Peer-to-peer based Recommendations for Mobile Commerce

Amund Tveit*
Department of Computer and Information Science
Norwegian University of Science and Technology
N-7491 Trondheim, Norway
amund.tveit@idi.ntnu.no

ABSTRACT

With the increasing number of mobile commerce facilities, there are challenges in providing customers useful recommendations about interesting products and services.
In this paper a Peer-to-Peer (P2P) based collaborative filtering architecture for the support of product and service recommendations for mobile customers is considered. Mobile customers are represented by software assistant agents that act like peers in the processing of recommendations.

1. INTRODUCTION

Mobile commerce, or e-commerce in the wireless web, is providing commercial services that are accessible using mobile devices, typically a mobile phone. The main advantage of such services is their high availability, independent of physical location and time.

However, mobile commerce has its limitations, particularly regarding communication and the resources of mobile devices. Communication have less and more expensive bandwidth, higher latency, lower connection stability, less predictability, and currently less standardized protocols [10]. Mobile devices in contrast to their desktop PC alternatives, are severely constrained due to less processing and memory resources, smaller and less convenient user interfaces, in addition to a limited energy supply.

Problem

An important question related to most types of commercial activities is: *Which products or services should be recommended to a particular customer in order to make him/her satisfied?*

In e-commerce settings the automization of personalized recommendations has been sought solved using methods such as information filtering and collaborative filtering [4]. *Information filtering* recommendations is based on the analysis of a profile describing a (potential) customer's needs and preferences. *Collaborative filtering* is based on using votes or opinions about products and services from *similar* customers in order to give recommendations. Collaborative filtering has been shown as the best approach of these two methods, but it is highly dependent on a large number of customers to give good recommendations and it doesn't scale well in terms of processing.

In this paper an approach for making a scalable recommendation system for mobile commerce using a Peer-to-Peer (P2P) is considered. Peers are represented as software assistant agents interfacing a mobile customer. This approach adds the opportunity of product and service recommendations to the mobile commerce agent architecture presented in [6].

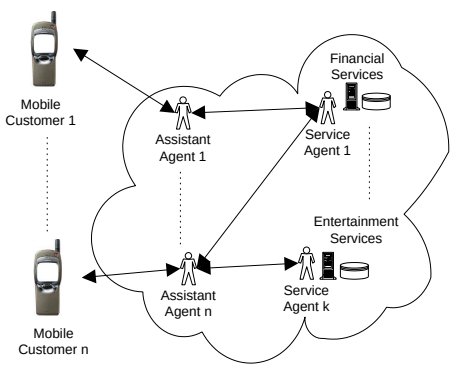

Figure 1: Mobile Commerce Agents

The rest of the paper is organized as follows. Section 2 describes collaborative filtering theory. Section 3 describes the proposed approach for P2P-based recommendations. Section 4 describes related work, and finally the conclusion with future work.

*http://www.idi.ntnu.no/~amundt/

ACM Mobile Commerce Workshop 2001, July 21, 2001, Rome, Italy

2. COLLABORATIVE FILTERING

The most important step for creating recommendations using collaborative filtering for a particular customer u is to find an ordered set of customers $\mathcal{N} = \{N_0, .., N_l\}$; $u \notin \mathcal{N}$, where the ordering is based on similarity of N_i and u, this should be performed for each customer u (i.e. $O(n^2)$ calculations of similarities with n customers in the database) [9].

Similarity between two customers u and N_i can be found by using a proximity measure. The essential data used in the calculation of the similarity is the two customers' vectors with votes on products or services. Votes are typically numerical values that are either entered by the user for products s/he dis/liked, or values added by the e-commerce provider system (e.g. if a customer has browsed or purchased a product or service, it may get a good score automatically). Two common proximity measures are the *Pearson correlation* or the *cosine measure*, shown below for customer a and b. One advantage of the *cosine measure* is that it is suitable for dimension reduction with sparse data sets (i.e. few products or services voted for).

$$corr_{ab} = \frac{\Sigma_i(r_{ai} - \bar{r_a})(r_{bi} - \bar{r_b})}{\sqrt{\Sigma_i(r_{ai} - \bar{r_a})^2 \Sigma_i(r_{bi} - \bar{r_b})^2}} \quad (1)$$

$$cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|_2 * \|\vec{b}\|_2} \quad (2)$$

When all proximity measures between all customers are calculated, a neighborhood is formed (a reasonable size subset of customers in $\mathcal{N}$ for a particular customer u). If the the vectors in the data set are sparse, a aggregate neighborhood approach can be used (calculates the centroid of the neighborhood before adding new customers). If the vectors in the data set are dense, a center-based approach can be used (picking the l most similar customers from $\mathcal{N}$).

When a set of neighborhood vectors is found, recommendations can be calculated by selecting items in the vectors from the neighborhood that the customer u hasn't voted for, e.g. if a customer u hasn't voted for particular book b, and another (i.e. the most similar) customer u' in the neighborhood has given the book a high rating, the book will be recommended for customer u.

3. PROPOSED APPROACH

The main idea of this approach is to transform the problem of finding good product and service recommendations using collaborative filtering, into a search problem that can take advantage of the scalability and privacy possible in a P2P-based architecture [8] similar to Gnutella or Freenet.

3.1 Query Flow

3.1.1 Gnutella

In Gnutella, a query (i.e. for a MP3 file) from a Peer node in are broadcasted to all its neighbours. The neighbour Peers (independently) check if the query matches one of their hosted files, if so, they return a found message back to the sending node, otherwise they decrease the TTL field (TTL = Time To Live) and then pass on the query to their neighbour peers. If the TTL count reaches 0, the message is not sent any further. In other words, the query is broadcasted in "all directions" from the quering node, but only the query matches are sent back the network path they arrived from [8]. This flow of queries ensures that a peer doesn't know if a received query was from a neighborhood peer, or another peer that broadcasted the message, this gives a limited degree of privacy for queries.

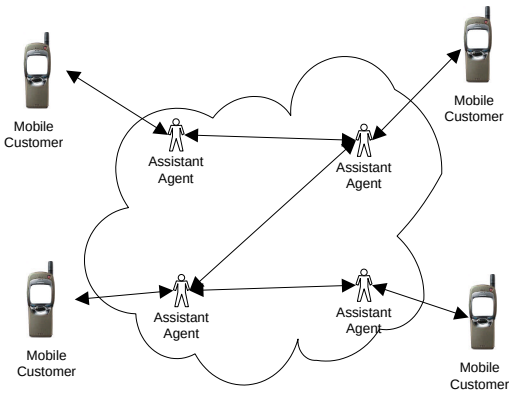

Figure 2: P2P Network of Agents

3.1.2 Proposed Approach

Queries are broadcasted to neighbour Peers as for Gnutella, but a query is not a text string (as the case for Gnutella's file search), but rather a vector with votes on products and recommendations. This voting vector is the same as used in collaborative filtering.

3.2 Routing Algorithms

3.2.1 Gnutella

In Gnutella, there is a very simple routing algorithm that either stops broadcasting from a peer if a match is found, or continues broadcasting messages to other peers if no match was found.

3.2.2 Proposed Approach

When a peer receives a query voting vector: It calculates the proximity (e.g. with Pearson correlation from collaborative filtering) between the voting vector in the message and cached previous messages at the node. If the proximity measure is higher than a threshold τ, the cached voting vector is sent back to the neighbour that sent the query voting vector (and possibly further back to the origin from there). If the proximity measure is lower than τ, the query voting vector is broadcasted further to the receiving peer's neighbours. If there are too many messages received per time unit, they are simply passed on to neighbour peers without calculating the proximity measure against cached items. The size of the cache tries to balance against the traffic, if the traffic is high the cache is decreased in order to be able to calculate proximity measures.

3.3 Dynamic Network Topology

3.3.1 Gnutella

Gnutella have a mainly static peer network topology, i.e. peers can disconnect or connect to the network, but they rarely switch places or routing tables after the initial connection the network.

3.3.2 Proposed Approach

Since each peer is calculating the proximity on incoming messages versus cached messages, the peer is able to monitor the resemblance of traffic from different neighbour peers. So if two neighbour peers send similar types of queries to a peer, it can ask the peers if they wan't each others addresses, and if both reply yes the peers establish a new connection. The reason for doing this is to try to cluster similar peers in terms of number of hops to improve performance and quality of recommendations.

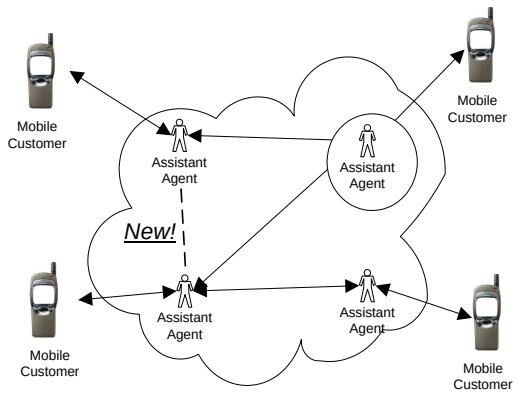

Figure 3: Dynamic Network Topology

3.4 Ranking methods

When a potentially large set of voting vectors has been returned to querying node they have to be ranked in order to get the best recommendations on top of the list before communicating to the mobile user (similar to Internet search engines). The straightforward solution would be to use collaborative filtering (rank based on the calculation proximity measures and then the neighborhood), this may sound like the same as performing traditional collaborative filtering, but the difference is that the voting vectors to be processed is only a relatively small subset of *all* voting vectors for all users of the system, this is because of the filtering done by peers that processed the query.

An alternative approach could be to return only the recommendation and the proximity measure, e.g. if the querying voting vector, represented as with $item : vote$ elements, is $[10 : 0, 22 : 2, 37 : 8]$ and the matching vector is $[10 : 0, 22 : 2, 24 : 9, 37 : 8]$ then only the recommendation $24 : 9$ together with the proximity measure will be returned, this is more efficient both regarding bandwidth usage and a useful preprocessing that makes ranking into a simple sorting job (of proximity measures).

3.5 Query Compression scheme

If voting vectors are sparse, it is more efficient to represent them as hash tables than vectors for internal processing, but to get more efficient compression when sending the vector between peers one can take advantage of the *binary interpolative compression algorithm* [7].

The binary interpolative compression algorithm was developed to efficiently compress inverse files used in document indeces in information retrieval, it utilizes the difference and context information between sorted numeric values for compression. By requiring the data to have certain properties, it can at best compress some items to 0 bytes (depending on position, value and neighbour values in the array). In order to work, the algorithm requires the data to be: *numeric*, *sorted* and *in a known numeric range*. The index positions representing items in the vector (and hash table keys) fulfill these requirements (e.g. ISBN numbers representing books), and can therefore gain from using binary interpolative compression.

3.5.1 Compression example:

Original query vector represented using a hash table (item:vote pairs)

$$\vec{q} = \{24 : 5, 37 : 8, 45 : 0\} \tag{3}$$

for item 24 with rating 5, item 37 with rating 8, and item 45 with rating 0 respectively. The compressed representation is created with

$$IPolComp([24, 37, 45], 3 - 1, 24, 45) : 580 \tag{4}$$

(i.e. compressing the list of items, and appending the votes for items after the separating colon)

3.6 Implementation

The system is planned to be implemented using the java programming language.

3.6.1 Feasible Performance?

Critical issues for high performance in peer-to-peer networks are efficient handling of requests (i.e. processing and routing time), and network bandwidth and topology [8].

Calculating similarities using a proximity measure such as the cosine (2) has linear algorithmic running time with respect to the vector length $|\vec{q}|$. Routing based on comparing an incoming query vector $\vec{q}$ with the set of cached vectors C has a run time of $O(|C||\vec{q}|)$, this can potentially be reduced by ordering the vectors in C based on proximities (whenever there are few messages to handle for the peer) and use binary search to find the closest match(es) for $\vec{q}$ in C. For each similar vector $\vec{v}$ in C $\vec{q}$ is forwarded in $\vec{v}$'s direction (i.e. the neighbour peer which $\vec{v}$ was received from).

Network bandwidth for each peer is not easily changed, but the topology can be changed with a relatively low cost since it involves a simple interaction with neighbour peers about allowing them to be introduced to each other. However, in order to avoid the peer-to-peer topology to become a complete graph, the dynamic topology heuristic should be able

to forget, i.e. remove peers from the routing table if they are of little relevance (get low proximity measures) to the majority of requests to the peer.

These approaches are similar to existing and scalable peer-to-peer networks such as gnutella and freenet, and supports the feasibility of the proposed approach.

3.6.2 Trust

Handling trust in the system could possibly be done by introducing trusted third-parties or authority peers/hubs which keeps public keys for the peers in the network. Each query should then be signed by the peer's private key so that it is possible to check that a query came from a valid peer. In order to decrease the opportunity for malicious behavior, e.g. hidden and fraudiant advertising in recommendations, one can add the opportunity for mobile users to add simple feedback on recommendations. If a vector $\vec{v}$ gave a good recommendation to a user, the origin peer p of $\vec{v}$ could gain trust points which could for instance increase the rating of the user in terms of recommendation ordering and degree of replication of p's queries throughout the peer-to-peer network.

4. RELATED WORK

Varshney et al. propose a taxonomy and application framework of mobile commerce [11]. The W^3IQ proxy based recommender system presented by Joshi uses peer-to-peer (proxy-to-proxy) based recommendations of URLs satisfying information requests [5]. Related work on personalization for mobile users include work by Cotter and Smyth [3]. They present PTV, a recommender system for TV-programmes based on both collaborative filtering and content-based recommendation strategies. Personalization for mobile users is supported by the Gulliver system by Austaller et al. [2]. Sarwar et al. gives an overview of various recommendation algorithm approaches for e-commerce [9].

5. CONCLUSIONS

In this paper a P2P-based approach for scalable recommendations for mobile commerce has been presented. The main contribution is the P2P-based recommendation approach described and the efficient compression algorithm of the communication of voting vectors.

This P2P-based approach differs from others, in the sense that there is no free-riding [1]. Free-riding is when peers do not produce and consume equal balanced of data, production is equal to hosting files, and consumption is equal to downloading files (for Napster and Gnutella). In the presented approach consumption equals production, since a query is a voting vector which is also the data object of interest in the P2P service.

Future work include implementing the system for the recommendation of mobile phone ringing tones and logos, and try to meet challenges such as: How to deal with fraudiant behavior? How to maintain consistent and unique identification of products and services in the voting vectors used in queries? How to keep cache consistency? How to deal with privacy and encryption of queries and results? Where are the bottlenecks for true scalability (millions of mobile commerce users and profiles). Is a true P2P-based systems for recommendations or is hybrid approach like Napster or Gnutella with Gnutellahosts needed?

6. ACKNOWLEDGEMENTS

I would like to thank my supervisor Mihhail Matskin, and my Agentus colleague Thomas Brox Røst for motivating discussions. This work is partially supported by the Norwegian Research Council in the framework of the Distributed Information Technology Systems (DITS) program and the ElComAg project.

7. REFERENCES

[1] Adar E., and Huberman B. A. Free Riding on Gnutella. *First Monday*, 5(10), October 2000.

[2] Austaller G., Hartl A., Kappel G., Lechleitner C., Muhlhauser M., Reich S., and Rudisch R. Gulliver Beans: Generating Device Optimized and Individualized Content for WAP Applications. In *Proceedings of the 9th International World Wide Web Conference*, 2000.

[3] Cotter P., and Smyth B. WAPing the Web: Content Personalization for WAP-Enabled Devices. In *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, pages 99–108, 2000.

[4] Good N., Schafer J. B., Konstan J., Borchers A., Herlocker B., and Riedl J. Combining Collaborative Filtering with Personal Agents for Better Recommandations. In *Proceedings of the 1999 Conference of the Americian Association of Artificial Intelligence (AAAI-1999)*, pages 439–446, 1999.

[5] Joshi A. On Proxy Agents, Mobility and Web Access. *Mobile Networks and Applications, Special Issue on Software Architectures for Mobile Applications*, pages 233–241, 2000.

[6] Matskin M., and Tveit A. Mobile Commerce Agents in WAP-based Services. *Journal of Database Management - Special Issue on Mobile Commerce*, pages 27–35, July-September 2001.

[7] Moffat A., and Stuiver L. Exploiting clustering in inverted file compression. In *Proceedings of the 1996 IEEE Data Compression Conference*, pages 82–91, 1996.

[8] Oram A., editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly and Associates, 2001.

[9] Sarwar B., Karypis G., Konstan J., and Riedl J. Analysis of Recommendation Algorithms for E-Commerce. In *Proceedings of the 1st ACM SIGecom Conference on Electronic Commerce*, 2000.

[10] Tsalgatidou A., and Veijalainen J. Mobile Electronic Commerce: Emerging Issues. In *Proceedings of EC-WEB*, pages 477–486, September 2000.

[11] Varshney U., Vetter R. J., and Kalakota R. Mobile Commerce: A New Frontier. *IEEE Computer*, 33(10):32–38, 2000.