# Social Shopping Adviser: Recommendation platform based on mobile services

Elena Burceanu, Ciprian Dobre and Valentin Cristea
Department of Computer Science
University POLITEHNICA of Bucharest
Spl. Independentei, 313, Bucharest
Romania
E-mails: elena.burceanu@cti.pub.ro, {ciprian.dobre, valentin.cristea}@cs.pub.ro

**KEYWORDS**
Recommandation, shopping assistant, mobile services, relevance score.

**ABSTRACT**
As nowadays society is a consumer orientated one, choosing which goods to buy is a very common and time consuming activity. Developing an intelligent, social based recommendation system is a good way to overcome the problem of products information overload. Social recommendations for products refers to the fact that consumers trust a product more if there is a review (of any kind) for it, confirmed by many other users (in psychology this is known as Social Proof). Moreover, if the users that confirms it shares common 'tastes' with the consumer, not only that he/she will consider the product more reliable, but the 'good' will also match his preferences with a high probability and, therefore, he/she will prefer to buy the verified product. A social shopping adviser is what the following paper proposes. We present implementation details, together with experimental results designed to evaluate the system in a first prototype implementation.

## 1. INTRODUCTION

We live in a market-oriented society, where customers are constantly looking for the products that best satisfy their needs. Customers need information about products, they want to make the most informed decisions about the place to buy from, or which product to choose from many alike. Depending on person and product, customers might base their decisions on reviews given by (possibly familiar) people, or they might be willing to search for public information. The result is, nevertheless, a large amount of information that, today more than ever, can be near-impossible to sort and filter by humans, filled with irrelevant data or conflicting opinions. Many times we face decisions regarding what a product to choose (in a supermarket, for example) from a dozen possible similar choices - and all search engine results and/or opinions obtained on social recommending networks did little to help (many times contradicting results or opinions even harden the effort). All this make it difficult to find suggestions that would suit our needs. And even if we manage to filter the information and get accurate information about products, they might be coming from persons having opinions quite different from our own. Such evidence motivates the research of novel approaches based on 'trust' and 'tags based recommendation' in a social network context (Cantador, *et al*, 2010)(Sigurbjörnsson&van Zwol, 2008). The idea is to let recommendations come only from close friends (where the term 'friend' is, even today, a disputed one, since the relation between two people can be established using different metrics, depending on the architecture of the social network and the targeted performances or objectives).

In this paper we present Social Shopping Adviser (SSA), a client-server system for recommending (in a social network context) products that have assigned a bar-code. Using modern-day technology, we aim to help the customer put in front of choosing between a large diversity of (almost) similar products or insufficiently described ones (regarding the aspects the customer wants to hear about). The client is overwhelmed with ads, and spends a lot of time choosing which product to buy, hoping to make the best decision for every decision he/she has to take. The mobile application uses barcode scanner data to find personalized recommended information relevant for each user.

We propose several novel contributions. First, we classify product information as: *relative data* (subjective data that depends on user preferences, such as 'what my friends like' or 'what tags am I am interested in') and *immutable data* (for instance 'what price has that product in a specific shop', 'what tags are associated with a product' or 'how does it look like'). Second, the system dynamically builds a *social network*, transparent to users, used only for product recommendation. This removes the privacy invasion feeling, usually given by a regular social network. Third, the system uses only a *small granularity* for each detail of a product. For example, when a user reviews a detail for some product, e.g. an incorrect photo of the product, he/she can vote only for the picture, not for all set of information, because the location or tags of the products might be correct. This way the wrong information is taken out of the system (with negative votes) without affecting good pieces of information. This leads to less effort from the clients to 'fill in' the correct details.

The rest of this paper is organized as follows. We first present several recommendation products related to the proposed system. We present the advantage of SSA, followed by a presentation of its architecture (in Section 3) and relevant implementation details (in Section 4). In Section 5 we present evaluation results for the proposed system. Section 6 gives the conclusions and presents future work.

## 2. RELATED WORK

The recommendation systems can roughly be divided in four categories (Kazienko&Kolodziejski, 2006). Demographic

recommendation systems classify users by location (based on the user profile - statically and dynamically created based on history). Products are organized in classes and the user is interested in items from a class close to his profile. In collaborative recommendation systems the user manually gives ratings to items. There are recommended items with high ratings explicitly delivered by similar users (by profiles). The content-based recommendation systems are concerned with similarities between products. The products that are recommended are related to the recently (re)viewed ones, regardless of user preferences. Finally, in case of statistical approaches, the user is presented with products having a high score given by some statistical method (the best buy, the best rated, etc).

Such algorithms can offer either too weak or too powerful personalization (single session or taking into account user history). The problem with too personalized ones (collaborative and demographic) is that they require the user to register and fill in his personal data (which requires time, and raises privacy issues on the transactions).

Collaborative and content-based methods face the sparseness issue. If the user has an identity valid only in a session, there is too little information to process, and the recommendation is poor. Statistical and collaborative methods are not able to recommend new items to the user. The remedy for these disadvantages of recommendation methods consists in the use of hybrid systems.

Folksonomies, derived from the practice and method of collaboratively creating and managing tags to annotate and categorize content (Kazienko&Kolodziejski, 2006)(Lee, *et al*, 2002), can also be used to perform collaborative tagging (Cao&Li, 2007), social classifications, social indexing, and social tagging. Recommending systems that use collaborative data can successfully be based on a folksonomy. In this case the score is created according to the vocabulary of the users, not that of the experts or of the program designers. Lee *et al* (2002) proposed a learning agent that does not recognize a consumer, but uses the information that user inputs in the system when he/she searches for suggestions. But the only problem aimed to be solved with the proposed solution is that of a customer that can identify the type of product he/she uses and describe its features. A customer profile, based on his past activity, is enough only for recommending frequently-purchased products. For others (such a laptop or a digital camera) expert advices are required (Cao&Li, 2007). The problems that occur are the synonyms, ambiguous words and phrases instead of words. Such systems have also to cope with the added noise to the search (in which case the quality or relevance of the results becomes questionable).

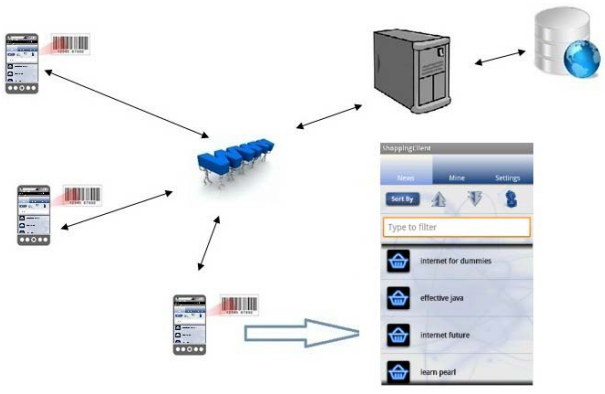
Several shopping recommendation application are also available today. TheFind (TheFind, 2012) is integrated with databases from lots of stores and offers more than an application (sign in, user profile based on static data, website support, integrated with Facebook friends and likes). But this is different from the solution proposed in this paper by the fact that TheFind is product orientated (not user orientated), and the social network is imported from Facebook (the system uses only social friends, not 'preferences' friends), not created ad-hoc, based on user preferences. In TheFind the relations are static, compared with our proposed solution, where relations are dynamically and constantly modified.

Facebook Ads (FaceBookAds, 2012) is a shopping recommendation system based on diversity and decentralization of information. Based on semantic web technologies, the system classifies ads based on user's profile, in a non-intrusive way. The problem with these ads is that the heuristics are not exact, and the advertisers constantly push their ads disregarding the user's profile. Our proposed system uses complex information. The problem of information validity is solved by letting the users vote for every suspicious part of it. This way, you can walk through details provided for a product (from different users), and vote (positive or negative) for this reviews (given tags, price, location, etc). Also one can see comments for a product, comments that can add another detail level (information that is not integrated in the current details structure).

Comparing SSA with Amazon.com gives also common different points in algorithms main heuristics. Amazon is based mainly on the facts that people who bought an specific item also bought other specific ones and that people who rated an item highly also rated other items highly. Differences in using are that in order to access information from Amazon, you need to be logged in. Also, the database needs to be statically populated. Moreover, the user identity on Amazon is public; anyone can see the reviews posted by someone. Even if both approaches uses a preferences network, the one from Amazon starts from the product: who bought/rated high this also bought/rated high this, while the solution from this paper brings in a concept oriented on the user: when someone rate high other person's input in the system (not an item), the unidirectional friendship is increased. Nevertheless, our solution doesn't use semantics yet and Amazon does.

## 3. ARCHITECTURE

Our proposed system, SSA, dynamically constructs the relations between tags and trust. The end-results is a dynamic, ad-hoc social network of users (see Figure 1). Product recognition is accomplished using barcode scanning, using the client's device camera. The central server is responsible with all information management (products and users). The data is matched against the profile of each user, such that to be relevant to each user. All recommendation and trust algorithms run on this server.




Figures 1: The system architecture.

In this architecture the server is connected to a product database. Clients exchange data with the server directly from their mobile smartphones. They can interrogate the server (over the Internet), make a change in their personal settings, or submit a new vote for a review of an item.

A first prototype implementation of the mobile client-side application was developed over the Android platform. The application is composed of three main views: list of products retrieved from the server, the details of a scanned product, and user settings. Figure 2 shows a list of products, as retrieved from the server based on the trust recommendation algorithms and the current client context in the social network. From the upper buttons, the list can be either sorted or filtered.

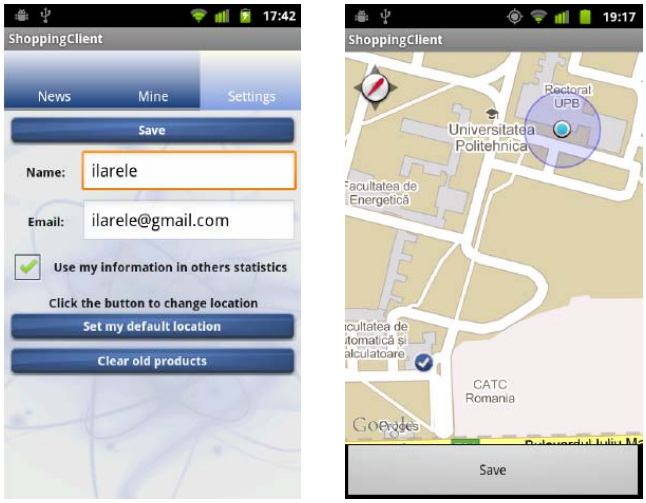

Figures 2: The list of products returned from the server.

Next, the user can scan the barcode of a certain product and communicate with the server to find relevant details about that particular item (Fig. 3a). He/she can click on any product and see details about a particular item. The details of a product are ordered according to the number of points (score) of each information, computed using the algorithm presented in the next Section. The user can train the system and insert details about products (Fig. 3b).


a. Production details. b. Initialization of product details.

Figures 3: Product dialog.

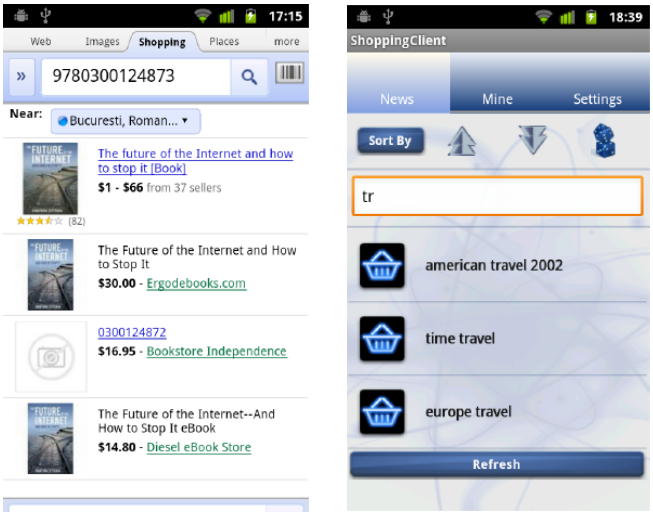
The user can specify settings such as the name and email (optional fields, used when showing comments about products), his/her current location and the privacy settings (if he/she allow for activity recordings, or the set of preferences to be used to compute trust over the social network) (Fig. 4a). A location-based dialog (Fig. 4b) can also be used to specify a search for a certain products within a designated area, or the information inserted by the user (pictures of a product) can be augmented with location information.




a. Settings dialog. b. Location dialog.

Figures 4: Personalized details.

The general data for a product refers to the information that does not depend on the user. For each product a picture (the quality of the picture may vary, but the one with the highest number of votes will represent the product) is required to better describe it. Also each product has a name, possible location data (where was spotted by users), rating, price (the price and rating are associated with different buying locations, but the price can also relate to a certain period of time – the price of a product from a specific location can vary in time), and tags.




a. Google product search. b. Filtered items.

Figures 5: Searching for products.

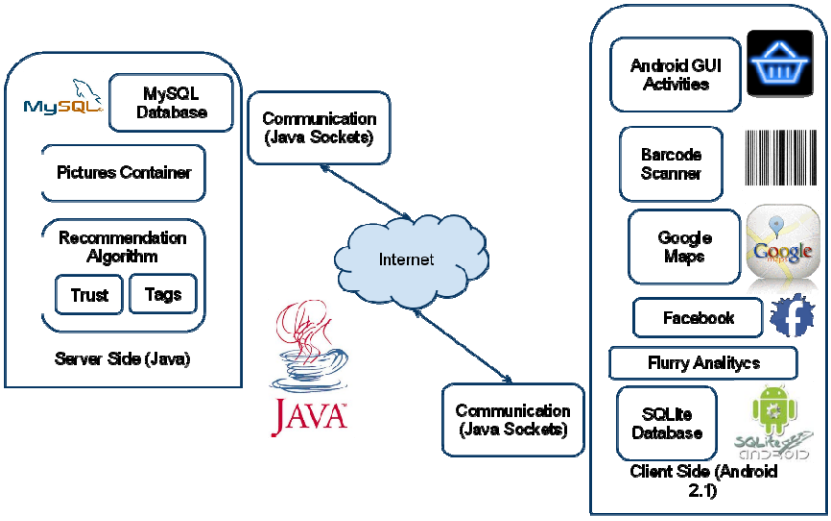
The user can insert several details. The picture of the product is recorded using the device's camera. For setting the location, a map view activity can be used, with the default location being the current user location. Other fields (such as

the rating, price, name, tags) can also be set by the user. He/she can even add comments for a particular product, or can be posted on Facebook directly from the mobile application.

The list of products retrieved from the server (Fig. 2) contains a brief description of the product description (picture and name). When touching a product the user is presented with advanced details about that product. He/she can find information about the product from an external source (via Google Product Search - Fig. 5a) or local to application (entered in the system by users of the application).

Every product detail can have multiple values, as inserted by different users. The ones showed in the dialog are those having the highest rating. The user has the possibility to iterate through those values (back and forward) to the next value. He can filter the products (by name or tags, Fig. 5b) and sort them (by date, location, price - ascending/descending or random).

On the server side there are three main modular modules: a picture container (for storing the images associated with different products), a database (the prototype uses MySQL), and the trust and recommendation module (responsible with the execution of the algorithms, and the management of the trust relations in the virtual social networks, as well as tagged preferences). Those modules can be easily replaced by similar ones for testing different implementations (Fig. 6).




Figures 6: Modular Architecture and APIs.

The communication with the server is only initiated by the client. Pushing messages from server to a mobile device is expensive for the back-end because it needs a periodic scanning from the client (through a service running in the background). This implies a persistent connection to the Internet. Both the connection and the background scanning discharge the battery with a high rate. So, to avoid this major problem for nowadays mobile devices, the server never pushes information to the client. For an update, the client sends a request and waits for answers.

## 4. IMPLEMENTATION DETAILS

The tag recommendation algorithm builds clusters of related tags. These clusters are computed based on the relation between tags (computed as the percentage of common occurrences in a description divided by the total number of occurrences for the first tag).

The virtual social network is dynamically built using the computing preferences of each user. The relations between users are used to compute the trust between any two users. The network cannot be modified manually, thus increasing the security level. The friendship between two users is computed as the 'friendship product' along the shortest path, and as an average weight for paths of the same length.

The resulted points for a product approximate the degree of interest that the user has for that product (how good it scales his/her needs for information about the product).

The server manipulates two types of data. The *relative* (subjective) one refers to which products a user might be interested in (not interesting in buying it, the user might want to know 'not' to buy a particular product). The score for this data is computed by the recommendation algorithm.

The *absolute* data refers to the details of a product for which the 'correct' value cannot vary too much between users: the picture of the product (if there are several pictures for one product, the user is presented with the most accurate one), the location and price (in a location, there should be only one single price for a specific product, in a certain time window). Also the associated tags should be clear (even if there are more sets of 'correct' tags, they are voted and the user will easily see the first entries from the list).

The recommendation algorithm is split in two parts: the first one selects the products, and the second one sorts them based on their score. The filtered products are the first *n* products to be shown to the user and are used as a base for the two main algorithms that rate products. The tag recommendation and the trust algorithms are used to decide the order of the products in the list returned to a specific client.

The static algorithm is the one that gives the details of the products, sorted by the voted results. When the user asks for a new detail of a specific kind, for a specific item, the returned result is chosen only based on number of votes for him (not in relation with the social network).

Next we present the tag recommendation and the trust algorithms, as well as the modality to combine them together.

### 4.1. The tags recommendation model

The algorithm is executed: 1) when a product is saved with more than one tag associated (to update the relation between tags); 2) when the user adds a scanned product with tags (the tags points are incremented in the user tags list); and 3) when a user votes for a picture (selecting either location, rate, price) or a name (even if the vote is positive or negative, the tag score is incremented, because it means the user is interested in that product).

The server retrieves all products in the same cluster and sorts them according to their modification dates, and the relations between tags. For each product the client votes and product id are kept synchronized. Next, the server retrieves the tags associated with the user's preferences (based on its previous votes) and the product information. Only the tags that are rated positive are considered in this step. The algorithm iterates through the product tags and finds the ones related to the user's votes. For each of them, it searches for similar tags (in the same cluster), and normalizes the score, using the value of the common references for two tags divided by the total occurrences of the first tag (intuitively, this corresponds to the probability of the two products to be of interest to the user in the same search). In the end all products are sorted by their semantic relevance to the client.

In the prototype implementation, the recommendation score is computed based on the scores from the network of friends and from the score obtained by the tag preferences. The score of the product is $score=score_{trust}+score_{tagsrecommentation}$, where $score \in [0,2]$.

The tags preferences are computed similarly. For each user, there is a list of (tag, points), updated based on votes, on the following actions: on taking picture (if an user take a picture for a product means that he/she is interested in that product, so in the product tags) and on voting for location, picture, name. The algorithm complexity is *O(maxTagsPerProduct * (maxTagsPerUser + maxPrefTags))*.

In order to reduce the noise from the results, we also eliminate some entries when computing the points for a product. This means that not only some results will weight very little, but they will not be considered at all. So, if the percentage of a tag is too small (below 1%), its contribution in the score computation is removed.

### 4.2. Trust model

Similar to the tags recommendation, the relation between two users is formed whenever a user votes for other user's review. In this case, the friendship relation between the users is updated (positive or negative) (e.g. if user *A* votes for a picture that *B* submitted, then in *A* friends list *B*'s rating is increased).

The relation between a user and a product (the interest degree) is updated whenever a user adds or comments a product (in this case he/she's interest on that product doubles), and when the user places a vote (the user's preference score is incremented by one).

The friendship network is asymmetric, so if *B* is a friend with *A*, user *A* is not necessarily a friend of *B*. If user *A* votes a detail (picture, name, tags or location group) about a product entered in the system by user *B*, then user *A* is added to the user's *B* friend list with the proper rate (the vote rate is added to the current rate that *A* has in *B*'s list).

Given a product id (*idProd)* and a user id (*toId*), the trust algorithm first finds the users interested in a certain product. It computes an average of the trust between the initial users and each of his friends. This last value is computed like:

Table 1: The algorithm for computing application permissions.

```
Set fromUserFriends ← fromUser.friends.
       entrySet();
fa (Entry friendEntry : fromUserFriends) {
   String friendId ← friendEntry.getKey();
   if (friendId == toId) {
      Float friendPoints ←friendEntry.getValue();
      Float interesRate ← getInteresRate(toId,
         idProd);
      if (interesRate > 1) interesRate = 1;
      friendPoints ← friendPoints /
         users.get(toId).numberContributions;
      if (friendPoints > 1) friendPoints = 1;
      return friendPoints * interesRate;
   }
}
nextLevelFriends.add(friendEntry);
...
```

The algorithm goes through the graph of friends in a breath first order, starting from the client id who's friends we are interested in. If any of them is the destination user, the algorithm finds the points that determines what is the relation between that user and the product (*interesRate*) and multiplies it with the 'friendship' that labels the edge in the graph (*friendPoints*/*allFriendsPoints*). Then we go on the other levels:

Table 2: The algorithm for computing application permissions (the case of subsequent levels).

```
float secondSum ← 0;
int secondNo ← 0;
fa (Entry  secondLevel : nextLevelFriends) {
   String secondId ← secondLevel.getKey();
   Float secondPoints ← secondLevel.getValue();
   Float interesRate ← getInteresRate(toId,
      idProd);
   if (interesRate > 1) interesRate = 1;
   secondPoints ← secondPoints /
      users.get(toId).numberContributions;
   if (secondPoints > 1) secondPoints = 1;
   secondSum += secondPoints *
      getTrustFromTo(secondId, toId, idProd,
      alreadyChecked) * interesRate;
```

```
    secondNo++;
}
if (secondNo == 0) return 0;
return secondSum / secondNo;
```

The friends from next levels are 'friends of the friends'. Their score is computed in a similar way with the first level friends. The difference consist in the fact that their friendship points are computed as the parent trust, multiplied with last edge value (the score along a path is the product of the edges). The score is normalized: $score_{trust} \in [0, 1]$. The algorithm only measures the trust in a friend 'taste'. The algorithm complexity is *O(maxFriendshipLevel * log maxFriendshipLevel)*.

## 5. EVALUATION AND RESULTS

To evaluate the system, we considered several book products. In the first evaluation scenario we considered the case of a user performing several operations: registering into the system, scanning, saving the list of items, taking picture, updating location, submitting reviewers, and updating data associated with certain books.

In the second evaluation scenario we cleared the database, added using the mobile application 30 books from three different domains (Computers, Cooking and Travel), created seven user profiles, used the existent books, checked the recommended products to see if they match the user's wanted profile. Finally, in the last evaluation scenario, the group of test subjects was divided in two. The first group was asked to search for the information on books using a popular search engine. The other one was asked to perform the same queries using the SSA mobile application.

### 5.1. Usability

The users selected for these experiments are students within the Computer Science Department of at University Politehnica of Bucharest. They were selected based on their expertise, such that to cover a statistically wide range of possible profiles. The users were split in two study groups, one was asked to search for information using other means, while the second group was using the proposed system. In the end, the users completed a questionnaire, with questions regarding the relevance of the obtained information, ease of use and access, the design of the application, etc. The conclusion of the usability test was that the application is able to select information that is closer to the user's own interests, and much faster than similar competing solutions. The main drawbacks, for now, were identified with the message fields (e.g., inappropriate error messages or status, or the absence of further use of contextual information).

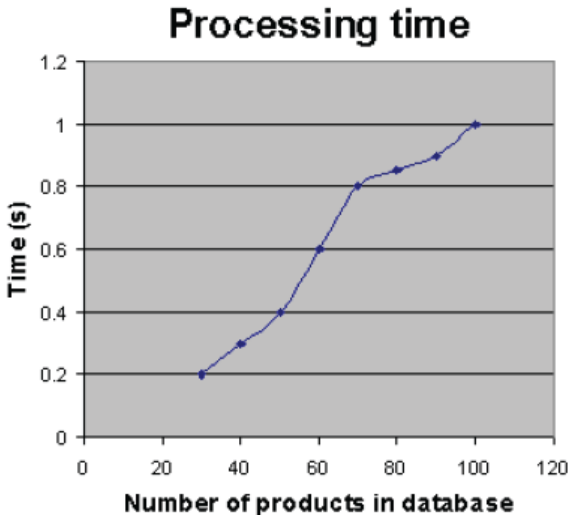
### 5.2. System's performance

Next we evaluated the performance of the system using the Flurry Analytics framework (Flurry, 2012). The monitoring framework is able to provide aggregated usage and performance data. It is a robust analysis tools because it helps exactly identifying issues (errors raised and clear details about them).

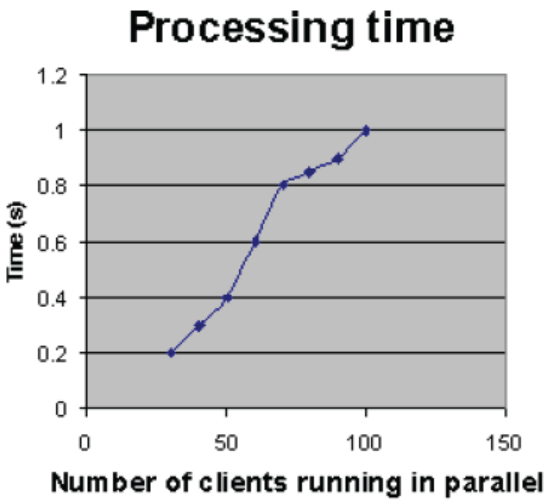
The propagation delay for a new preference depends on the synchronization with the database, and it is really small, the change is visible in a future interrogation. Once the number of users is increased, the processing time grows. This is due to the fact that in the prototype implementation the server is not distributed, and it does not scale on large amount of data (Fig. 7a). Running multiple clients in parallel also affects the response time (Fig. 7b).

For a regular utilization scenario (recommendation request, vote for 3 properties, iterate through 5 characteristics) the measured traffic was of approximately 5 KB. Using the application 20 times per day, and 3 days per week, we get 300KB. Assuming an average pricing model over a mobile operator, this leads to an operation cost of approximately 0.02 euro per week.

The delay for serving a request is composed of timerequest + timeprocessing + timeresponse. The first and the last terms




a. Multiple products




b. Multiple clients

Figures 7: Experimental results.

depend on the connection quality. The processing time depends on the load on server and on database size and indexes. For the current implementation, with 30 books in database, the complete processing was accomplished in approximately 0.2 seconds.

## 6. CONCLUSIONS AND FUTURE WORK

In this paper we proposed a system to replace the regular methods of choosing between available products on the market. The system is able to recommend products based on the real needs of the customers. It is able to aggregate data from multiple sources and form a transparent social network between users. Using barcodes, a database and a recommendation algorithm a customer can find details about a product (such as locations and prices, pictures, etc.) or he/she can obtain recommended products, based on his preferences. We proposed several algorithmic approaches for computing the recommendation scores, and for managing the list of users and products. The results show the feasibility of the system to exceed the possibilities of modern search engines, and help the user find personalized information about products.

As future work, we intend to investigate the addition of a distributed server and database to address the scalability issue. Also, for a better recommendation algorithm, we want to investigate the addition of the distance within in a social network (friends of friends) and different friend lists for each domains ('I have similar tastes in books, but not in travels'). In terms of security, the server will also be extended to include mechanisms to protect it from receiving corrupted data (e.g. a client giving more votes than allowed, a client changing his id and acting as multiple users).

## ACKNOWLEDGMENTS

The research presented in this paper is supported by national project: "TRANSYS – Models and Techniques for Traffic Optimizing in Urban Environments", Contract No. 4/28.07.2010, Project CNCSIS-PN-II-RU-PD ID: 238. The work has been co-funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/89/1.5/S/62557.

## REFERENCES

Cantador, I., A. Bellogín, and D. Vallet. 2010. 'Content-based recommendation in social tagging systems'. In *Proceedings of the fourth ACM conference on Recommender systems* (RecSys '10). ACM, New York, NY, USA, 237-240.

Sigurbjörnsson, B., and R. van Zwol. 2008. Flickr tag recommendation based on collective knowledge. In *Proceedings of the 17th international conference on World Wide Web* (WWW '08). ACM, New York, NY, USA, 327-336.

Flurry official website, last accessed February 3, 2012, from http://www.flurry.com.

Kazienko, P., and P. Kolodziejski. 2006. 'Personalized integration of recommendation methods for e-commerce'. *Journal of Computer Science and Applications*, 2006.

Lee, W.P., C.H. Liu, and C.C. Liu. 2002. 'Intelligent agent-based systems for personalized recommendations in internet commerce'. *Expert Systems with Applications*, 22(4), pp. 275-284.

Cao, Y., and Y. Li. 2007. 'An intelligent fuzzy-based recommendation system for consumer electronic products'. *Expert Syst. Appl.*, 33(1), pp. 230-240.

TheFind official website, last accessed February 2, 2012, from http://www.thefind.com/.

Facebook Ads, last accessed February 3, 2012, from http://www.facebook.com/home.php.

## BIOGRAPHY

**ELENA BURCEANU** finished her Bachelor studies within the University POLITEHNICA of Bucharest. Her research interests include distributed mobile applications and social collaborative systems. Currently she pursuits her Master studies within the Distributed Systems track, with the University POLITEHNICA of Bucharest.

**CIPRIAN DOBRE** PhD, is lecturer with the Computer Science and Engineering Department of the University POLITEHNICA of Bucharest. The main fields of expertise are Grid Computing, Monitoring and Control of Distributed Systems, Modeling and Simulation, Advanced Networking Architectures, Parallel and Distributed Algorithms. His research activities were awarded with the Innovations in Networking Award for Experimental Applications in 2008 by the Corporation for Education Network Initiatives (CENIC).

**VALENTIN CRISTEA**, PhD, is the Head of the Computer Science and Engineering Department of University POLITEHNICA of Bucharest. He teaches courses on Distributed Systems and Algorithms. As a PhD supervisor he directs thesis on Grids and Distributed Computing. Valentin Cristea is Director of the National Center for Information Technology of UPB and leads the laboratories of Collaborative High Performance Computing and eBusiness..