density the recommendation results of all PLSA (tag, CF and HyPLSA) approaches become similar.

6. REFERENCES

[1] J. Arenas-García, A. Meng, K. B. Petersen, T. L. Schiøler, L. K. Hansen, and J. Larsen, ‘Unveiling music structure via PLSA similarity fusion’, in *Proc. of MLSP'07*, (2007).

[2] K. Bischoff, C. S. Firan, W. Nejdl, and R. Paiu, ‘Can all tags be used for search?’, in *CIKM '08*, (2008).

[3] T. Brants, F. Chen, and I. Tsochantaridis, ‘Topic-based document segmentation with probabilistic latent semantic analysis’, in *Proc. of CIKM '02*, (2002).

[4] D. A. Cohn and T. Hofmann, ‘The missing link - a probabilistic model of document content and hypertext connectivity’, in *NIPS*, pp. 430–436, (2000).

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin, ‘Maximum likelihood from incomplete data via the em algorithm’, *Journal of the RSS. Series B*, **39**, (1977).

[6] L. Hennig, ‘Topic-based multi-document summarization with probabilistic latent semantic analysis’, in *RANLP '09*, (Sep. 2009).

[7] P. Heymann, D. Ramage, and H. Garcia-Molina, ‘Social tag prediction’, in *Proc. of SIGIR '08*, (2008).

[8] T. Hofmann, ‘Probabilistic latent semantic analysis’, in *Proc. of UAI '99*, (1999).

[9] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme, ‘Information retrieval in folksonomies: Search and ranking’, in *ESWC*, pp. 411–426, (2006).

[10] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme, ‘FolkRank: a ranking algorithm for folksonomies’, in *Proc. FGIR 2006*, (2006).

[11] S. Noël and R. Beale, ‘Sharing vocabularies: tag usage in citeulike’, in *Proc. of BCS-HCI '08*, (2008).

[12] S. Oldenburg, M. Garbe, and C. Cap, ‘Similarity cross-analysis of tag / co-tag spaces in social classification systems’, in *Proc. of SSM '08*, (2008).

[13] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, ‘Tag recommendations based on tensor dimensionality reduction’, in *Proc. of RecSys '08*, pp. 43–50, (2008).

[14] T. Vander Wal. Folksonomy coinage and definition. `http://www.vanderwal.net/folksonomy.html` (retrieved on June 10, 2009), February 2007.

[15] R. Wetzker, W. Umbrath, and A. Said, ‘A hybrid approach to item recommendation in folksonomies’, in *Proc. of WSDM - ESAIR '09*, pp. 25–29, (2009).

[16] G.-R. Xue, W. Dai, Q. Yang, and Y. Yu, ‘Topic-bridged plsa for cross-domain text classification’, in *Proc. of SIGIR '08*, (2008).

[17] V. Zanardi and L. Capra, ‘Social ranking: uncovering relevant content using tag-based recommender systems’, in *Proc. of RecSys '08*, pp. 51–58, (2008).

Cobot: Real Time Multi User Conversational Search and Recommendations

Saurav Sahay
College of Computing
Georgia Tech
ssahay@cc.gatech.edu

Anushree Venkatesh
College of Computing
Georgia Tech
avenkatesh6@gatech.edu

Ashwin Ram
College of Computing
Georgia Tech
ashwin@cc.gatech.edu

ABSTRACT

Cobot is a new intelligent agent platform that connects users through real-time and off-line conversations about their health and medical issues. Intelligent web based information agents (conversational/community bots) participate in each conversation providing highly-relevant real-time informational recommendations and connecting people with relevant conversations and other community members. Cobot provides an innovative approach to facilitate easier information access allowing users to exchange information through a natural language conversational approach. Conversational Search(CS) is an interactive and collaborative information finding interaction. The participants in this interaction engage in social conversations aided with an intelligent information agent (Cobot) that provides contextually relevant factual, web search and social search recommendations. Cobot aims to help users make faster and more informed search and discovery. It also helps the agent learn about conversations with interactions and social feedback to make better recommendations. Cobot leverages the social discovery process by integrating web information retrieval along with the social interactions and recommendations.

Categories and Subject Descriptors

H.5.0 [**Information Systems Applications**]: General; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.7 [**Artificial Intelligence**]: Natural Language Processing

General Terms

Design, Human Factors, Algorithms

Keywords

Real time Collaborative Information Access, Social Search, Contextual Collaborative Filtering, Conversational Search

1. INTRODUCTION

This paper introduces a novel Conversational Search and Recommendation system that involves finding relevant information based on social interactions and feedback along with augmented agent based recommendations. People in social groups can provide solutions (answers to questions)[3], pointers to databases or other people (meta-knowledge)[3][6], validation and legitimation of ideas[3][4], can serve as memory aids[7] and help with problem reformulation[3]. "Guided participation"[11] is a process in which people co-construct knowledge in concert with peers in their community[12]. Information seeking is mostly a solitary activity on the web today. Some recent work on collaborative search reports several interesting findings and the potential of this technology for better information access.[5][2][1][9]

We are building a system called Cobot[1] to address some of these challenges. Cobot introduces a conversational environment that provides social search through conversations integrated with intelligent semantic meta-search from the web. Users want to simplify their experience when performing an information finding task. Conversational Search is about letting users collaboratively search and find in natural language, leaving the task of user intent comprehension on the system. The participating agent interacts with users proving recommendations that the users can accept, reject, like, dislike or suggest.

2. SYSTEM DESCRIPTION

Cobot is an intelligent agent platform that connects users through real-time and offline conversations. Cobot lives in a community, has a limited understanding of domains through ontologies and brings relevant information to the users by participating in the conversations. Cobot's 'conversation engine' monitors user conversations with other users in the community and provides/receives recommendations (links and snippets) based on the conversation to the participants. Cobot's 'community engine' models conversations to capture user-user and user-information interactions.

Design Goals

1. Near real time conversational agent
2. Personalized as well as generic recommendations
3. Agent learns with interaction
4. Uses a structured internal organization of content

[1]We use the term Cobot for Cobot system as well as Cobot agent interchangeably

5. Dynamically connects conversations to the right set of people for participation

6. Helps the user talk about health issues. (Real time conversations)

7. In the real time conversational process, provides recommendations. ('who to talk to', 'what to look at')

In the following sections, we briefly describe the features of the Cobot system.

2.1 Real time Conversational Search

Cobot provides a conversational interface that combines semantic language understanding and real time collaborative environment for information retrieval with contextually relevant recommendations. Cobot helps people find information faster with the aim for finding useful responses for completing the search intent. The conversational interface allows for much more interactivity than one-shot search style interfaces, which aids usability and improves intent understanding. For example, Cobot recommends links and snippets from relevant articles on the internet. It also makes social recommendations to connect contextually relevant users to the conversation.

The approach we have taken to address CS problems is by developing dynamic data structures that model it. We call this structure the "Socio-Semantic Model" - these conversation nets maintain in memory models of the conversation, participants, participants' immediate social connections, concepts, relationships and information flow.

2.2 Socio-Semantic Model

The Socio-Semantic Conversation Model is a dynamic memory data structure based on principles of experience based agent architecture.[10] It supports interleaved retrieval of information by applying different memory retrieval algorithms. The model maintains the user's social graph, the conversation graph with the extracted semantic net for the conversation.

Some essential properties of the model are as follows:

- The model is socially aware of the participant and his social network's availability (to aid with Cohort Matching)

- The model provides bi-directional recommendation and feedback. (Both agent and the participant can add recommendations)

- The model understands limited domain terminology and is able to find semantic relationships amongst concepts extracted from conversations.

- The model is aware of user's profile (such as interests and ratings) for the agent to be able to use that information.

The Socio-Semantic Model aims to provide storage and memory based retrieval for dynamic representation, update and reuse of users' knowledge and experiences. Figure 1 depicts the user-centric domain information modeling approach to jointly model the information context from users' perspective.

2.3 Aggregated Web Search

Identifying relevant documents for a particular user's need without extensive search, in conversational manner is the key objective for precise search. The right search queries need to be figured out with situation assessment from the conversational snippets. It is not desirable to return dozens or hundreds of remotely relevant results, even if some of them will be highly relevant. The aim is to retrieve successive recommendations that try to address the search problem precisely. Cobot uses different shallow semantic parsing techniques for operationalizing a user's intent into computational form, dispatching to multiple, heterogeneous services, gathering and integrating results, and presenting them back to the user as a set of solutions to their request.

2.4 Real time matching of participants to conversations

Communities are made up of users who are grouped by different information needs into dynamic cohorts. These online communities, through effective sharing and collaboration, increase the utility of systems and help solve individual problems more effectively. Cobot allows for connecting two or more individuals to an online conversation based on the topic and context of conversation, mutual interests, and what they want to talk about at that time. The system allows any individual to find/join that conversation.

2.5 Socio-Semantic Collaborative Filtering

Filtering and recommendation are crucial in collaborative systems enabling users to navigate an ever-growing deluge of information more effectively. Cobot's recommendation engine delivers quality information delivered through filters achieved from semantic and contextual understanding of text along with captured users' interests. It uses various personalization techniques such as collaborative filtering on conversations and other entities in context. Natural language processing techniques are used to enhance the content based recommendations.[8]

3. SYSTEM ARCHITECTURE

Figure 2 depicts the high level architecture of the Cobot system. The Conversational Agent uses different modules for conversation analysis, search and recommendation and maintains a short-term conversation memory for each conversation. The socio-semantic model/net is analogous to the agent's long term memory model where it stores all processed information about users, conversations, activities and content descriptors.

4. SYSTEM PROTOTYPE

Figure 3 shows one screenshot of the initial system prototype which is work in progress. This prototype is designed for health related searches by incorporating medical ontologies. Users actively engage in conversations by multi-user chat, rating or adding recommendations. The agent monitors the environment to build user interaction models and to improve search relevance.

5. CONCLUSION

This paper proposes a collaborative system for conversational search and recommendations. We are hypothesizing

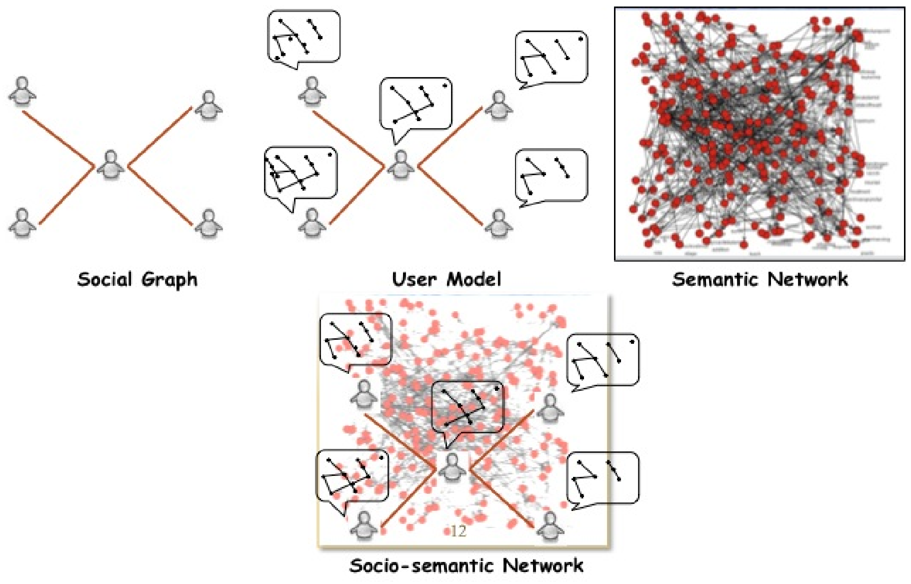


Figure 1: Socio-Semantic Net

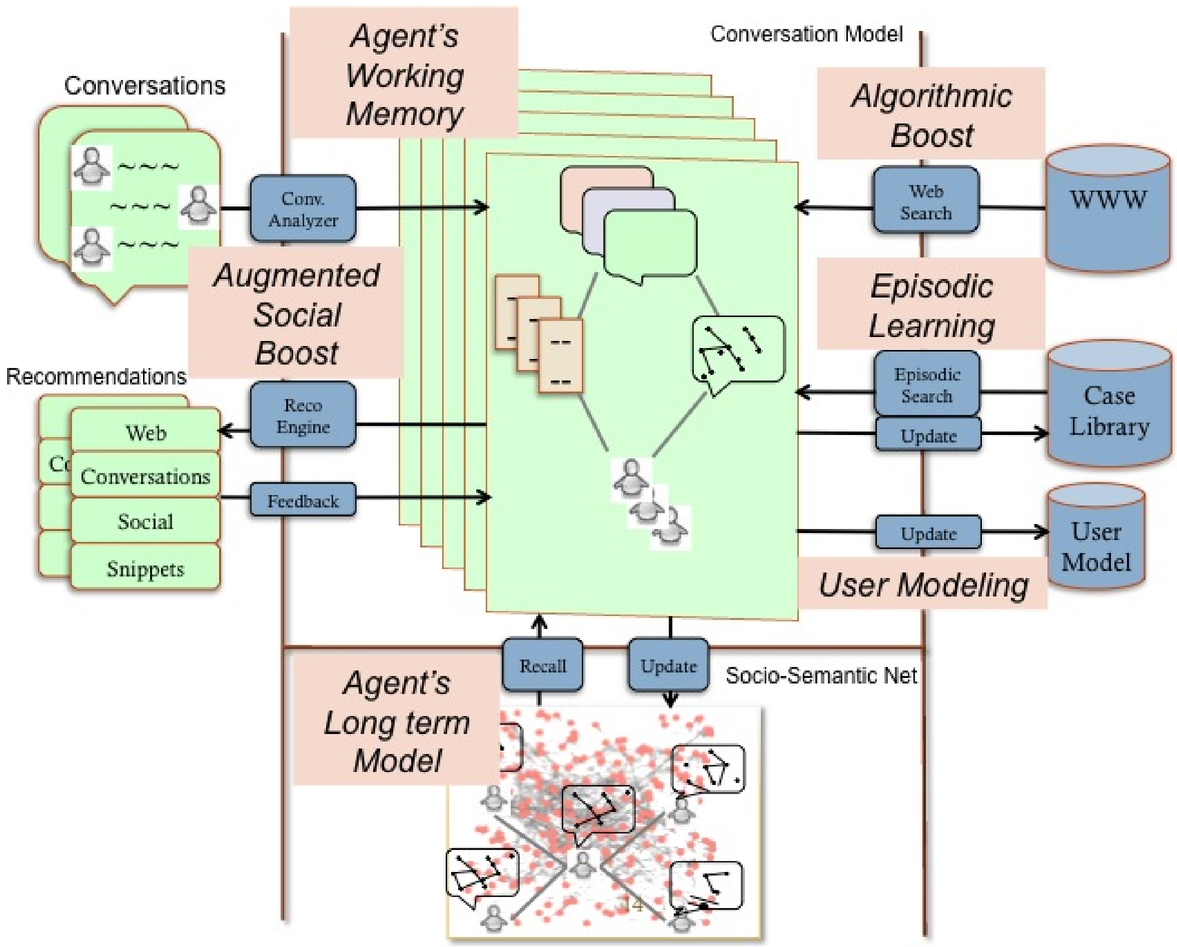
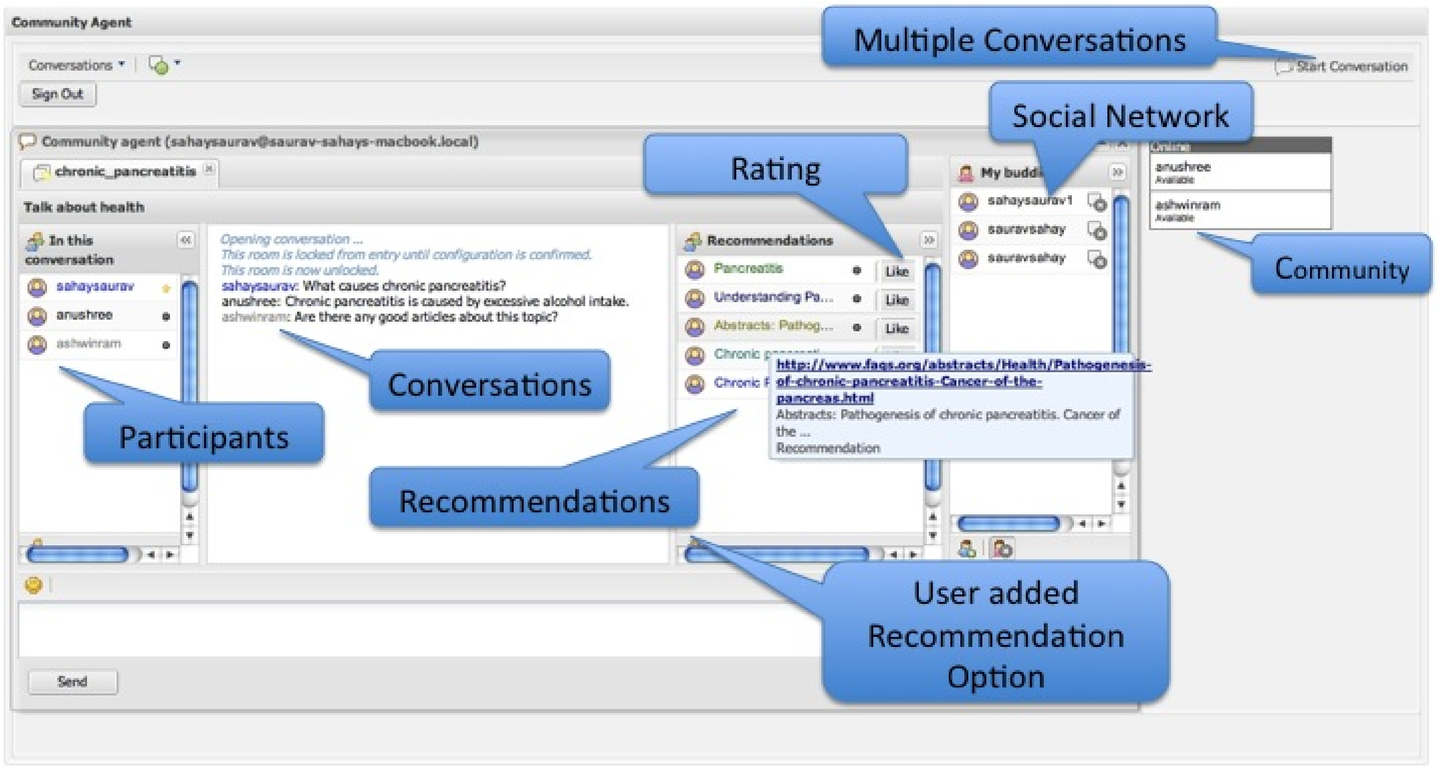


Figure 2: System Architecture



Opening conversation ...
This room is locked from entry until configuration is confirmed.
This room is now unlocked.
sahaysaurav: What causes chronic pancreatitis?
anushree: Chronic pancreatitis is caused by excessive alcohol intake.
ashwinram: Are there any good articles about this topic?

http://www.faqs.org/abstracts/Health/Pathogenesis-of-chronic-pancreatitis-Cancer-of-the-pancreas.html
Abstracts: Pathogenesis of chronic pancreatitis. Cancer of the ...
Recommendation

Figure 3: Prototype Interface

that such a Conversational Search system is more usable for information access as compared to a solitary web search experience. We briefly describe the design goals and features involved in construction of the Cobot system. Socio-Semantic Conversation Modeling using Experience-based Agency is a unified approach for addressing Conversational Search problem. The dynamic and self configuring memory structures and the semantic net details enable memory retrieval from the storage. Automatic Cohort Matching based on Conversations and User Profiles incorporate a methodology to dynamically pull users for conversations. Unlike users themselves having to find relevant conversations, the conversations find the users using this approach.

6. REFERENCES

[1] S. Amershi and M. R. Morris. Cosearch: a system for co-located collaborative web search. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1647–1656, New York, NY, USA, 2008. ACM.

[2] O. Boydell and B. Smyth. Enhancing case-based, collaborative web search. *Lecture Notes in Computer Science*, 4626:329, 2007.

[3] R. Cross, R. E. Rice, and A. Parker. Information seeking in social context: structural influences and receipt of information benefits. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 31(4):438–448, 2001.

[4] B. M. Evans and E. H. Chi. Towards a model of understanding social search. In *CSCW '08: Proceedings of the ACM 2008 conference on Computer supported cooperative work*, pages 485–494, New York, NY, USA, 2008. ACM.

[5] D. Feng, E. Shaw, J. Kim, and E. Hovy. An intelligent discussion-bot for answering student queries in threaded discussions. In *Proceedings of the 11th international conference on Intelligent user interfaces*, pages 171–177. ACM New York, NY, USA, 2006.

[6] E. A. Fox, D. Hix, L. T. Nowell, D. J. Brueni, D. Rao, W. C. Wake, and L. S. Heath. Users, user interfaces, and objects: Envision, a digital library. *J. Am. Soc. Inf. Sci.*, 44(8):480–491, 1993.

[7] I. Karasavvidis. Distributed Cognition and Educational Practice. *Journal of Interactive Learning Research*, pages 11–29, 2002.

[8] P. Melville, R. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the National Conference on Artificial Intelligence*, pages 187–192. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002.

[9] S. Paul and M. Morris. Cosense: enhancing sensemaking for collaborative web search. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 1771–1780. ACM New York, NY, USA, 2009.

[10] A. Ram and A. Francis. Multi-plan retrieval and adaptation in an experience-based agent. *Case-Based Reasoning: experiences, lessons, and future directions*, pages 167–184, 1996.

[11] B. Rogoff. *Apprenticeship in thinking: Cognitive development in social context.* Oxford University Press New York, 1990.

[12] B. Wilson and H. Meij. Constructivist learning environments: Case studies in instructional design. *IEEE Transactions on Professional Communication*, pages 0361–1434, 1997.

[13] O. Ybarra, E. Burnstein, P. Winkielman, M. C. Keller, M. Manis, E. Chan, and J. Rodriguez. Mental Exercising Through Simple Socializing: Social Interaction Promotes General Cognitive Functioning. *Pers Soc Psychol Bull*, 34(2):248–259, 2008.