Using location for personalized POI recommendations in mobile environments

Tzvetan Horozov, Nitya Narasimhan, Venu Vasudevan
{horozov, nitya, venuv}@motorola.com

Abstract — **Internet-based recommender systems have traditionally employed collaborative filtering techniques to deliver relevant "digital" results to users. In the mobile Internet however, recommendations typically involve "physical" entities (e.g., restaurants), requiring additional user effort for fulfillment. Thus, in addition to the inherent requirements of high scalability and low latency, we must also take into account a "convenience" metric in making recommendations. In this paper, we propose an enhanced collaborative filtering solution that uses location as a key criterion for generating recommendations. We frame the discussion in the context of our "restaurant recommender" system, and describe preliminary results that indicate the utility of such an approach. We conclude with a look at open issues in this space, and motivate a future discussion on the business impact and implications of mining the data in such systems.**

***Index Terms*— collaborative filtering, recommendation, location-based, location, mobile, services.**

I. Introduction

Recommender systems have been an active area of research interest since the advent of e-commerce. Companies such as Amazon [7] have shown that a retail experience can be substantially enhanced by statistically correlating macro patterns in buying and browsing behavior. Amazon's success has often been copied by a variety of retailers (e.g. CDNow, Barnes and Noble) and there have been a substantial number of experiments on different flavors of recommendation systems and their effectiveness in a variety of retail domains.

However, the usage of recommender systems in e-commerce is likely to differ substantially from the requirements of similar systems in mobile environments in three aspects:

1. **Scalability**: Unlike infrastructure systems that deal with specific items, and are used by a restricted user community (e.g., Amazon shoppers), mobile systems have to support a rapidly expanding population of mobile users and information. Recommender systems must scale to user and item populations potentially numbering in the multi-millions.
2. **Latency:** Unlike traditional users of recommender systems (who may access the results from a desktop at home or at work), the mobile user is on-the-move (i.e., has a shorter attention span) and carries a more resource-limited device (less storage and real-estate for display). Hence, recommendation results need to be chosen well *and* need to be delivered quickly.
3. **Convenience:** Desktop recommender systems typically deliver "digital" information without having to worry about the user's convenience in following up on the recommendation. For instance, a movie database will recommend movies that the user may like – but rely on the user to follow-up on theatres where the movie is currently shown. Mobile systems on the other hand inherently indicate the potential of a user to follow-up on the recommendation – thus, a mobile user requesting a restaurant recommendation typically wants to know *not* about the best match for his dining preferences, but rather about the best match that also happens to be *conveniently-situated* given his current location.

Scalability and latency concerns apply equally well to both mobile and enterprise environments, though they may face additional challenges in the former due to the inherent resource constraints and rapidly-expanding user population. However, the issue of "convenience" is more dominant in mobile environments. Consequently, we focused our attention on investigating these issues in the context of a recommender system for location-based points of interest (POI). In particular, we discuss our experiences with *GeoWhiz,* a real-world deployment of our restaurant recommender system that is currently in use by over 200 users in the Chicagolands area.

The rest of the paper is organized as follows: in section 2, we provide background information on collaborative filtering and recommender systems. In section 3, we provide details on the GeoWhiz system implementation, with section 4 dedicated to discussing preliminary results obtained from the real-world deployment of the system. Section 5 focusses on identifying some of the business utility to mining the data from such location-based recommender systems. We conclude with a discussion of related work and future directions for research work in this space.

II. Background

Recommendation systems use a well-known technique called collaborative filtering when trying to predict the rating of a product to a particular user. The general idea behind collaborative filtering is that similar users vote similarly on similar items. Therefore, if similarity is determined between users and items, a potential prediction can be made for the vote of a user for some item. There are two major flavors of collaborative filtering:

- **Memory-based collaborative filtering** – Also known as user-based collaborative filtering [10], these algorithms try to establish a correlation between users based on their voting pattern. Such correlation is computed dynamically between different pairs of users, *every time a prediction is to be made*. This puts a big computational and memory load on the system if the prediction is to be delivered in real-time. For that reason, such systems do not scale particularly well for large datasets and are not very popular in real applications.
- **Model-based collaborative filtering** – Also known as item-based collaborative filtering [9], these algorithms are widely popular today and are used primarily because of their scalability with huge datasets [7]. Instead of focusing on similarity between users, such systems compute the similarity between items on which users have voted. The idea behind the algorithm is that if all people who bought item A also bought item B, then items A and B must be "similar" in some context.

In the latter item-based approach, information about item-to-item similarity is stored in a static data structure usually in the form of a matrix. Each row of the matrix lists all items in descending order of their similarity with comparison to a given item. For example, row *i* lists all items in such a way that column 1 contains the item that is most similar to item *i*, column 2 contains the item that is second in terms of similarity to item *i* and so on until the last column contains the item that is least similar to item *i*. Whenever a user selects a given item, the system can quickly retrieve the first *X* columns from the row of the item which would be the items that are most similar to the selected item. This allows the system to scale really well even in the case of very large datasets (tens of millions of items in the case of Amazon). The data in the matrix is updated periodically (e.g. once a day) by computing the item-to-item similarity taking into account the new votes for each item. The update process can be very time consuming but since it is done offline the update does not affect the performance of the system.

Although the item-based approach performs extremely well for recommending "digital" items such as textbooks, songs or even news articles, it could be *impractical* for recommending location-based "physical" items or points of interest (POI) to the user for two reasons:

- *Relevance:* With the default item-to-item correlation, the system will select the top "*X*" POIs that correlate well to POIs previously voted on by that user. However, all the *X* items may not necessarily be in the vicinity of the user's stated location, and thus have less utility to users who are not prepared to make the extra effort to change locations. In other words, recommending a downtown restaurant to a suburban dweller fails the 'convenience' check unless there is an overriding criterion (e.g., destination restaurant for a special occasion) that warrants the recommendation.
- *Latency:* Because all default recommendations may not be relevant, such systems may need to perform additional computation (e.g., filter the results in each row by location) to generate relevant results. While such computation is possible, the time taken to do this potentially increases in a linear fashion with the size of the dataset. Given that typical POI datasets in the United States are on the order of millions of items, the latency overhead can make such approaches impractical without special partitioning of the dataset[1].

Instead, we propose a novel extension to the well-known memory-based collaborative filtering algorithm, optimizing it for its application to location-based items. We achieve the optimization by employing personalized location-based data partitioning method that allow the system to scale even for very large datasets. For testing the utility and performance of our optimizations, we developed and deployed *GeoWhiz*, a practical restaurant recommender system that uses traditional user-based collaborative filtering techniques coupled with location-based partitioning. GeoWhiz enabled us to observe real-world usage of such systems, and the collected voting data provided data-mining fodder for interesting observations on potential categorization of such points of interest (POIs).

III. The GeoWhiz Restaurant Recommender System

The biggest challenge in getting a scalable recommendation algorithm is to quickly be able to reduce the complete dataset to a more manageable subset that can produce relatively accurate results. In Geowhiz, we started with the assumption that *people who live in the same neighborhood are likely to visit the same local places* (no one likes to travel). With collaborative filtering, since people can be correlated only if they have common voting items, we can further infer that there is a higher probability of correlating people who live close to each other than correlating people who live further apart.

In order for us to validate such assumption, we needed some

[1] We note that item-item CF databases are relatively static, requiring sizable changes in user voting patterns to impact item correlations. As a result, we can envision speedup of location-based filtering of data by exploiting the inherent parallelism in searching such databases; e.g., given that rows are already sorted by relevance, the top *n* items per row can be allocated to one host, the next *n* to a second; results collected need only be sorted by priority before presentation.

actual voting data for location-based POIs. We decided to build our own collaborative filtering system for recommending restaurants and use the collected voting data to prove or disprove our theory. Hence, GeoWhiz Restaurant Recommendation engine was born. The service is still accessible at *http://geowhiz.motlabs.com/*. We used a database of about 12,000 restaurants in the vicinity of Chicago and its suburbs as POIs for our service. The service recommends restaurants to mobile users at a given location. It also allows users to rate some restaurants and uses the ratings data to compute a restaurant recommendation for the user at a given location.

In addition to exposing the functionality of the system through a web based interface, we developed a J2ME client that can be installed on a mobile (GPS enable) Motorola phone. Using the phone, users can get restaurant recommendations at a particular location, as well as cast their vote on a given selection. The J2ME client applications are available for download at the site of the service and contributed tremendously to the popularity and usage of the system.

The biggest challenge we faced in building the system was in dealing with the “cold start” problem [11]. This is a well-known phenomenon in the collaborative filtering world and has to do with the fact that if there are no votes in the system, a user can not be correlated to anyone else and no prediction can be made. This reduces the utility of the system and makes user participation low. In order for us to overcome this we introduced the following optimizations:

A. *Pseudo-users*

Pseudo-users are users who are artificially introduced in the system in order to generate votes. We started by categorizing every restaurant that we had in our database according to the type of cuisine that it provides (e.g. italian, greek, thai, etc.). For every category we created one pseudo user. For every restaurant that we managed to get the category of we created a positive vote in the database for the pseudo-user of the appropriate category. As a result, even if there are very few actual votes in the system, a user can easily be correlated with some category user without special modification to the recommendation algorithm.

B. *Serendipity*

This is a well-known technique [11] and allows some randomness to be introduced to the recommendation. Whenever a user is given a recommendation for some restaurants a random recommendation is made for a restaurant. This not only allows serendipity to be introduced to the system, but also provides the capability of generating additional recommendations, in the case when there are insufficient number votes. Note that instead of completely serendipitous random selection we can instead employ an affinity-driven random selection, which limits the selection set of recommended items based on some affinity metric (e.g. location)

C. *Average*

Recommending statistical average is another technique that is often used to benchmark the performance of recommendation systems. In case when there are no votes available for the current user, the system recommends the POIs that have the highest average rating in the vicinity of the user.

As mentioned earlier system uses traditional person-to-person correlation using the Pearson (REF) coefficient:

$$w(a,i) = \frac{\sum_j (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_i)}{\sqrt{\sum_j (v_{a,j} - \bar{v}_a)^2 \sum_j (v_{i,j} - \bar{v}_i)^2}}$$

In this formula the Pearson coefficient between user a and user i is computed as j is the total number of items and $\bar{v}_a$, $\bar{v}_i$ are the mean votes of users a and i.

Our person-to-person correlation algorithm works as follows: for a given location we determine the available POIs within a certain user-defined radius. From all users in the database, we select all people who have cast a vote on any of those POIs. On these people and their votes, we compute a person-to-person correlation with the given user and using the weighted average of their similarity, we predict a vote on each of the selected POIs. The POI with the highest predictions are recommended to the user. In addition to that we use the results from each of the optimization techniques describe above (currently 3 recommendations from each technique) and add those to the final recommendation set. Any duplicates that arise in this process are removed before final presentation to the user.

As described above, the algorithm is not likely to scale well when the size of the voting population increases. However, the primary purpose of our system was to collect as many votes as possible. Subsequently, we use the collected voting data to analyze an optimization technique that would improve the scalability and performance of location-based recommendation systems. We discuss the results of such optimization in the next section.

IV. Analysis and Observations

As of the writing this paper, there are currently 208 registered users in our database who have cast 1550 votes on restaurants in the Chicago area. The votes for each restaurant are on a scale 1 to 5 (5 being the best experience and 1 being the worst). When registering to use the service, every user is asked to provide the zip code, where he/she lives. By using a geo-map of the postal codes in US we can determine the latitude/longitude of each zip code and hence the location where a user resides. Figure 1 shows the geographical distribution of the user population and Figure 2 shows the geographical distribution of the restaurants, where vote has been cast by users.

From the attached figures it can be seen that the voting population spans over a rectangular territory with size 50x50 miles. With the collected data we decided to verify our hypothesis that users who live nearby are easier to correlate than user who live further apart.

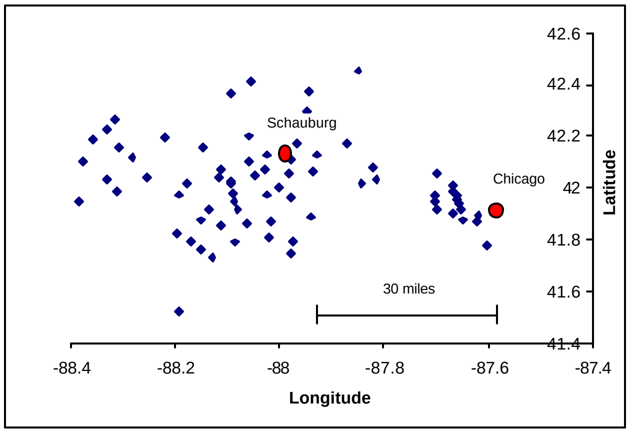

Figure 1. Geographical distribution of the user population

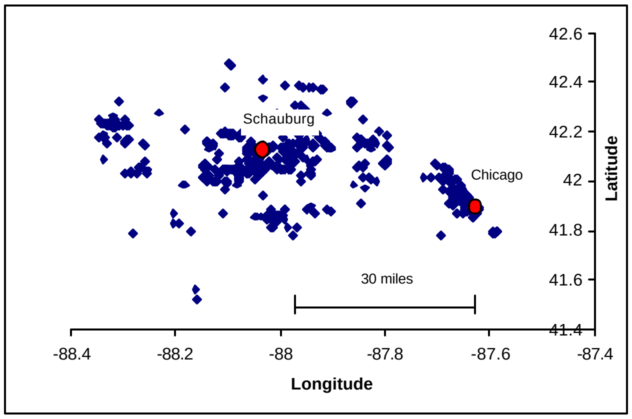

Figure 2. Geographical distribution of the restaurants with at leas

The first step that we took was to find out how much users are willing to travel on average in order to get to a restaurant. This was important information because if people tend to travel short distances from where they live, we can easily access most of someone's votes by applying location-based filtering around the person's home location. In order to test if this was the case, we computed the average distance between the home area of every user in the database and the location of each visited restaurant. Given the data at hand we computed that users on average travel about 6.79 miles to get to a restaurant. This was a very promising result as it led to the conclusion that people will tend to cast most of their votes in an area that is 14 miles in diameter.

We next proceeded to find any location-based dependency between the voting patterns of users and their home locations using the following method: first we computed the distance between the home areas of any two users in our database. Then we summed all distances and divided the result by the number of pairs, thus getting the average distance between two users in our database. With the data in hand the average distance between two users in our database is computed to be 18.22 miles. In our next step we measured the average distance only between pairs of users, who have at least one vote in common. The result was 13.68 miles on average. This was our next important discovery: users, who have at least one vote in common, are more likely to live close to each other (closer to each-other than the average pair of users in the system). Using the Modus Tollens rule of inference we can conclude that "if people do not live close to each other, then they are not *likely* to have a common vote". Even though this conclusion does not prove our hypothesis, it is well aligned with it and we continued our investigation with the collected data.

We next designed a metric that defines the capability of the system to correlate people with one-another. Given the fact

TABLE I

Distance between users (miles	Number of possible predictions	Average size of population used	Accuracy
No restriction	760	208	0.77
40	760	190	0.76
30	737	164	0.77
20	673	115	0.74
10	565	55	0.76
5	398	35	0.81

that a prediction for a user for a particular restaurant can be made only if the user can be correlated with another user (who has visited the given restaurant), the number of total predictions that can be made in the system is indicative of the number of correlations that can be made between users. The more predictions the system can make the more people can be correlated to one-another. Given all the data in the database there is a total of 760 predictions that the system can make, without imposing any restriction on the location of the users whose votes are used. We then decided to restrict the correlation between users to only pairs, who live within a predefined distance. By imposing that restriction, we monitored the number of predictions possible with the resulting subset of users. The results are shown in Table 1.

The first column indicates the distance restriction that we imposed on each pair of users when making a prediction for one of the users. For example, the last row indicates that correlating only people who live at most 5 miles away produces 398 possible predictions. The second column indicates the number of predictions that are possible given the distance limitation. The third column indicates the average number of people used, when correlating users who live closer than a predefined distance. For example the last row indicates that when correlating people who live only 5 miles apart, only 35 users of the database population is used on average. The last column indicates the accuracy of the correlation and is measured using the average error between all predictions for our person-to-person algorithm. The error is based on the 1 to 5 voting scale (e.g. 0.77 indicates that the prediction is +/- 0.77 of the actual user vote). Figure 3 plots column 1 and 2 of the table for more clear analysis of the data.

The first observation that can be made from Figure 3 is that the rate at which the number of predictions increased grows much faster than the population from the database used, when the dis tance between individual users is increased. For example if we impose a 20 miles radius when correlating pairs of users, we find out that we have reduced the number of possible predictions by just 11% (from 760 to 673), therefore substantially preserving the prediction capability of the system. In the mean time, we are using only about 55 % of the voting population database which drastically reduces the amount of data that we must deal with. This leads us to conclude that by using a simple Pearson person-to-person correlation and ignoring users who live further apart when we do the correlation, we can successfully reduce the number of voting data needed, without significantly reducing the capability of the system for a prediction.

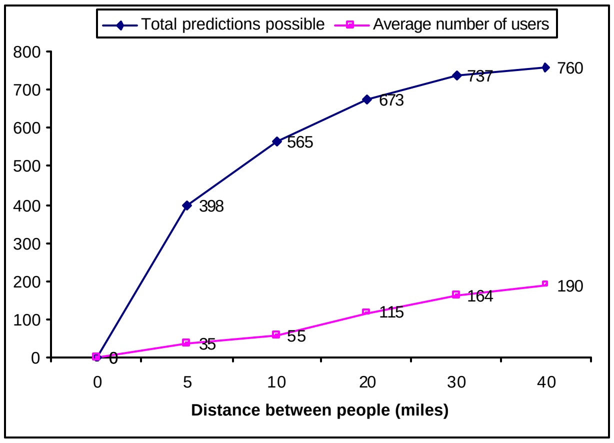

Figure 3. Correlation between total possible predictions and average number of users used for each prediction with respect to the limiting distance between people

As a summary of our findings we can claim that there is sufficient evidence supporting our primary hypothesis: people who live closer to each-other are easier to correlate than people who live further apart. Such conclusion has tremendous implications for the scalability of location-based prediction systems. For example, when we want to make a prediction for a given user on a given restaurant, from the entire voting set (consisting of millions of users and votes on different items) we can only use the subset of users, who live near the active user to make the correlation. Given appropriate voting database partitioning (e.g. according to location) a system for location-based recommendations can achieve reasonable scalability maintaining sufficient accuracy of the predictions.

Note than in the last context, we may need to differentiate between user's *current* location and user's *home* location. Default correlation can be made with the user's home since that is his center of operations from a convenience perspective. However, should the user enters a specific zip code or provide related location information, the default correlation can then be made with users resident at the specified location – implication is that users resident at a location would be the best judges on neighborhood restaurants

V. Mining Location-Based Data

While the purpose of the GeoWhiz recommender system was to validate our initial assumptions on location-driven recommendations, mining the resulting data also provided us additional insights. In particular, we see a number of potential business implications to mining user behavior patterns in such systems. For instance, data mining the restaurant recommender system highlighted the following possibilities:

- **Destination Restaurants:** Data mined from votes on the restaurants indicate that the average distance traveled by patrons is 6.8 miles. However, certain restaurants with higher-than-average commute distances managed to garner a sizable number of user votes indicative of their popularity despite the inconvenience to the user. Such locations – that we refer to as *destination* restaurants – can attract parasitic businesses that "feed" off the same user population. Thus, knowledge of the popularity of a sports bar may interest a retailer of sports memorabilia.
- **Franchising Opportunities:** The categorization of a POI as a destination restaurant can also be useful to its owner in that it indicates an unfulfilled demand. For instance, the knowledge that a sizable population from Schaumburg IL frequents a steakhouse in downtown Chicago could be an incentive to the steakhouse owner to open a franchise in the Schaumburg area.

While the previous examples focus on the value of mining static information (e.g., user votes on restaurants), the usage of recommender systems on a mobile device also has the advantage in being able to gauge the user's follow-through on the recommendation. For instance, a restaurateur who lists his establishment in the system may gain two benefits:

- **Recommendation Uptake:** He can determine the profile of users to whom the system recommended the restaurant based on the collaborative filtering algorithm. Further, he can determine if users actually followed-through on the recommendation based on explicit feedback (user queries directions to restaurant) or implicit inference (phone correlates the restaurant's location to that of the user's GPS location at some future point in time). Such models can be used to emulate the "click-through" model for payment on advertising where the recommender system operators are paid by the business owner only for users who follow-through on system recommendations.
- **Targeted Incentives:** By the same token, a business owner can also obtain the profile of users who did *not* follow-up on the recommendation. Understanding the "pain points" that resulted in this dismissal can provide the business

owner with ideas for "incentives" that could be provided to that user to encourage him/her to get over that obstacle. For instance, a restaurateur who recognizes his location as being the inconvenience can mine the user profile and send a personalized coupon for a free dessert (knowing the user has a sweet tooth) or for a 10% off (to offset the cost of travel to his location).

VI. RELATED WORK

Collaborative filtering is a well-known solution that, given the user's voting history, identifies affinities (with other users, or between items) to identify other items that may be of interest to a given user, given that user's voting history. Traditionally such approaches have been used for recommending "digital" items – ones where the user location is relatively fixed (e.g., is at home) and access to the recommended items requires little additional effort (e.g., additional keystrokes or web-browsing). For instance, the GroupLens system [12] highlights articles of interest (to users) from USENET feeds with potentially large article populations. Similarly, the TechLens has the goal of recommending research papers, while MovieLens alerts the user to movies of interest from a movie database and Ringo [10] recommends music items based on user's music interests.

Recently however, we are seeing an increased interest in tuning such techniques for use by mobile users. In particular, such research has focused on two aspects – user interface and location. The "MovieLens Unplugged" project [5] explores the usability of such recommender systems on devices with limited display area and intermittent connectivity to the backend data store. However, their focus is on improving the recommender interface – not the algorithms themselves – for the mobile user. On the other hand, the "PocketLens" project [8] redesigns the algorithms for mobile environments, focusing on portability and trust as the criteria and leveraging peer-to-peer networks to achieve acceptable accuracy and latency for mobile users.

The popularity of location-based services has also seen the birth of numerous systems that generate and consume location-based data. These range from simple location-based filtering systems such as "Shopper's Eye" [4] that simply alert users to interesting information at that location, to systems like "GeoNotes" [3] that allow users to post informative content (e.g., comments, opinions) at a server for retrieval by future visitors to that site. One of the problems that GeoNotes tries to resolve is to come up with techniques that give users the capability to both post and retrieve large quantities of relevant GeoNotes in a scalable way preventing user disturbance and information overload. To resolve the latter, the authors resort to personalized information filtering techniques such as categorization of the information in which every GeoNote is assigned some well-known category. The system then recommends GeoNotes to users at a given location if those GeoNotes are annotated with the appropriate category and that category is of interest to the user.

Of more interest are systems that combine location-based filtering of information with personalized recommendations to users. The Pilgrim system [2] leverages user history (e.g., what URLs the user clicks on at that location) to build a preferences model for that user; the history collected over many users at that location is used to build rankings that are subsequently used to deliver location-based recommendations. It was not evident that the system exploited collaborative filtering to correlate different users' rankings in order to select best-fit results for the target user. The LPR-S system [6] takes the idea one step further by proposing a generic framework for developing location-based personalized recommender systems.

REFERENCES

[1] M. Balabanovic abd Y. Shoham, "Fab: Content-based, collaborative recommendation," *Communications of the ACM*, vol 40, pp66-72, Mar 1997..

[2] M. Brunato and R. Battiti, "Pilgrim: A Location Broker and Mobility-Aware Recommendation System", Technical Report DIT-02-092, Informatica e Telecomunicazioni, University of Trento.

[3] F. Espinoza, P. Persson, A. Sandin, H. Nystrom, E. Cacciatore, M. Bylund, "GeoNotes: Social and Navigational Aspects of Location-Based Information Systems", in Proceedings of *Ubiquitous Computing International Conference* (Ubicomp 2001), Berlin, pp 2-17, Sep 2001.

[4] A.E. Fano, "Shopper's Eye: Using location-based filtering for a shopping agent in the physical world", Proceedings of the *Second International Conference on Autonomous Agents* (Agents '98), Minneapolis, MN. pp. 416-421

[5] B. N. Miller, I. Albert, S. K. Lam, J. A. Konstan, J. Riedl, "MovieLens Unplugged: Experiences with a recommender system on four mobile devices", ACM Intelligent User Interfaces (IUI 03), Jan 2003.

[6] T. Kim, J. Song and S. Yang, "L-PRS: A Location-based personalized recommender system", Proceedings of the *International Conference of Korea Intelligent Information Systems Society* 2003, pp.113-117, Nov 2003.

[7] G. Linden, B. Smith and J. York, "Amazon.com Recommendations: Item-to-item collaborative filtering", *IEEE Internet Computing*, pp 76-80, Jan-Feb 2003.

[8] B. Miller, J. Konstan, L. Terveen and J. Riedl, "PocketLens: Towards a Personal Recommender System", *ACM Transactions on Information Systems*, 22(3), July 2004, pp. 437-476

[9] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms", In Proceedings of the *10th International World Wide Web Conference (WWW10)*, Hong Kong, May 2001

[10] U. Shardanand and P. Maes, "Social Information Filtering: Algorithms for Automating Word of Mouth", Proceedings of *CHI 95*, Denver CO.

[11] J. L. Herlocker, J. A. Konstan, L. G. Terveen and J. T. Riedl, "Evaluating Collaborative Filtering Recommender Systems", in ACM Transactions on Information Systems, vol. 22, no. 1, pp5-53, Jan 2004.

[12] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon and J. L. Riedl, "GroupLens: Applying collaborative filtering to USENET news", *Communications of the ACM*, vol. 40, pp 77-87, Mar 1997.