MUADDIB: A Distributed Recommender System Supporting Device Adaptivity

DOMENICO ROSACI and GIUSEPPE M.L. SARNÉ and SALVATORE GARRUZZO
DIMET Department, University Mediterranea of Reggio Calabria

Web Recommender systems are Web applications capable of generating useful suggestions for visitors of Internet sites. However, in the case of large user communities and in presence of a high number of Web sites, these tasks are computationally onerous, even more if the client software runs on devices with limited resources. Moreover, the quality of the recommendations strictly depends on how the recommendation algorithm takes into account the currently used device. Some approaches proposed in the literature provide multi-dimensional recommendations considering, besides items and users, also the exploited device. However, these systems do not efficiently perform, since they assign to either the client or the server the onerous cost of computing recommendations. In this paper, we argue that a fully distributed organization is a suitable solution to improve the efficiency of multi-dimensional recommender systems. In order to address these issues, we propose a novel distributed architecture, called MUADDIB, where each user's device is provided with a *device assistant*, that autonomously retrieves information about the user's behaviour. Moreover, a single *profiler*, associated with the user, periodically collects information coming from the different user's device assistants to construct a global user's profile. In order to generate recommendations, a *recommender* pre-computes data provided by the profilers. This way, the site manager has the only task of suitably presenting the content of the site, while the computation of the recommendations is assigned to the other distributed components. Some experiments conducted on real data and using some well-known metrics show that the system works more effectively and efficiently than other device-based distributed recommenders.

Categories and Subject Descriptors: I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms: Recommender Systems

Additional Key Words and Phrases: Recommender Systems, Adaptivity, Personalization

1. INTRODUCTION

A large number of different recommender systems [Sarwar B., Karypis G., Konstan J. and Riedl J. 2000; Schafer J.B., Konstan J.A. and Riedl J. 2001] have been proposed in the last years to support users' Web navigation. Generally, they are partitioned in three main categories [Burke, R. 2002], namely: (*i*) *Content-based*

recommender systems, that suggest to a user those items which appear the most similar to those he has already accessed in the past; (*ii*) *Collaborative Filtering* recommender systems, that suggest to a user those items which have been also considered by similar users; (*iii*) *Hybrid* recommender systems, that exploit both content-based and collaborative filtering techniques to generate recommendations (e.g., a Web site can generate suggestions considering user's personal interests and user's commonalities among other known users). In these situations, hybrid recommender systems have been usually recognized as the most promising solution [Burke, R. 2002].

Some of these systems are capable of providing very effective recommendations for a user that visits a given Web site. However, it has been widely remarked that their performances significantly decrease when the number of the users present in the system or the dimension of the site catalogue increases. The influence of the dimension of the site catalogue is immediate to understand for content-based recommendations, that imply to compare the user's profile with the site catalogue. Another important parameter is the number of features considered in the user's profile to characterize each item, since recommendation algorithms compare the user's profile with the site catalogue based on these features. If we denote by m the dimension of the site catalogue (i.e. the number of the items contained in the catalogue), and by f the number of features considered in the personal user's profile, then the computational complexity of content-based approaches is generally $\mathcal{O}(f \cdot m^2)$. Instead, the cost of the collaborative filtering techniques generally depends, besides on the dimension of the site catalogue and the number of features, also on the number n of users present in the system. This happens because the algorithms for generating recommendations are commonly a variant of the traditional collaborative filtering algorithm [Breese J., Heckerman D. and Kadie C. 1998], and similarly to the original technique they are computationally expensive. In particular, the cost necessary to generate collaborative filtering recommendations is $\mathcal{O}(n \cdot m \cdot f)$ in the worst case, since it is necessary to examine n visitors and up to m items for each visitor, considering m features for each item. However, because the average visitor's profile is extremely sparse, the cost tends to be closer to $\mathcal{O}(f \cdot (n + m))$. If we have a huge number of users and a huge size of site catalogue, the scalability of the algorithm becomes an issue to face. To address such an issue, a variant to the traditional collaborative filtering, called *item-to-item* collaborative filtering [Linden G., Smith B. and York J. 2003], has been proposed. This technique, rather than matching the current user to similar visitors, compares each of the accessed items to similar items, then combines those similar items into a recommendation list. In order to determine the most similar match for a given item, a similar-items table is exploited by finding items that visitors tend to access together. Given a similar-items table, the item-to-item technique finds items similar to each of the current visitor's profile, aggregates those items, and then recommends the most interesting among those items. This computation is very efficient, because it only depends on the number of items the current visitor accessed. Other algorithms, as for instance Eigentaste [Goldberg K., Roeder T., Gupta G. and Perkins C. 2001], introduce further improvements, even computing recommendations in constant time. It is important to point out that all the aforementioned collaborative filtering methods

operate in two phases, each of which differently contributing to the overall computational complexity of the algorithm. In the first phase, that is performed off-line, each algorithm computes the similar-items table with a cost that is $\mathcal{O}(n \cdot f \cdot m^2)$, as it is necessary to compute $m \cdot (m-1)$ similarities, each potentially requiring n operations. In the second phase, the recommendations are generated on-line with the cost analyzed above for each algorithm.

1.1 Distributed Recommender Systems

The recommender techniques described in the previous subsection are basically conceived for generating recommendations for a user that visits a Web site having a number of different items. Therefore, the recommendations are generated based on the items that the users have accessed on the given site. In the middle of 1990s, a different type of recommender system has been proposed which considers the interests of the users with respect to all the sites they have visited in the past. Therefore, in this case, it should be necessary to compute a similarity matrix between the users (or between the items, if we use an item-to-item technique) related to all the visited sites and not only for just a given site. But in this case the task of computing the similarity between users or items requires that each site must know how the users behave in the other sites, and this is a task that heavy influences the recommendation activities.

A new promising solution to implement recommender systems that deal with different Web sites is represented by distributed recommender systems. The main component of such a system is a *software client*, which is an application capable of assisting the user in his activities on the Web. The client is able to observe the user's behaviour during his navigation and in this way it can builds a profile which represents the user's interests with respect to all the visited Web sites. When the user accesses a Web site, his client exploits the profile in the interaction with the site. Finally, the site can use both content-based and collaborative filtering techniques to provide recommendations to the user's client by adapting the site presentation. It is clear that the task of generating recommendation is, in this case, strictly related to the interaction between the site and the user's client. The site should be able to compare the profile of the user with the analogous profile of all the past users. A trivial approach to this problem could be to store all the user's profiles in the Web site, and to compute off-line the similarity table between users or items. However, the difficulty arises of updating the users' profiles while the users continue their navigation on the external sites. For these reasons, this trivial approach does not seem feasible.

1.2 Device-Adaptive Recommender Systems

To make this scenario more complicated, an emerging issue is that nowadays users navigate on the Web using different devices as desktop PCs, cellular phones, palmtops, etc. Each of these devices presents: (*i*) its own interface characteristics (e.g., display capability), (*ii*) a different cost of Internet connection, (*iii*) different storage space and computational capability. These differences can influence the user's preferences; for example, when he accesses a site with a cellular phone, he could desire a light site presentation, i.e. a presentation without high-size graphical objects, multimedia, etc. Consequently, a different profile should be built for each device

exploited by the same user. This leads us to argue that a user should be provided with a different client for each device he exploits.This problem can be viewed as a particularization of that of providing multidimensional recommendations, studied in [Adomavicius G. and Tuzhilin A. 2001; Adomavicius G., Sankaranarayanan R., Sen S. and Tuzhilin A. 2005]. In this works, authors observe that in many applications it may not be sufficient to consider only users and items for generating recommendations, but it is also important to incorporate the contextual information about the users decision scenario into the recommendation process. Such an issue, preliminarily introduced in [Adomavicius G. and Tuzhilin A. 2001], has been deeply investigated in [Adomavicius G., Sankaranarayanan R., Sen S. and Tuzhilin A. 2005], where authors present a multi-dimensional model for supporting recommendations and formulate the multi-dimensional recommendation problem.

In this paper, the additional dimension that we deal with, besides item and user, is represented by the exploited device, that should be strongly considered in the generation of recommendation. Moreover, such a particular dimension involves the possibility that also devices with limited resources could be exploited by the users, and in this perspective the possibility to distribute the computation on a large system of collaborating resources becomes crucial.

Some recent approaches tried to consider the use of the exploited device for generating recommendations as, for instance, EC-XAMAS [De Meo P., Rosaci D., Sarné G.M.L., Ursino D. and Terracina G. 2007] and X-COMPASS [Garruzzo S., Modafferi S., Rosaci D. and Ursino D. 2002]. However these approaches, exploiting a client-server architecture, assigns the task of performing the recommendations to the client, making difficult the applicability of the system to devices having limited resources. A more recent distributed recommender system, called MASHA [Rosaci D. and Sarné G.M.L. 2006], overcomes such a difficulty. The new contribution given by MASHA is represented by the use of a distributed architecture to make the task of building a user's profile independent of the task of generating recommendations. This way, differently from the other device-based recommender systems as EC-XAMAS and X-COMPASS, MASHA avoids the client, running on a device that could have limited resources, to perform the onerous task of generating recommendations. However, MASHA presents two main limitations. The first is the significant computational cost of the adapter activities, due to the execution of the recommendation algorithm. More in detail, if n is the number of site visitors and m is the maximum number of items in a site, and considering that the number of features considered in the user's profile is equal to 1 (being limited to a synthetic interest coefficient), the computational complexity of the recommendation algorithm is $\mathcal{O}(m \cdot n^2)$ in the worst case, since it compares the profile of each visitor with those of the other visitors and considers up to m items for each visitor.

To the best of our knowledge, the limitation of MASHA is common to all the others distributed recommender systems, as discussed in [Rosaci D. and Sarné G.M.L. 2006; De Meo P., Rosaci D., Sarné G.M.L., Ursino D. and Terracina G. 2007; Garruzzo S., Modafferi S., Rosaci D. and Ursino D. 2002]. None of these systems perform better than $\mathcal{O}(m \cdot f \cdot n^2)$.

The second limitation involves the quality of the generated recommendations. MASHA exploits information related to the device in order to adapt the site pre-

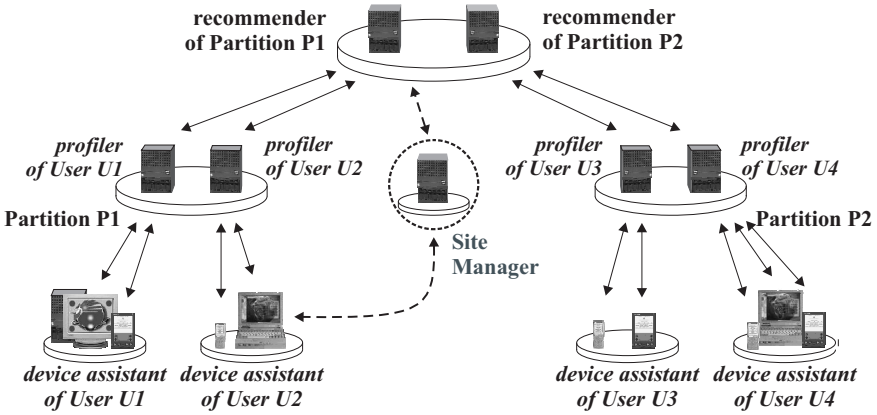

Fig. 1. The MUADDIB Architecture

sentation, but the content of the recommendations does not significantly depend on the exploited device.

1.3 MUADDIB: a solution to improve both effectiveness and efficiency of multi-dimensional recommenders

To overcome the two limitations discussed above, in this paper we present the *MUlti ADaptive and DIstributed Behaviour* (MUADDIB), that is an extension of the MASHA architecture. Like MASHA, also MUADDIB allows to support the adaptivity of the recommendations with respect to the exploited device. However, the architecture of MUADDIB represents a deep evolution of MASHA, since it has been conceived to implement a new recommendation algorithm which is more effective and presents computational costs significantly reduced with respect to MASHA. The main feature of this recommendation algorithm is that the computation of recommendations is fully distributed on the whole system, while in MASHA is only partially distributed.

The MUADDIB architecture (see Figure 1) maintains the three MASHA components: (*i*) *device assistant*, associated with each device, (*ii*) *profiler*, associated with each user, and (*iii*) *site manager*, associated with each Web site. Differently from MASHA, the recommendations are not autonomously generated by the site manager, but they are the result of a collaboration between the site manager and a new software component, called *recommender*. Indeed, the basic idea underlying MUADDIB is that of clustering the community in groups of users having similar domains of interest and similar global profiles, where each group is managed by a recommender. The clustering is periodically performed in two phases. In the first phase the users are partitioned simply based on their declared domains of interest. In a second phase, a clustering manager associated to each group produced in the first phase refines the clustering, dividing the group in sub-groups of users having similar profiles. Figure 2 graphically describes how the recommender system works. In particular, we observe that the four components (device assistant, profiler, site manager and recommender) collaborate in two different and parallel path. The first path, depicted on the top of Figure 2, is related to the construction of the user pro-

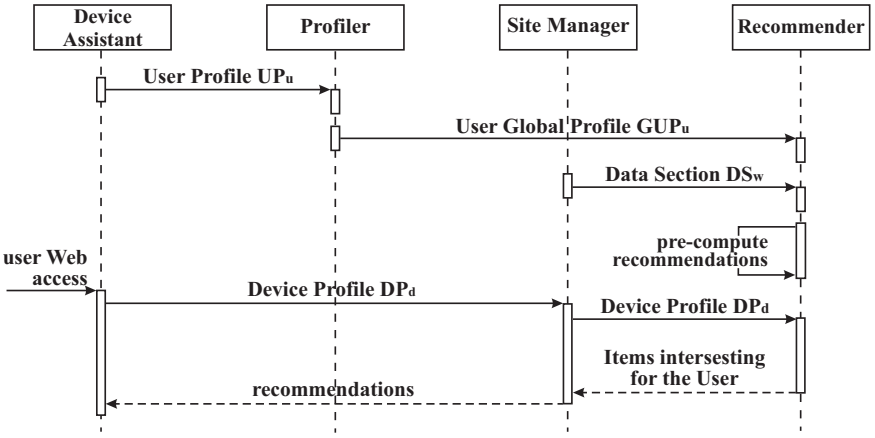

Fig. 2. The UML sequence diagram of MUADDIB

files and their exploitation in computing similarities between users. In particular, the device assistant i sends its user's profile UP to the profiler, that collects all the user's profiles coming from the device assistants in order to construct the global user's profile GUP. Then, the profiler sends the global user's profile GUP to the recommenders of all the partitions which the user u belongs to. This way, the recommender collects all the global user's profiles related to the users of its partitions. Moreover, the site manager of the site W sends to the recommender some information that describes how the users that belong to its partition accessed in the past to the categories contained in the site. This information is stored in a data structure DS_W that will be described in detail in Section 4.4. The second path, represented on the bottom of Figure 2, is related to the recommendation activities. When the user u accesses a Web site W, the device assistant of u interacts with the site manager of W and sends to it some information about the u's preferences. These preferences are related to the presentation format desired by u when exploiting that device, and are stored in the device profile DP_i whose structure will be described in Section 4.1. Then, the site manager contacts the recommenders of the partitions which u belongs to (recall that a user can belong to one or more partitions based on his domains of interest). These recommenders pre-computed the items of the site that best match with the device profile of u, to support content-based recommendations. Moreover, recommenders also pre-computed the items accessed by other users similar to u and that exploit the same device of u, to support collaborative filtering recommendations. Then, these items are transmitted to the site manager of W that generates recommendations for u with a suitable site presentation.

Indeed, the computational cost on the site side is $\mathcal{O}(m+p)$ (where p is the number of different partitions) that is significantly lower than the cost required by MASHA.

It is important to point out in this analysis of computational cost, that the advantages in terms of efficiency introduced by MUADDIB have as a counterpart the necessity of performing a clustering activity using the clustering managers. This necessity does not affect the efficiency of the recommendation performances, since in MUADDIB the clustering of the users is realized off-line, at pre-fixed instants of

time. Recommendations are computed on the last configuration of the clustering, therefore their computation is independent of the clustering computation performed by the clustering managers. However, since the recommendations are computed on the last available configuration of clustering, an approximation is introduced in the recommendation quality. This approximation is more significant as more is the period tc between two consecutive clustering computations. If we fix a small value for tc we obtain a high frequency for clustering computation and consequently a good quality of the recommendations. Vice-versa, high values for tc lead to low-frequency re-clustering and therefore worst results in terms of recommendation quality. As we will deeply analyze in Section 4.3, the possibility to fix a small value for tc strictly depends on the number of available clustering managers. Indeed, if we have a little number of clustering managers, each clustering manager has to handle a great number n of users, and the clustering activity has a computational cost $\mathcal{O}(n^2 \cdot m^2)$, very expensive with respect to n. In the number of clustering managers is large, the value n for each group of users becomes small enough to allow a high frequency of the re-clustering. Due to the presence of the recommenders, that are able to perform off-line pre-computed recommendations, MUADDIB can also consider in its recommendation algorithm more detailed information about the exploited device than MASHA. This leads to overcome also the second limitation of MASHA discussed above. The above considerations highlight that MUADDIB introduces a new methodology of computing recommendations in presence of a multi-device environment, based on (i) the idea of pre-computing recommendations on a set of clustered users in order to improve the efficiency and (ii) the idea of exploiting a large number of distributed resources to cluster users in partitions homogeneous from the viewpoint of the users' interests, so improving also the recommendation quality. We point out that, with respect to our past proposal in this setting (i.e. MASHA), we introduce a completely novel recommendation algorithm, based on the pre-computation of recommendations exploiting the distributed architecture, while MASHA does not use any pre-computation mechanism. Furthermore, as another important difference, MUADDIB introduces the idea of clustering the users having similar profiles to efficiently support the pre-computation of the recommendations, while MASHA does not support clustering. Finally, the new recommendation algorithm used in MUADDIB has implied a substantial change of the distributed architecture with respect to MASHA, introducing two new fundamental components, namely the recommender and the clustering manager. In words, MUADDIB can be seen as a way to maintain multi-dimensional information on a distributed environment, in order to achieve high efficiency and a good effectiveness also in presence of devices with limited resources.

We have experimentally evaluated MUADDIB by comparing it with both MASHA and other recent distributed recommender systems that take into account the particular used device. In this evaluation we have used some well-known accuracy metrics, as MAE, ROC, CROC and NPDM, that measures the effectiveness under different perspectives, and we have observed that MUADDIB presents a better quality of the recommendations than the other systems. Moreover, an analysis of the time cost with respect to the number of users has shown that MUADDIB introduces a significant improvements of the performances.

1.4 Scenarios of applicability

The above experiments lead us to conclude that our approach is very suitable to be applied in presence of a high number of users, since the efficiency of MUADDIB in this situation is significantly better than other recommender systems. Another scenario of applicability is a community of users that mostly exploit devices with limited resources, as palmtops and cellular phones. In this case, the advantage of using MUADDIB with respect to other systems becomes significant also in presence of a medium-low number of users, due to the generally high cost of the network connection. Finally, the analysis of accuracy shows that MUADDIB is particularly suitable for those applications where it is necessary to predict with high accuracy the rates the users give to the items. The analysis based on the above measures indicate that, for this applications, MUADDIB largely outperforms the traditional recommenders, especially if the number of users in the system is large. This is the case, for example, of e-learning recommender systems, where it is necessary to predict the rates the students give to educational resources.

The plan of the paper is as follows. Related work is examined in Section 2. Section 3 introduces the knowledge representation model used in MUADDIB. In Section 4 we describe in detail both the MUADDIB architecture and the proposed recommendation algorithm. In Section 5 we provide a theoretical evaluation of the efficiency of the approach in comparison with other recommender systems proposed in the literature. An experimental comparison between MUADDIB and the other considered systems is described in Section 6. Finally, in Section 7 some final conclusions are drawn.

2. RELATED WORK

A relevant number of Web recommender systems have been proposed in the last ten years. All these systems aim at proposing next pages to users, for helping them to choose among many different possibilities. Most of these systems only exploit usage data to generate recommendations, disregarding the structural properties of the Web graph. For instance, in [Taghipour N., Kardan A. and Ghidary S.S. 2007], authors propose to adopt a machine learning approach to face the problem, based on reinforcement learning. This system learns to make recommendations exploiting techniques commonly applied in the Web usage mining domain. MUADDIB is different from this approach, since it does not adopt machine learning-based techniques to build the user's profile, being its learning activity based on simply computing an interest value for each category of interest. However, also the interest value computed by MUADDIB can be considered as an example of exploiting usage data to generate recommendation. As another example of using data for recommending, SUGGEST [Baraglia R. and Silvestri F. 2007] is a system capable of dynamically generating personalized contents of potential interest for users of large Web sites, based on an incremental personalization procedure tightly coupled with the Web server. Such a system is able to continuously update the knowledge base obtained from techniques can be applied to enriched Web logs to extract knowledge that can be exploited to generate recommendations. MUADDIB differs from this system since the recommendations are not directly generated by the interaction client-server, but instead it is introduced a fully distributed architecture

to this purpose. Other systems are based on the idea that the importance of each page in a Web graph is defined by the number and the importance of the pages linking to it, as in the case of the popular algorithm PageRank [Brin S. and Page L. 1998]. For example, in [Eirinaki M. and Vazirgiannis M. 2005; 2007], authors propose the Usage-based PageRank (UPR), a PageRank-style algorithm which combines usage data and link analysis techniques to assign probabilities to Web pages based on their importance in the Web sites navigational graph, and applies this algorithm for generating online Web page ranking and recommendation. Experimental results show that this approach produces more objective and representative predictions with respect to pure usage-based approaches. With respect to the aforementioned algorithm, our system exploits a usage-based approach that generates both content-based and collaborative filtering recommendations, thus MUADDIB can be considered as a hybrid system. However, the focus of this paper is not on the recommendation algorithm, since it is possible to use in our system any existing algorithm, either usage-based or PageRank based. As highlighted in Section 1, the contribution of the paper is in the use of a distributed architecture to improve the system efficiency and in the support of multi-device environments. Then, besides the algorithmic issue, below we discuss Web recommender systems under three main perspectives, namely the distributed/centralized architecture, the support of multiple devices and the clustering activities. In particular, in subsection 2.1 we present a qualitative comparison between some distributed recommender systems and MUADDIB, pointing out differences and similarities. Other quantitative comparisons will be presented in Section 6. Since any distributed systems have been proposed in the past that support recommendations for different devices, we analyze in subsection 2.2 some recommender systems which are not distributed, but however deal with the use of multiple devices. Obviously, such a comparison must take into account that a no distributed recommender system is generally conceived to support the user in the visit of a unique, given Web site, while MUADDIB (like the distributed systems that we describe in subsection 2.1) supports the user with respect to his global navigation on different Web sites. Finally, in order to make more complete our description of the context in which our work is placed, in subsection 2.3 we describe some recommender systems that combine clustering and collaborative filtering, similarly to MUADDIB, highlighting the main differences between these systems and our approach.

2.1 Distributed Recommender Systems

Distributed systems provide the possibility to effectively exploit independent computational resources to perform onerous tasks guaranteeing transparency, accessibility, openness and scalability [Tanenbaum A.S. and Van Steen M. 2001]. In particular, P2P architectures allow a decentralized organization of the resources, drastically improving efficiency in many cases [Aberer K. and Hauswirth M. 2004; Aberer K., Alima L.O., Ghodsi A., Girdzijauskas S., Haridi S. and Hauswirth M. 2005]. Although these characteristics of distributed systems seem to be very suitable for being applied to recommender systems, only a few of distributed recommenders have been proposed in the past. As an example, **EC-XAMAS** [De Meo P., Rosaci D., Sarné G.M.L., Ursino D. and Terracina G. 2007] is an XML-based and adaptive distributed systems for supporting a customer, visiting a Web site,

in the search of items present therein and appearing to be appealing accordingly to his past interest and behaviour. This system is distributed and adaptive with respect to both the user's profile and the device exploited by the user for visiting the site, similarly to MUADDIB, but it does not consider the effects of using different devices in the generation of recommendations, while MUADDIB is adaptive with respect to the exploited device. As another example, **Push!Music** [Jacobsson M., Rost M. and Holmquist L.H. 2006] is a model where media (e.g. music files) are autonomous entities that carry their own individual information. In this context, the authors explores how collaborative filtering-like behaviour could emerge out of large ensembles of interacting user clients, which are distributed over mobile devices in social networks. Similarly to MUADDIB, also this system gives the possibility of considering different devices in the generation of recommendations however, differently from our approach, in PUSH!MUSIC the recommendations are not adapted with respect to the exploited device. The recommender system **IMPLICIT** [Birukov A., Blanzieri E. and Giorgini P. 2005] combines distributed systems and traditional recommender system techniques, like our approach. Similarly to MUADDIB, also in IMPLICIT personal assistants associated with the users communicate and collaborate in order to produce suitable recommendations in the context of the current community. As another similarity with our system, also IMPLICIT attempts to learn the user's interests and preferences from the observations of his behaviour. However, the recommendations generated by IMPLICIT do not take into account the exploited device and the recommendation costs strongly depend on the number of users, while in MUADDIB this cost is constant. **CBCF** [Melville P., Mooney R.J. and Nagarajan R. 2002] (Content-Boosted Collaborative Filtering) uses a content-based predictor to enhance existing user data, to exploit collaborative filtering for generating personalized suggestions. The content-based approach views content information as text documents, and user ratings as one of six class labels. Differently from MUADDIB, that uses for the collaborative filtering a purely similarity-based approach, the collaborative filtering component of CBCF uses a neighborhood-based algorithm, where a subset of users similar to the active user, and a weighted combination of their ratings is exploited to generate recommendations for the active user. Also the XML-based distributed system **X-Compass** [Garruzzo S., Modafferi S., Rosaci D. and Ursino D. 2002], similarly to MUADDIB, constructs and manages a user's profile; such an activity is performed automatically, by monitoring the behaviour of a user during his accesses to Web pages. After the user's profile has been constructed, X-Compass exploits it for performing both content-based and collaborative filtering recommendation activities but, differently from MUADDIB, the device is not considered in the generation of recommendations.

Similarities and Differences with MUADDIB All the aforementioned distributed recommender systems exploit, similarly to MUADDIB, an internal profile for storing information related to the user. As for the differences with MUADDIB, we point out that: (i) All the systems store such a global user's profile in a unique device client that supports the user and that performs the task of constructing and updating the profile, while MUADDIB stores the profile into a profiler (a machine with large computational and storing resources) that constructs it on the basis

	Client Global Profile	Server Global Profile	Support Mult. Devices	Device-based Recommender
MUADDIB	No	Yes	Yes	Yes
PUSH!MUSIC	Yes	No	No	No
IMPLICIT	Yes	No	No	No
EC-XAMAS	Yes	No	Yes	Yes
CBCF	Yes	No	No	No
X-COMPASS	Yes	No	Yes	No

Table I. Comparative table between different distributed recommenders

of the information coming from the different device assistants associated with the different user's devices; this leads to the advantage that the client assistants only collect information about the user while the effective profile construction is left to the profiler, making more soft the task of the single client; (*ii*) only EC-XAMAS and X-COMPASS, among the above described systems, consider the effect of using different devices in the construction of the profile and only EC-XAMAS takes into account the exploited device in the recommendation algorithm; instead MUADDIB uses this information in the collaborative filtering and also its content-based recommendations are influenced by the consideration of the device in the profile construction. It is also important to point out that EC-XAMAS and X-COMPASS compute the recommendations entirely on the client-side, involving significant problems for devices having limited computational resources, while MUADDIB exploits the distributed architecture to drastically decrease the computational costs. Table I summarizes these similarities and differences between the described distributed recommenders.

2.2 Recommender Systems that are not distributed and support the presence of multiple devices

PocketLens [Miller B.N., Konstan J.A. and Riedl J. 2004] is a peer-to-peer collaborative filtering algorithm which tackles the portability limitation of current recommender systems. Under this perspective it is similar to MUADDIB, since portability refers to the user's preference to be able to obtain good recommendations whenever and wherever he is. To this purpose, users should be able to run the recommender system efficiently on architectures with slow processors and limited memory, such as palmtop computers. As another example of recommender that can run on multiple devices, **Daily Learner** [Billsus D. and Pazzani M.J. 2000] is a learning agent that helps users to access interesting news stories. This agent has been implemented on 3Coms Palm VII organizer, that is an example of a wireless information device, but it is possible to generalize the approach to other devices such as cell phones or two-way pagers. As in MUADDIB, also Daily Learner deals with the issue of delivering the same content to different devices; however, differently to MUADDID, the effect of the device in the generation of the recommendation is not taken into account.

Similarities and Differences with MUADDIB. Differently to MUADDIB, PocketLens and Daily Learner are not distributed systems. Like MUADDIB, both

systems are suitable to be used for generating recommendations on devices that have limited capabilities, as palmtops and cellular phones. PocketLens does not deal with the adaptivity of the presentation on the site side, while Daily Learner and MUADDIB try to face this issue. Differently from PocketLens, both Daily Learner and MUADDIB allow the Web site to generate recommendations that take into account the exploited device, and in addition to visualize these recommendations in a format adapted to the preferences of the user when he exploits that device. PocketLens requires the user to explicitly rate the items, while MUADDIB and Daily Learner use a technique to automatically detect the interest of a user in an item by simply observing the user's navigation. Finally, we remark that: (*i*) Daily Learner uses a client-server architecture, i.e. a fully centralized approach; (*ii*) PocketLens still needs a central server to synchronize with from time to time hence it could be said to be quasi-decentralized; (*iii*) MUADDIB is fully decentralized.

2.3 Recommender Systems Combining Clustering and Collaborative Filtering

Several approaches have been proposed in the past to exploit clustering techniques for supporting recommendation activities. For example, in [Li Q., Kim B.M. and Myaeng S.H. 2005], authors remark that clustering techniques can be used to classify users or items into user groups where users share similar preference or item groups where items have similar attributes or characteristics, and propose a method of probabilistic model estimation for collaborative filtering, where objects (users or items) are classified into groups based on the content information and ratings at the same time and predictions are made considering the Gaussian distribution of ratings. In [Díez J., del Coz J.J. Luaces O. and Bahamonde A. 2008], a new algorithm to build clusters of people with closely related tastes is presented, and hence people whose preference judgment sets can be merged in order to learn more reliable ranking functions. Authors argue that these clusters can be seen as market segments that demand different kinds of products, and illustrate the applicability of the approach to a collaborative filtering situation. As another example, in [Symeonidis P. , Nanopoulos A., Papadopoulos A.N. and Manolopoulos Y. 2008] authors propose to apply clustering techniques to both users and items, by grouping them in both dimensions simultaneously. They propose a novel nearest-biclusters algorithm, which uses a new similarity measure that achieves partial matching of users' preferences. Finally, [Kim K.-J. and Ahn H. 2008] introduces a novel clustering algorithm based on genetic algorithms (GAs) to effectively segment an online shopping market, showing the usefulness of the proposed model for recommendation systems. All the aforementioned techniques introduce clustering techniques in the computation of collaborative filtering recommendations, generally to improve efficiency thanks to partitioning users (and/or items) in different homogeneous clusters. This idea is also exploited in MUADDIB, where clusters of users having similar interests are built. However, the main difference between MUADDIB and the aforementioned clustering-based recommendation techniques is that in MUADDIB the clustering activity is periodically performed off-line, independently of the computation of the recommendations, exploiting a distributed architecture to completely separate the activity of clients and sites from the activity of clustering managers. Obviously, this implies that MUADDIB computes the recommendation on the last available cluster configuration, while the traditional techniques update the clusters each way a

new information is entered in the system, and then generate the recommendations. This introduces an approximation in the recommendations produced by MUADDIB, that is the price to pay for obtaining a high efficiency of the recommendation activity. However, this price can be arbitrarily reduced by increasing the number of the recommenders, i.e. fully exploiting the possibilities provided by the distributed systems.

3. MUADDIB KNOWLEDGE REPRESENTATION: THE USER'S PROFILE

In order to generate recommendations, MUADDIB builds a *personal profile* of each user of the system. This profile represents the user's interests, i.e. those categories of objects that the user considers the most attractive when navigating on the Web. In the MUADDIB environment, a category simply identifies a set of objects (texts, images, media, etc.) contained in Web pages. In its turn, a item is an actual object contained in Web pages. We assume that all the objects contained in the pages of a Web site are collected into a *site catalogue*. Consequently, each item is an element of the site catalogue. In MUADDIB, all users share a *common dictionary* of categories, publicly available, that represents the set of all the categories with which the users can deal. The profile of a user is a data structure that stores all the categories of the common dictionary in which the user has been interested during his navigation on different Web sites. In particular, the profile associates with each category two numerical values, namely: (*i*) an *interest rate*, denoted by *ir*, that quantitatively represents the interest rate that the user gives to that category and (*ii*) a *last date*, denoted by *ld*, that represents the last date on which the user has shown interest in that category. The interest rates will be used by the system in order to select the more interesting categories for the user, while the last date values allow to periodically prune from the profile those categories no longer accessed by the user. Then, following many user modeling approaches (see, for instance, [Chai W. and Vercoe B. 2000; Joerding T. 1999; De Meo P., Rosaci D., Sarné G.M.L., Ursino D. and Terracina G. 2007; Chi E. H. and Mytkowicz T. 2007]), we use a set of triplets containing the above data to represent a user's profile. More formally:

Definition 3.1 *user's profile.* A *user's profile* is a set of l triplets $\langle c, ir, ld\rangle$, where l is the number of categories contained in the profile, c is a category, ir represents the user's interest in c and ld is the last date on which the user accessed c.

As we will describe in details below, MUADDIB provides each device exploited by a user u with a software client, called *device assistant*. This client stores in the device a user's profile that has the structure described above and that contains all the categories accessed by u using that device.

In order to choose a suitable indicator to represent the user's interest in a category, we have considered several approaches proposed in the literature [Badi R., Bae S., Moore J.M. Meintanis K., Zacchi A., Hsieh H., Shipman F. and Marshall C.C. 2006; Al Halabi W.S., Kubat M. and Tapia M. 2007; Kim H.-R. and Chan P.K. 2008; Claypool M., Le P., Wased M. and Brown D. 2001; Goecks J. and Shavlik J. 2000; Middleton S.E., Shadbolt N.R. and De Roure D.C. 2003; Kelly D. and Belkin N.J. 2004]. In [Badi R., Bae S., Moore J.M. Meintanis K., Zacchi A., Hsieh H.,

Shipman F. and Marshall C.C. 2006], the reading time of a document is recognized as a good indicator of the user's interest, together with other secondary information depending on the interaction with the user, as the number of performed clicks and the inversions of the scrolling direction. Also in [Al Halabi W.S., Kubat M. and Tapia M. 2007], authors argue that the time spent on a Web page is sufficient to infer the user's interest, while in [Claypool M., Le P., Wased M. and Brown D. 2001] different types of indicators are proposed, as the number of mouse clicks and explicit ratings provided by the users. All these approaches agree with the particular importance of considering the reading time for computing the user's interest, therefore we have decided to adopt this solution also in our approach. In our case, the focus is on categories instead of Web pages, where a category is a category of interest. Consequently, in our system the device assistant computes the interest rate ir associated with each category c by measuring the time t spent by the user on the instances of that category during his navigation. The value t is considered as a rough measure of the user's interest in the category and it is strictly related to the characteristics of the exploited device. We also consider, in order to better evaluate the user's interest in the category, the possible actions that he can perform on the items. More in detail, since u can store, print or simply read the Web page that contains c, this is taken into account by weighting ir with a coefficient ρ_a for each action a, where $a = 1$ (resp. $a = 2$, $a = 3$) represents the action of visiting (resp. storing, printing).

The ir of a category which has not been accessed by the user in the past is equal to 0. When the user accesses to an instance of the category, and he spends a time t on this instance, possibly performing an action a, then the value of the ir associated to the category is updated by using the following formula:

$$ir = \begin{cases} (ir + \frac{t}{T} \times \rho_a)/2 & \text{, if } t < T \\ (ir + \rho_a)/2 & \text{, elsewhere} \end{cases} \quad (1)$$

In words, ir is computed as the mean value between the previous value of ir and the new contribution given by the current access. This contribution is assumed to be maximum, i.e. equal to 1, if the time spent by the user on the item reaches a threshold value T. Otherwise, it is equal to the ratio $\frac{t}{T} \times \rho_a$, weighted by the parameter ρ_a to take into account the importance of the action performed by the user. Both the threshold T and the parameters ρ_a are stored into the device assistant, and they can be set by the user.

The device assistant also stores, for each category c, the last date of access ld. This value is used to periodically decrease the ir value of the no longer visited categories, based on the temporal distance from the last date ld. However, the device assistant is not the only software component of MUADDIB that constructs and maintains a profile of its user. Indeed, each user u is also provided with another software, called *profiler*, that runs on a server machine and stores a user's profile called *global user's profile*. Such a user's profile contains all the categories accessed by the user u using whatever device, and the profiler computes the *global interest rate gir* and the *global last date gld* associated with each category based on the corresponding values computed for the same category by all the user's device assistants. More in particular, if ND is the number of device assistants that are

exploited by a user, then the user's profile assistant computes, for each category c accessed by the user, the global interest rate equal to the weighted sum of all the interest rates computed by the user's device assistants. In order to weight each interest rate coming from a device, it is adopted the *cost* for using the device, that is measured by the *price per kilobyte* related to the Internet connection of the device. More formally:

$$gir = \frac{\sum_{i=1}^{ND} \epsilon_i \times ir_i}{\sum_{i=1}^{ND} ir_i} \tag{2}$$

where ir_i is the interest rate computed for the given category c by the i-th device, $i = 1, ..ND$, and ϵ_i is the device cost of the i-th device.

In its turn, the global last date of the category c is computed as the maximum value among the ld values associated to c and stored in the different device profiles. Formally:

$$gld = max\{ld_i\}_{i=1}^{ND} \tag{3}$$

where ld_i is the last date of access computed for the given category c by the i-th device, $i = 1, ..ND$.

More details on the device assistant and profiler will be provided in subsections 4.1 and 4.2.

4. THE MUADDIB ARCHITECTURE

This section describes the distributed architecture of the system MUADDIB, that is capable of performing the two main tasks that we have introduced in Section 1, namely:

(1) supporting the user during his Web navigation by generating personalized suggestions that help him to choose the next Web page to visit.
(2) supporting the site manager to generate a site presentation in a format suitable for the device currently exploited by the user that is visiting the site.

MUADDIB is a platform in which both users and Web sites can be registered. When a user or a Web site's manager is registered, the platform assigns a unique identifier to him.

Five types of components are defined in MUADDIB. Below we describe in detail each of these types.

As shown in Figure 1, each user's device is associated with a *device assistant* that monitors the user and builds a user's profile, containing all the categories accessed by the user exploiting only that device. Then, each user is associated with a *profiler*, running on a server machine, that constructs a complete profile of the user's interests. To this purpose, the profiler collects the user's profiles provided by the different device assistants and computes interest rate and last date values based on the corresponding values present in the device user's profiles. In MUADDIB, the profilers associated to different users, are clustered in partitions. This clustering activity is periodically performed by a *clustering manager* (that is not represented in Figure 1), based on both the domain of interest and the similarity between the

users' global profiles. Since a user can be interested in one or more domains, each profiler can belong to one or more partitions. In order to manage these partitions, each partition is associated with a *recommender* that runs on a server machine and that is able to determine similarities between the profilers of the partition. However, the recommender of a partition does not have this only capability. Indeed, for each Web site of the MUADDIB community that has been visited at least one time by a user of the partition, the recommender of the partition contains a complete list of the categories of the site and, for each user of the associated partition, a list of the categories of the site accessed by that user. This way the recommender has a detailed picture of both the user's interests and the contents of the visited sites. The information about the content of a Web site are provided by a *site manager* associated to each Web site. This software is also capable of generating a suitable presentation of the site for the user that accesses the site exploiting a given device. In the next subsections we deal in details with each of the five components that compose the MUADDIB system.

4.1 The Device Assistant

We associate a device assistant with each device exploited by the user. During a Web session, the device assistant stores some device information and updates the user's profile based on the visited categories. We describe below both the data structure and the behaviour of the device assistant.

4.1.1 *Device Data Structures.* The device assistant internally stores two data structures, namely the *Device Profile* (DP) and the *Device user's profile* (DUP).

DP contains some parameters that characterize the user that is exploiting the device. These parameters can be set by the user and allow him to specify the partitions to which the user desires to belong (i.e., his domains of interest), his preferences about the visualization of multimedia content, the weights to assign to the different actions when computing the interest rate, the preferences about the periodical pruning of the profile and his preferences about the recommendations to be generated. Below we report a complete list of the parameters contained in DP, together with their meanings.

—R_Set: it is the set of recommenders associated to the partitions which u belongs to;

—$MSSet$: it is the *Maximum Size Set*, containing three parameters that represent the maximum size (in Kbyte) of text, audio and video contents, respectively, that u desires to handle when using this device;

—ρ_1, ρ_2, ρ_3: they are three parameters that belong to the interval $[0, 1]$, associated to the actions performable by the user (i.e., visiting, storing or printing a Web page);

—T: it is a threshold coefficient used to evaluate the u's interest in a item by the formula 1;

—P: it is the *attenuation period* expressed by the number of days between two consecutive user's actions after which the interest for a no longer visited category decreases;

—ψ: it is a parameter used to decrease each P days the interests of the user related to categories that are no longer accessed;

—$maxCB$, $maxCF$, $maxInstCB$ and $maxInstCF$: they are parameters exploited by the device assistant in its interaction with the site manager of each visited site (see Section 4.4). In particular, $maxCB$, $maxCF$, $maxInstCB$ and $maxInstCF$ respectively represent: (i) the maximum number of categories belonging to the visited site that the user desires to be considered as content-based recommendations; (ii) the maximum number of categories that the user desires to be considered as collaborative filtering recommendations coming from each other user; (iii) the number of maximum items that the user desires to be visualized as content-based recommendations; (iv) the number of maximum items that the user desires to be visualized as collaborative filtering recommendations.

It is important to point out the importance of the graphical adaptivity of the content to the exploited device. In order for the site manager to generate a site presentation in a format suitable for the device currently exploited by the user visiting the site, there are a lot of content formatting/adaptation and content management issues that would need consideration. It's not easy through only automatic means to get meaningful content and structure from desktop purpose Web pages to a mobile device. Many mobile services spend much resources simply on this issue. In our paper, we have not the possibility to deeply deal with this issue, since our focus is on the mechanism for generating the recommendations also taking into account the device. We point out that our device assistant can store into a device profile some parameters that represent the user's preferences with respect to the content to be visualized by the exploited device, and we provide an example of these parameters (i.e. the maximum sizes of text, audio and video content) but we do not presume to give a complete picture of the problem. However, the mechanism we have proposed for filtering the recommendations based on the requirements contained in the device profile can be extended from our simple parameters to whatever set of other, more complex, information, as those proposed in the related literature on this issue (see, for example, [Mourland C. and Germanakos P. (Eds) 2007; Lee W.-P. 2004; Baus J. and Krüger A. and Wahlster W. 2002]).

The Device user's profile DUP of the device assistant is the user's profile as obtained and gathered on that device. It contains a set of triplet $\langle c, ir, ld\rangle$, where c is a category accessed by the user in the past exploiting the device associated to the device assistant and ir (resp. ld) is the interest rate (resp. the last date of access) associated to the category. The value ir is computed by using the formula 1, while ld is computed by simply storing the access date each time the user accesses to c.

4.1.2 *Device assistant Behaviour.* The device assistant supports the user as follows: (i) in order to construct the device user's profile DUP, it monitors the Web navigation sessions of the user, recording those categories that have been accessed by him. The device assistant periodically sends the device user's profile DUP to its profiler. (ii) When the user visits a Web site, the device assistant sends to the site manager the parameters related to the exploited device, contained in the device profile DP, to generate a personalized presentation of the Web site for the

user. (*iii*) In order to take in account the "age" of the interest rate, each P days the device assistant updates the interest rate values ($ir = \psi \cdot ir$) associated to each category c .

4.2 The Profiler

Each user is associated with a profiler that collects by each user's device assistant the information about the categories visited during the user's Web activities. These information are sent to the recommenders of the user's partitions. This is an important feature of MUADDIB, since the device assistants live on the associated devices and could have limited computation and storage capability. The contribution of the profiler, which runs on a more equipped machine, is fundamental to provide the user with an off-line collector of all the information obtained by the different device assistants which have monitored the user navigation. Below, both the data structure and the behaviour of the profiler are described.

4.2.1 *Profile Data Structure.* The profiler contains two data structures, namely the *Profile Setting* (PS) and the *Global User's Profile* (GUP).

In its turn, the Profile Setting PS contains those parameters that allow the computation of the global interest rate of each category following the formula 2, namely:

—ND: it is the number of device assistants associated with the user;

—ϵ: it is a vector containing ND elements, where each element ϵ_i is the cost of the Internet connection of the i-th device, measured in price per Kbyte.

The global user's profile GUP contains a set of triplet $\langle c, gir, gld \rangle$, where c is a category accessed by the user in the past and gir (resp. gld) is the global interest rate (resp. the last date of access) associated to the category, computed by using the formula 2 (resp. 3).

4.2.2 *Profiler Behaviour.* The behaviour of the profiler consists in updating the global user's profile GUP by exploiting the data that each device assistant of the user periodically sends to the profiler. These data consist, for each category c visited by the user with a given device i, of a pair of the form $\langle c, ir_i, ld_i \rangle$. If c also occurs in the GUP, ir_i (resp. ld_i) is immediately exploited to update the global gir (resp. gld) of the category; elsewhere, if c is a new category, for the first time visited by the user, a new element is added in GUP.

4.3 The Clustering Manager: Formation of the User Partitions

The global user's profile GUP is used to partition the n users of the system in p different groups. Users are partitioned by a two-level clustering. A first clustering is based on the *domain of interest* of each user. We consider a fixed number NDI of domains of interest (e.g. *book*, CD, *computer*, etc.), and we denote by $\mathcal{D}$ the set of the domains. Each user can be interested in one or more of these domains. We denote by G_i the group of the users interested in the i-th domain, $i = 1, 2, ..NDI$.

When a new user u joins the first time with the system, he has to specify one ore more domains of interest $u_1, u_2, ..u_g$ belonging to $\mathcal{D}$. Based on this choice, the user is placed in the groups $G_{u_1}, G_{u_2}, .., G_{u_g}$.

This means that the groups are not disjoint, since two different groups can have some element in common.

In this first phase, we have obtained to group different users based on similar interests, since all the users of a group are interested to the same domain. We fixed a threshold k for the cardinality of a group, where k is an integer number, over which the group needs to be further partitioned. In other words, if the cardinality of the group G_i is lesser than or equal to k, the group is not further partitioned, otherwise the group has to be partitioned since it is considered too large.

For each of these group G_i, a *Clustering Manager* CM_i associated to G_i periodically verifies if the cardinality of G_i is greater than k and in the positive case it performs a clustering of the users belonging to G_i, based on the similarity between the global user's profiles. This operation allows to further refine, for those groups having of a significant cardinality, the simple initial grouping that was only based on the domain of interest.

More in particular, the clustering manager computes a *similarity matrix* Γ, where $\Gamma[u][q]$, $u, q = 1, 2.., n$ contains a measure of the similarity between the global profiles of the users u and q.

In order to determine the similarity between the global profiles of u and q, we propose to exploit the Jaccard similarity measure [Grefenstette G. 1994]. Generally, the Jaccard similarity measure between two sets A and B is defined as the number of elements shared by A and B, divided by the total number of unique elements in both A and B (i.e., it is equal to $\frac{|A \cap B|}{|A \cup B|}$). In our approach, we define the similarity measure between two global user's profiles as the Jaccard similarity measure between these global user's profiles.

Definition 4.1 *Similarity value. Let GUP_u, GUP_q the global user's profiles of the user u and q, respectively. The* similarity value *between GUP_u and GUP_q is defined as:*

$$\sigma_{uq} = \frac{|GUP_u \cap GUP_q|}{|GUP_u \cup GUP_q|}$$

After the similarity matrix Γ is constructed, the clustering manager is now able to perform a clustering of the set D_i. We denote as l the number of different clusters that we want to obtain for each domain of interest D_i, and we denote as CL_i^j, $j = 1, 2, .., l$ the j-th cluster of the i-th domain. Obviously, the total number of partitions p is equal to the product of NDI times l.

Cluster construction is obtained by performing any clustering algorithm that exploits a similarity matrix. As an example, in the experiments of Section 6, we use the well-known algorithm *CLARANS*, based on the k-medoids method [Ng R.T. and Han J. 2002], since it performs more efficiently on large data set than other classical algorithms as PAM and CLARA. The cost for computing the similarity matrix is $\mathcal{O}(\delta^2 \cdot m^2)$, where δ is the maximum number of users belonging to the same partition and m is the largest dimension of site catalogue. In fact, we remark that it is necessary to compute the δ^2 elements of the matrix, and each of this element is the Jaccard similarity that can be computed in $\mathcal{O}(m^2)$. The cost for computing the clusters using CLARANS is $\mathcal{O}(\delta^2)$ [Berkhin P. 2002]. Consequently, the overall cost of the clustering computation periodically performed by the clustering manager is $\mathcal{O}(\delta^2 \cdot m^2)$. As noticed above, when a new user u joins with the system, it

is assigned for the first time to the groups $G_{u_1}, G_{u_2}, .., G_{u_g}$. One or more of these groups could be already clustered, in the case they have a cardinality greater than k. Let G_u be one of the groups to which u has been assigned and that is already clustered in the subgroups. In this situation, the user u is randomly assigned to a subgroup, without performing any update of the clustering. This means that the clustering activity of the clustering manager of the group is performed only at pre-fixed instants of time, in off-line mode, completely independent of the users' activity. This is a crucial point of our method. Since the clustering activity and the users' activities are performed independently, the cost of the clustering activity does not influence the efficiency of the users' activities, in particular the generation of the recommendations. The price to pay for this efficiency is that the recommendations are generated on the current configuration of the clusters, that cannot be the best configuration since the clustering is executed periodically and not each time a modification is produced in the system. However, since the first grouping of the users are performed based on the domains of interest, we are sure that the users of a cluster are homogeneous from the viewpoint of the domain of interest. This leads to accept, as a reasonable approximation, to use the results of the last clustering activity, performed in the last fixed period of re-clustering, instead to recompute the clustering each way the system is modified. Obviously, if the temporal period tc between two consecutive re-clustering instances increases, the activity of the clustering manager becomes less onerous. However, if tc is too large, the approximation introduced in the recommendations can appear unacceptable. Our method allows the system administrator to fix a suitable value for tc, taking into account the number of available clustering managers (i.e. available groups of users). If this number is large, since each clustering manager runs on a different machine, the period tc can be fixed sufficiently small. In fact, in this case each clustering manager will handle a little number of clusters, being the users strongly distributed on the large number of available groups. In this case, the cost $\mathcal{O}(n^2 \cdot m^2)$ of the clustering activity is little enough (in presence of a small value for n) for allowing the clustering manager to perform the re-clustering with a high frequency. Vice-versa, if the number of available clustering managers is small, each clustering manager will operate on a large number of users n, and the cost of the re-clustering will become onerous. In this situation, the system administrator will fix a large value for tc, so limiting the frequency of re-clustering and paying the price of degrading the performances of the recommendation activity.

4.4 The Recommender and the Site manager

Two other types of MUADDIB components are the *recommender* and the *site manager*. These two types of components strictly collaborate with each other in order to propose recommendations to the user, then we describe them together in this section. Each recommender is associated with a partition of users. Each site manager is associated with a Web site in order to manage the site content. Below, the data structure of the recommender and the behaviours of recommender and site managers, that interact together, will be briefly described. We omit to describe the structure of the site manager since it contains only the site catalogue. Note that the behaviours of recommender and site managers represent our recommendation algorithm, that we will describe in form of pseudo code in Figures 5 and 6.

4.4.1 *Recommender Data Structure.* The recommender contains two data structures called *Site Catalogue* (SC) and *Device Profile Set* (DPS) .

The site catalogue SC contains, for each site W that has interacted with the recommender in the past, a copy of the catalogue CAT_W that stores all the items present in W. Each catalogue CAT_W is periodically updated by the corresponding site manager of W. The device profile set DPS contains the device user's profiles of all the users associated with the recommender. More in particular, DPS is a list of elements. Each element, denoted by $DPS[u, d]$ represents the device user's profile of a given user u associated to his device d. The data structures described above are exploited by the recommender to compute two matrices, denoted by LCB and LCF, that contain pre-computed recommendations for the users, that will be provided by the recommender when a site manager will request its help to support the visit of a user. More in particular, each element $LCB[W, u]$ of LCB is a list of categories, associated to a site W and a user u, that contains all the categories that the recommender has determined to be the most interesting for the user u in the catalogue of the site W. Analogously, each element $LCF[W, u, d]$ of LCF is a list of categories, associated to a site W, a user u and a device d. It contains all the categories that the recommender has determined to be the most interesting for the users that have visited in the past the site W exploiting the device d and that are the most similar to the user u in terms of interest.

4.4.2 *Recommender and Site Manager Behaviour.* Figure 3 provides a graphical representation of the method exploited by MUADDIB in order to generate recommendations. Each Web site W is associated with a site manager s. Suppose that the user u visits the site W by exploiting a given device; then the device assistant d of u sends to the site manager the device profile DP. Moreover, suppose that the user u belongs to NP partitions, each of them associated to a recommender r_i, $i = 1, .., NP$. In this case, the site manager contacts each recommender r_i and sends to it the device profile DP. In its turn, each recommender r_i returns to the site manager two lists of categories, denoted by CB_i and CF_i, which store the content-based and the collaborative filtering recommendations, respectively, that r_i has computed. Finally, the site manager constructs two global lists of categories, CB and CF, to be provided to the user u as suggestions, where CB (resp. CF) contains the union of all the CB_i (resp. CF_i) lists.

The behaviour of the recommender is graphically represented in Figure 4. As we can see, in order to generate content-based recommendations (represented by the box CB), the recommender exploits the matrix LCB that is periodically precomputed based both on the information that comes from the profiler of each user and the information provided by the site manager of each site. In particular, each element $LCB[W, u]$ of the matrix LCB is a list of categories associated with a user u and a site W and contains those categories that the recommender provides to the user u when visiting the site W. In order to compute $LCB[W, u]$ the recommender selects those categories of the site W that belong to both the device user's profile $DPS[u, d]$ and the site catalogue of W (the site catalogue is stored as an element of the site catalogue set SC). Then, the recommender orders these categories in a decreasing fashion based on the coefficient ir of each category.

Finally, the recommender uses the list $LCB[W, u]$ to construct the list CB that

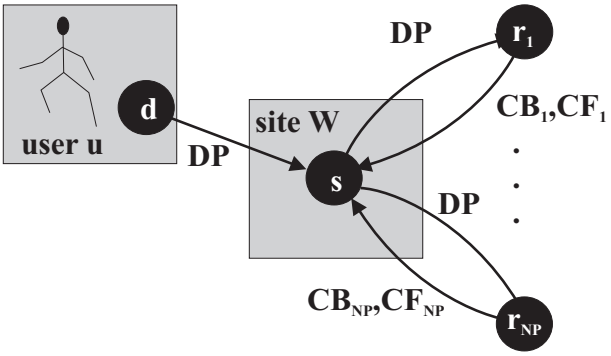

Fig. 3. The recommendation mechanism

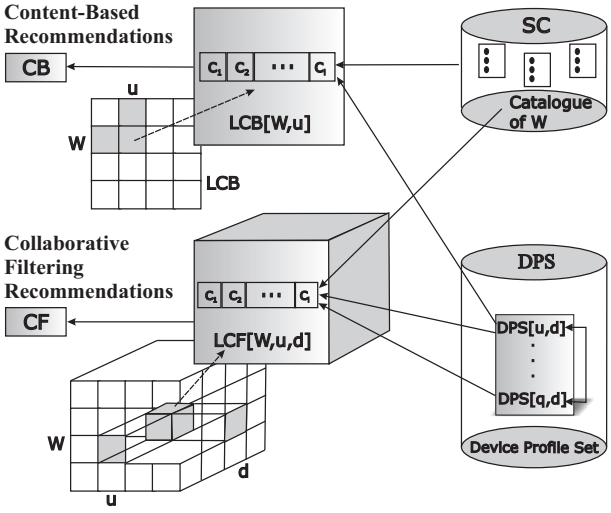

Fig. 4. The behaviour of the MUADDIB recommender

represents the content-based recommendations for the user u. These recommendations are generated when the site manager contacts the recommender and informs it that the user u needs recommendations. Consequently, the recommender considers the categories of the list $LCB[W, u]$, which have the highest value of interest rate, and it inserts into CB at most $maxCB$ categories present in both $LCB[W, u]$ and the site catalogue of W (remember that $maxCB$ is a parameter contained in the device profile DP). Analogously, in order to generate collaborative-filtering recommendations, the recommender periodically updates the matrix LCF. In particular, each element $LCF[W, u, d]$ is associated with a site W, a user u and a device d and contains those categories of the site W that the recommender suggests to the user u when visiting the site exploiting the device d. These categories are the result of a collaborative-filtering phase performed by the recommender. Indeed, the recommender compares the device profile $DPS[u, d]$ of the device profile set, with the

device profile $DPS[q,d] \in DPS$ associated to each other user q, that has exploited the same device d. The result of this comparison is the value $ID(u,q,d)$, a measure of the global *interest difference* of the users u and q with respect to the categories of W (see subsection 4.4.2.3 and formula 4 for the details about the computation of $ID(u,q,d)$). The recommender orders the users in a decreasing order with respect to the value $D(u,q,d)$ and for each user q inserts in $LCF[W,u,d]$ at most $maxC$ categories that q considers the most interesting based on the ir value (remember that $maxC$ is a parameter internally stored into the device profile DP). When the site manager W contacts the recommender to obtain recommendations for a user u that is exploiting a device d, the recommender inserts in the list CF at most $maxCF$ categories contained in the pre-computed element $LCF[W,u,d]$ that are in the first positions (where $maxCF$ is a parameter stored in the device profile DP). Finally, both the list CB and CF are sent to the site manager of the site W that uses it in order to generate a suitable presentation for the user u that is visiting the site. More in particular, the site manager visualizes, in a format according with the preferences set in the device profile DP, at most $maxInstCB$ items for each category present in the list CB and at most $maxInstCF$ items for each category present in the list CF (where $maxInstCB$ and $maxInstCF$ are two parameters stored in the device profile DP).

4.4.2.1 ***Updating LCB and LCF Matrices.*** The update of LCB matrix is automatically performed when both the two following conditions are verified:

—The last update of the element $LCB[W,u]$ has been performed at least t_{LCB} minutes ago, where t_{LCB} is an integer value fixed by the system administrator. This condition guarantees that the interval between two consecutive updates of the same element is sufficiently large (at least equal to t_{LCB}). It is important to point out that, if the administrator fixes a value t_{LCB} too small, then the recommender has to execute frequent re-computations that could imply an onerous task: in this case, it is necessary to have a powerful enough machine on which recommender software can quickly run. If t_{LCB} is too large, the updates of the LCB elements are very unfrequent, and this can lead to compute recommendations on obsolete data. Then, the administrator has to carefully fix the value of t_{LCB}, taking into account both the number and the computational capabilities of the available machine that host the recommender.

—As a second condition, some modification has been applied to either the catalogue of the site W or the global user's profile of the user u.

The actual implementation of the two conditions above is realized as follows, on the recommender site. When the recommender verifies that for the element $LCB[W,u]$ the first condition is verified, it sends a message to both the site manager of W and the u's profiler, containing the last date ldc of computation of $LCB[W,u]$. If the site manager of W (resp. the u's profiler) verifies that a change has been applied to the site W (resp. the u's profile) in the period between ldc and the current time, then the site manager of W (resp. the u's profiler) sends to the recommender the updated catalogue (resp. the updated profile). The periodical re-computation of the matrix LCF is performed in an analogous way. In this case, a value t_{LCF} is fixed by the administrator, which represents the minimum period

between two consecutive updates of the matrix. Then, an element of the matrix LCF is updated if these two conditions hold:

—The last update of the element $LCF[W, u, d]$ has been performed at least t_{LCF} minutes ago. In this case, the value of t_{LCF} is crucial since the update of the matrix LCF is a more onerous task than that of the matrix LCB, involving all the users of the community. For this reason, it is necessary to fix a value t_{LCF} more large than t_{LCB}, to avoid too onerous computations.

—As a second condition, some modification has been applied to either the catalogue of the site W or the device user's profile of a user q belonging to the community.

Then, each t_{LCF} minutes, in order to update the element $LCF[W, u, d]$, the recommender sends a message to all the device assistants associated to the device d currently on line, and to the site manager of W, containing the date of the last update of $LCF[W, u, d]$. If some modifications have been applied to the catalogue of W (resp. the device profile of a device assistant), the site manager of W (resp. the involved device assistant) sends to the recommender the updated catalogue (resp. the updated device profile).

4.4.2.2 ***The Recommendation Algorithm.*** Figure 5 shows an implementation, in form of pseudo-code, of the recommendation algorithm described above. The implementation is represented by the function *recommend*, that is called by the site manager of the Web site W when the device assistant of the user u visits the site. This function receives as input the user u, his device profile DP and the site W (remember that in the MUADDIB platform both users and Web sites are registered with a unique identifier). The function *recommend* builds two lists of categories (a list of categories has been represented in the code by the type CList), namely CB and CF. These two lists are initially empty, and they will be computed by the function as the union of the recommendation lists CB_i and CF_i coming from the recommenders, as described in the previous subsection. To this purpose, a function *getLists* is called for each recommender r_i associated to a partition which the user u belongs to (remember that a user can belong to one or more partitions). The function *getLists* is an internal function of the recommender r_i and performs the task of inserting in the lists CB_i and CF_i the pre-computed content-based and collaborative-filtering recommendations. Finally, the function *visualize* presents to the user u a maximum of $maxInstCB$ (resp. $maxInstCF$) instances that are associated to the categories of CB (resp. CF). Moreover, the function adapts the presentation to the parameters contained in $MSSet$. The code of the function *getLists* called by each recommender r_i is shown in Figure 6. This function receives as parameters the user u, the site W, the device d, the device profile DP and the two lists of categories CB_i and CF_i. The parameters CB_i and CF_i are used as output parameters, since they are passed to the function as empty lists, and the function inserts in CB_i (resp. CF_i) the categories selected from the element $LCB[W, u]$ (resp. $LCF[W, u, d]$) based on the method described in the previous subsection and using the parameter $maxCB$ (resp. $maxCF$) contained in the device profile DP. This task is performed by the function $CBinsert$ (resp. $CFinsert$).

```
recommend(user u, device profile DP, site W) {
    CList CB, CF, CB_1,..,CB_p, CF_1, CF_p;
    for each r_i ∈ DP.R_SET
        r_i.getList(u, DP, CB_i, CF_i, W);
    CB = ⋃_{i=1}^{NP} CB_i;
    CF = ⋃_{i=1}^{NP} CF_i;
    visualize(CB, CF, DP.MSSet, DP.MaxInstCB,
    DP.MaxInstCF);
    return;
}
```

Fig. 5. The Recommendation Algorithm

```
getLists(user u, device d, device profile DP, CIList CB_i,
CIList CF_i, site W) {
   CB_i=CBinsert(LCB[w, u], DP.maxCB);
   CF_i=CFinsert(LCF[w, u, d], DP.maxCF);
   return
}
```

Fig. 6. The function *getLists* of the recommendation algorithm

4.4.2.3 ***Computation of the Interest Difference*** ID. The difference between the user u and another user q which exploit the same device d is computed as follows. Let c be a category that belongs to both the device user's profile $DPS[u, d]$ of u and the device user's profile $DPS[q, d]$ of q, and let $ir_u(c, d)$ be the interest rate assigned to the category in $DPS[u, d]$ and $ir_q(c, d)$ be the corresponding interest rate in $DPS[q, d]$. The value $diff(c, d) = |ir_u(c, d) - ir_q(c, d)|$ is assumed to be a reasonably measure of the difference between the two users u and q in the evaluation of the category c. We measure the global interest difference between the two users u and q using the device d, denoted by $ID(u, q, d)$, by summing all the contributions $diff(c, d)$ related to all the categories c that the profiles of u and q share. More formally:

$$ID(u, q, d) = \sum_{c \in DPS[u,d] \bigcap DPS[q,d]} |ir_u(c, d) - ir_q(c, d)| \quad (4)$$

Example 4.2. Let $u1$, $u2$ and $u3$ three users exploiting both a PC desktop and a palmtop. Let $DPS[u1, PC]$ and $DPS[u1, palmtop]$ (resp. $DPS[u2, PC]$ and $DPS[u2, palmtop]$, $DPS[u3, PC]$ and $DPS[u3, palmtop]$) the device user's profiles of $u1$ (resp. $u2$, $u3$) associated with the PC desktop and the palmtop, respectively, as stored in the recommender of the user $u1$. Suppose that these device user profiles contains the elements represented in Figure 7, each associated to an interest rate, where for simplicity we have assumed that all the three users are interested in the same categories, namely books, sport and games. We can compute, applying equation 4, the following values of interest difference:

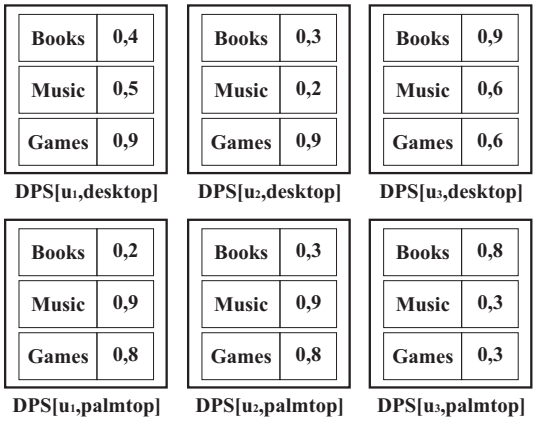

Fig. 7. The profiles of three users stored into a recommender

$$ID(u1,u2,PC) = \frac{0.3+0.6+0.5}{3} = 0.47; \qquad ID(u1,u2,palm) = \frac{0.1+0.0+0.0}{3} = 0.03;$$

$$ID(u1,u3,PC) = \frac{0.0+0.1+0.2}{3} = 0.10; \qquad ID(u1,u3,palm) = \frac{0.6+0.6+0.5}{3} = 0.57;$$

$$ID(u2,u3,PC) = \frac{0.6+0.4+0.3}{3} = 0.43; \qquad ID(u2,u3,palm) = \frac{0.5+0.6+0.5}{3} = 0.53;$$

Now suppose that a recommendation is requested by the site S for the user $u1$ when he exploits the PC desktop, and a maximum of two recommended items have to be provided. For simplicity, we suppose that all the three users have visited two items for each category in the site S. As another simplification, here we focus only on the collaborative filtering component of the recommendations, ignoring the content-based component. In this situation, the recommender finds that the most similar user to $u1$ is $u3$, and then it suggests the two items that $u3$ has visited in the category *books* in the site S, since this is the category having the maximum interest rate. However, if $u1$ is exploiting the palmtop, then the recommender finds that the most similar user is $u2$, and suggests the two items that $u2$ has visited in the category *music* of the site S.

4.4.2.4 ***Considerations on the Effectiveness of the Recommendation Algorithm.*** The MUADDIB recommendation algorithm described above is a variant of the MASHA recommendation algorithm presented in [Rosaci D. and Sarné G.M.L. 2006]. The important difference between the two algorithms is in the computation of both content-based and collaborative filtering recommendations. Indeed, in order to generate content-based recommendations, MASHA compares the global profile of the user with the site catalogue, selecting the common categories. The adaptivity with respect to the device is performed after the selection of the common categories, based on the information contained in the device profile. Therefore, we argue that MASHA produces an *adaptivity with respect to the presentation.*

Instead, the MUADDIB algorithm compares the device user's profile of the device assistant with the site catalogue, so considering the influence of the device in the selection of the common categories. This way, MUADDIB generates an *adaptivity with respect to the category selection*, besides the adaptivity with respect to the presentation (that is always produced, by using the same procedure than MASHA). Similarly, in order to generate collaborative filtering recommendations, MASHA compares the global profile of the user with the global profile of the other users, while MUADDIB operates on the device user's profiles. This leads MUADDIB to generate recommendations that should be more effective than those of MASHA. Indeed, these recommendations are computed by considering the interests of the user with respect to the device that is currently exploiting, while MASHA considers only synoptical information. This improvement has been possible due to the more powerful architecture of MUADDIB with respect to MASHA. Indeed, the introduction of the recommenders allows the (off-line) pre-computation of the recommendations based on the device user's profiles. Differently, the MASHA architecture, that delegates to the site manager the activity of generating recommendations, does not allow to consider the more detailed device user's profiles and limits its precision since it considers the more general global user's profile. As a final consideration, we remark that the idea of using a global user's profile (introduced in MASHA) is already exploited by MUADDIB to compute the different partitions of the user. Indeed, the activity of computing partitions should put in the same partition profilers that are similar with respect to their global interests, and not only with respect to a given device. The use of the global profile seems in this context particularly suitable.

5. THEORETICAL EFFICIENCY OF THE RECOMMENDATION ALGORITHM

The main issue that MUADDIB tries to face consists in the reduction of the time and storage space costs for the site server that has to provide the recommendations for the visiting user. Indeed, as noticed in Section 1, if the time cost is too large, then the user will wait for a too long time before to receive the recommendations. Furthermore, if the server of the site needs a storage space that increases very quickly with the number of users in the system, then in large systems the required dimension of the server space will not be tolerable. In this section we analyze how the time cost and the storage space required to the server of a MUADDIB Web site to perform the recommendation task depend on both the dimension of the system and the dimension of the site catalogue. In this analysis, the dimension of the system is represented by the number of the registered users, denoted by n and the number of the different partitions, denoted by p. The dimension of the site catalogue is represented by the number of different categories present in the catalogue, denoted by m. In order to make our analysis complete, we also consider the connection time necessary for the site manager for sending a message to a recommender. We assume that this connection time, denoted by CT, is constant and independent of the particular recommender. In order to evaluate the time cost of the recommendation task, that is represented by the time cost of the function *recommend* described in Figure 5, we observe that the cost of the function $getLists$ that is called from the p recommenders r_i involved in the recommendation

task is $\mathcal{O}(maxCB + maxCF)$. Consequently, the global time cost of the function *recommend*, can be computed as formally shown below.

PROPOSITION 5.1. *The time cost of the Web site server is* $\mathcal{O}(m + p)$

Proof. Let us consider the function *recommend* of Figure 5. The time cost of this function is the sum of (i) the cost necessary to the NP recommenders to perform all the calls $r_i.getList$ (note that NP is the number of partitions associated to the generic user, while p is the total number of partitions, thus $NP \leq p$), (ii) the cost to compute the union of all the CB_i lists and CF_i lists and (iv) the cost of the function *visualize*. We remark that each of the NP recommenders is a software object that have its own computational resources, independent from the Web site server, since it does not run on the site but on a remote server machine. Therefore, each function $r_i.getList$ is executed in parallel with each other. Consequently, the whole time cost to perform all the calls $r_i.getList$ is the same of the function $getList$, i.e. $\mathcal{O}(maxCB + maxCF)$. However, we have to consider the connection time CT for contacting each recommender. We also assume that the time for receiving a response from the recommender is equal to CT. Considering that the number of recommenders to be contacted is at most p, therefore the whole cost for obtaining all the CB_i and CF_i lists from the recommenders is clearly $\mathcal{O}(p \cdot CT + maxCB + maxCF)$. The cost to compute the union of all the CB_i lists is $\mathcal{O}(m)$ (remember that m is number of the different categories present in the catalogue of the site W). Also the cost to compute the union of all the CF_i lists is $\mathcal{O}(m)$. The cost of the function *visualize* is $\mathcal{O}(maxInstCB + maxInstCF)$, since it visualizes at most $maxInstCB + maxInstCF$ items. Thus, since $maxCB$, $maxCF$, $maxInstCB$, $maxInstCF$ and CT are constant, the time cost of the function *recommend* is $\mathcal{O}(m + p)$.□

This analysis shows that the time cost of the recommendation task of a MUADDIB site manager is independent of the number n of users registered in the system, but only from the number p of partitions and the number of different categories m. This is an important result if compared with the time costs of the other recommender systems proposed in the literature, that strictly depend on the number of registered users. Table II reports the costs of some well-known recommender systems that we have considered in Section 2. It is important to point out that the result proved above is only related to just the site manager activity. Indeed, in our distributed systems there are four different types of components, namely device assistants, profilers, recommenders and clustering managers, and each of them performs its activities independently of the other components. However, the focus of this Section is only on the activity of the site manager, a server that performs the task to provide recommendations to visiting clients. The amount of computational resources requested by the site manager is one of the most important evaluation parameter for any recommender system, since it directly measures the efficiency of the server in generating recommendations, i.e. the amount of time the client has to wait before of receiving recommendations. With respect to this key parameter, Table II shows the better performances of MUADDIB with respect to other traditional systems.

Table II. The time cost of different Distributed Recommender Systems

Recommender System	Time Cost
MUADDIB	$\mathcal{O}(m+p)$
MASHA	$\mathcal{O}(n^2 \cdot m)$
X-COMPASS	$\mathcal{O}(n^2 \cdot m^2)$
EC-XAMAS	$\mathcal{O}(n^2 \cdot m^2)$

However, we remark that this improvement of efficiency is mainly due to the distributed architecture, that allows the site manager to exploit the work of pre-computation performed by the recommenders, as well as the work of the clustering managers in computing the clusters. In order to make complete our analysis of cost, we evaluate below the cost of the task performed by the recommender and the clustering manager, respectively.

PROPOSITION 5.2. *The time cost of the recommender to pre-compute recommendations is* $\mathcal{O}(\omega \cdot l \cdot m^2 + \omega^2 \cdot l \cdot m^2)$*, where* ω *is the maximum number of users belonging to a partition and* l *is the number of Web sites.*

Proof. The cost is the sum of computing the matrices LCB and LCF. For each element $LCB[W, u]$ of LCB, it is necessary to compare all the items belonging to the catalogue of W with all the items belonging to the profile of u, and this cost is trivially $\mathcal{O}(m^2)$. Analogously, for each pair (W, u) in LCF, it is necessary to compare all the items belonging to both the profile of u and the site W with all the items of each other user in the partition, and this cost is $\mathcal{O}(\omega \cdot m^2)$. The overall complexity is obtained by multiplying each of this two costs for $\omega \cdot l$ (the maximum number of pairs (W, u) in each matrix) and then summing the two results. □

PROPOSITION 5.3. *The time cost of the clustering manager is* $\mathcal{O}(\delta^2 \cdot m^2)$*, where* δ *is the maximum number of users belonging to the same partition.*

Proof. The proof simply derives from the analysis performed in Section 4.3.□

The results above show that the significant improvements introduced by MUADDIB in the generation of recommendations is obtained thanks to a considerable computational effort performed by the other components of the system, namely recommenders and clustering managers. Therefore, it is clear that the idea underlying our proposal is that of using some sufficiently powerful machines, each dedicated to a particular task (pre-computing recommendations in the case of the profiler, computing clusters in the case of the clustering manager) that is performed independently of the main activity of the recommender system, that is the interaction from a client that requires suggestions and a server that provides them. We remark that client and server are the critical components of the system. Indeed, the client, in a multi-device environment, can have limited resources, while the server (i.e. the site manager) has to handle many requests in an efficient way.

6. EXPERIMENTS

In this section, we present some experiments devoted to evaluate the capability of MUADDIB to perform both content-based and collaborative filtering activities. To this purpose, we have compared the performances of MUADDIB with MASHA and two other recommender systems, namely EC-XAMAS and X-COMPASS, that we have described in Section 2. This choice is due to the following two motivations: (*i*) Similarly to MUADDIB, these systems are based on the exploitation of a user's profile, built based on the observation of the user's behaviour; this makes more comparable the behaviour of the systems which consists of the same basic steps, i.e. construction of the user's profile and its exploitation when generating recommendation; (*ii*) all these systems builds the user's profile based on the exploited device and both MASHA and EC-XAMAS consider effect of the device in the generation of the recommendations.

6.1 Common Dictionary

In the current implementation of MUADDIB, the common dictionary is realized as an XML-Schema document, where each element represents a *category*. All the sites of the MUADDIB community are XML sites that contains instances of categories that belong to the dictionary. The mechanism for accessing a category is as follows. Each Web page p of a site contains some *hyperlinks* represented by pairs (s, d), where s and d are instances of categories present in the common dictionary. We say that a user, that is visiting the Web page, *accesses* the item d when he clicks on a hyperlink (s, d) present in the page. The hyperlinks have been implemented by using the XML-Schema element *hyperlink*. Such an element is composed of three sub-elements, namely: (*i*) a *textURL* that contains the textual description of the hyperlink (e.g. "Dante Alighieri"), (*ii*) a *destinationCategory* that represents the category which the hyperlink refers to (e.g. *book*), and (*iii*) the *destinationURL* that contains the actual URL pointed by the hyperlink. This implementation of a hyperlink allows the device assistant to automatically detect the category to which the hyperlink points, and therefore understand the user's interest.

6.2 Dataset

For our experiment we have used as common dictionary the Italian-English Literature XML-Schema, publicly available at the MUADDIB site [MUADDIB site URL 2008]. This dictionary contains 94 different categories related to authors of italian and english literature. Moreover, we have exploited a set of 300 different XML Web sites which are based on the aforementioned dictionary (also these sites are downloadable at the MUADDIB site). As we will describe in detail in Section 6.1, the pages of each site are XML documents, and each page contains only instances of the common ontology. Furthermore, we have monitored 200 real (distinct) users, denoted by u_1,u_2,..,u_{200}. To study how the performances depend on the number of monitored users, we have repeated the measures of effectiveness and efficiency on different sub-sets of these 200 users. For this reason, we have considered the sets S1, S2, S3 and S4, where the set S1 contains the first 50 users, i.e. $S1 = \{u_1, u_2, .., u_{50}\}$, the set S2 contains the first 100 users, i.e. $S2 = \{u_1, u_2, .., u_{100}\}$, the set S3 contains the first 150 users, i.e. $S3 = \{u_1, u_2, .., u_{150}\}$ and finally the set S4 contains

Table III. The setting of the MUADDIB device assistants

device assistant	T (sec.)	ρ_1	ρ_2	ρ_3	P	ψ	k
desktop PC	200	0.6	0.8	0.9	3	0.90	4
palmtop	120	0.6	0.9	1.0	3	0.95	4
cellular phone	60	0.5	0.9	1.0	3	0.95	4

all the 200 users, i.e. $S4 = \{u_1, u_2, .., u_{200}\}$. For each user, we have registered in a log record the first 900 accessed URLs, in order to construct the user's profile, while other 600 URLs have been used in the test phase. Each user's access has been represented by a tuple $\langle u, c, t, d\rangle$ where u is the identifier of the user, c is the category associated with the accessed URL, t is the no-idle time spent on the page associated with the accessed URL and d is the exploited device. All the log records have been collected into a log file, that is publicly available at the MUADDIB site.

6.3 Software components

All the software components of our system have been implemented using the JADE libraries [JADE URL 2007]. The JADE framework provides some useful JAVA classes suitable to construct a distributed environment, making available a communication protocol, based on JAVA RMI, that allows the distributed components to send and receive messages, and a Yellow Page service that allows each component to be registered into a common platform. The JADE platform is running on an apposite machine, whose IP address is known to all the components of the system. The distributed architecture is composed by (i) the devices of the users, (ii) the Web server machines that host the Web sites and where the site managers run, (iii) the profiler server machines where the profilers run, (iv) the recommender server machines where the recommenders run, (v) the machine where the JADE platform has been started. Each of the components will be described in detail below.

6.3.1 *Device assistants.* Each user exploits for his navigation a device connected to the Internet, that can be a desktop PC, or a palmtop or a cellular phone. We have requested each user to exploit the PC in about 80 percent of cases, the palmtop in about 10 percent of cases and the cellular phone in about 10 percent of the cases, in order to have a realistic scenario. Each device is provided by a simple JAVA browser http-based, able to visualize the content of a Web page, and integrated with four device assistants, implemented in JADE, corresponding to the systems MUADDIB, MASHA, X-COMPASS and EC-XAMAS. The MUADDIB device assistant operates as described in Section 4.1. In particular, we have set device parameters as shown in Table III. It is important to point out that we have chosen to use for the device parameters the same value proposed in [Rosaci D. and Sarné G.M.L. 2006] for the client component (corresponding to the device assistant in the MUADDIB architecture), that derive from an analysis of sensitivity of the system performances with respect to these parameters. This choice allows us to compare our system with the best configuration of parameters determined for MASHA. However, we remember that the interest for a category has been assumed as "saturated" if the visit time of the category is higher than T seconds. The coef-

ficient ρ_1 (resp. ρ_2, ρ_3) weights the user's interest in a category in the case the user simply visits (resp. stores, prints) a page containing an instance of that category. Moreover, the attenuation period P is equal to 3 for each device assistant; this means that the interest for a category that has not been visited for three consecutive days is decreased by using the coefficient $\psi \in [0, 1]$. Finally, for each client the parameter k is equal to 4, thus showing to the user all the instances of the four most interesting categories. We have performed a clustering of the profilers each day, by using the CLARANS algorithm on the similarity matrix, as discussed in Section 4.3. MASHA, X-COMPASS, and EC-XAMAS device assistants are built by following the descriptions of the relative data structures and recommendation algorithms proposed in [Rosaci D. and Sarné G.M.L. 2006; Garruzzo S., Modafferi S., Rosaci D. and Ursino D. 2002; De Meo P., Rosaci D., Sarné G.M.L., Ursino D. and Terracina G. 2007], respectively. For each system, we have used the same data structures suggested in the referred paper, that are different from those used for MUADDIB.

MUADDIB site assistants, profilers and recommenders

The 300 Web sites are hosted on four different server machine (75 Web sites for each machine). Each server machine is a Pentium IV, 4GHZ, 8MB RAM, equipped with Apache Web server on Linux platform. We have associated a site manager to each site, implemented in JADE. When a Web server is contacted by a device assistant for a URL related to a given Web site, the site manager of the site is activated and behaves as described in Section 4.4.

Each user is associated with a profiler, thus we have 200 profilers, which are hosted on two profiler server machines (different from the machines hosting the Web servers), each machine (a Pentium IV, 4GHZ, 4MB RAM) hosting 100 profilers. Each profiler is implemented in JADE, and behaves as described in Section 4.2.

All the profilers adopt the same parameter values: (i) $ND = 3$, having only three types of device assistants for each user; (ii) the prices per Mbyte (in euro cents) that we have considered are: $\epsilon_1 = 0.9, \epsilon_2 = 1.4, \epsilon_3 = 1.8$, as those adopted in [Rosaci D. and Sarné G.M.L. 2006].

We have used four recommenders, each of them implemented in JADE and running on a different recommender server machine (a Pentium IV, 4GHZ, 8MB RAM).

We have realized two different experiments, that we describe in the next three subsections. The first experiment aims at performing a comparative analysis of the *quality* of the recommendations generated by the different recommender systems, in terms of effectiveness. The last experiment instead compares the time costs of the approaches.

6.4 Evaluation of the Effectiveness

The experiment described in this subsection aims at evaluating the relevance of the recommendations generated by MUADDIB, compared with that produced by the other recommenders. To this purpose, we have observed the choices of the users in their navigation among the available Web sites. More in particular, we have monitored each user for 30 days, in order to construct his profile based on the accessed Web pages, without the support of recommender systems. Then, we have monitored each user for other 15 days, supported by the recommender systems. In this phase, each user, when he visits a Web page using his client,

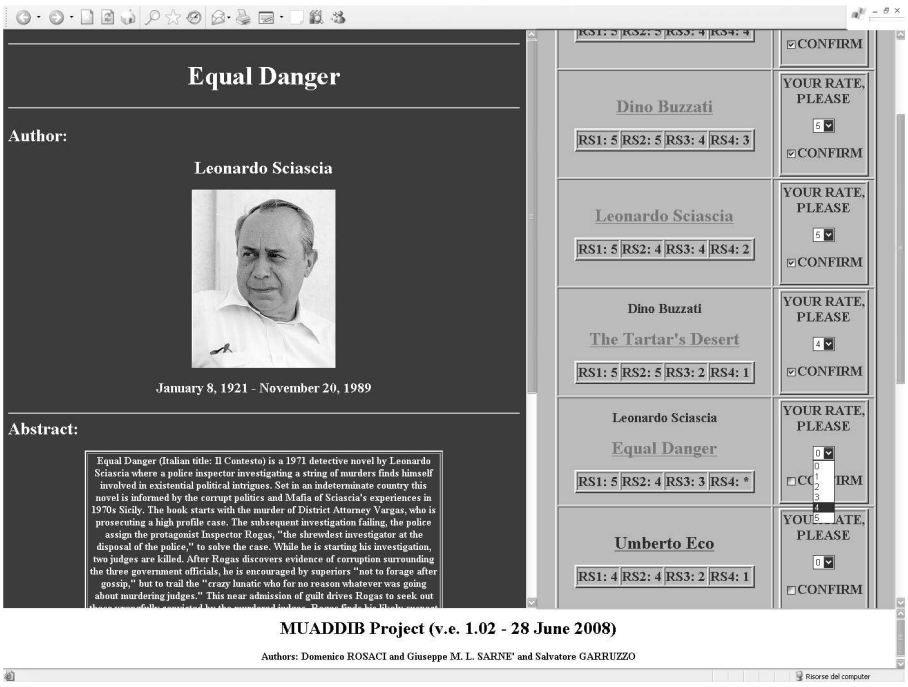

Fig. 8. The software used to evaluate the recommender systems

is provided with an ordered list of recommendations R, composed by the union of the list CB (content-based recommendations) and CF (collaborative filtering recommendations), as described in Section 4.4. The elements of R are ordered based on the interest coefficient. More in particular, to provide the user with a intuitively understandable relevance measure, each element i of R is associated with an integer rate belonging to the set $[1, 2, 3, 4, 5]$. Such a rate of i, denoted by p_i, represents a prediction of the user's interest about the element i, and it is computed based on the interest coefficient ir_i of i, as follows: $p_i = 1$ if $0 < ir_i < 0.2$, $p_i = 2$ if $0.2 \leq ir_i < 0.4$, $p_i = 3$ if $0.4 \leq ir_i < 0.6$, $p_i = 4$ if $0.6 \leq ir_i < 0.8$, $p_i = 5$ if $0.8 \leq ir_i \leq 1$.

In words, we have chosen to use the classical classification of the rates in the interval 1-5, and therefore we have partitioned in five classes the interval $[0, 1]$ to which the interest rate ir_i belongs. The user is required to provide his rating of each recommended element i, denoted by r_i. Figure 8 shows a screenshot of the software used for the evaluation.

6.4.1 *Metrics.* In the literature, three main categories of metrics have been proposed for evaluating the accuracy of a prediction algorithm, namely accuracy metrics, classification accuracy metrics, and rank accuracy metrics. (see [Herlocker J.L., Konstan J.A., Terveen L.G. and Riedl J.T. 2004]). Predictive accuracy metrics measure how close the recommender systems predicted ratings are to the true user ratings. Predictive accuracy metrics are particularly important for evaluating tasks

in which the predicting rating will be displayed to the user such as in the case of our experiment. To measure statistical accuracy we use the mean absolute error (MAE) metric, defined as the average absolute difference between predicted ratings and actual ratings. In our experiments we compute, for each test set, the MAE for each user, and then the average over all the users of the set. Formally, for each set S_j, $j = 1, 2, 3, 4$:

$$MAE_{S_j} = \frac{\sum_{i=1}^{N_{S_j}} |p_i - r_i|}{N_{S_j}} \tag{5}$$

where N_{S_j} is the total numbers of recommendations generated for all the users of S_j. Classification accuracy metrics measure the frequency with which a recommender system makes correct or incorrect decisions about whether an item is good. We use Receiver Operating Characteristic (ROC) sensitivity to measure classification accuracy. The ROC model attempts to measure the extent to which an information filtering system can successfully distinguish between signal (relevance) and noise. The ROC curve represents a plot of recall (percentage of good recommendations returned), versus fallout (percentage of bad recommendations returned). We consider a recommendation good if the user gave it a rating of 4 or above, otherwise we consider the recommendation bad. We refer to this ROC sensitivity with threshold 4 as ROC-4. ROC sensitivity ranges from 0 to 1, where 1 is ideal and 0.5 is random. Since comparing multiple systems using ROC curves is tedious and subjective, we provide as a single summary performance number the area underneath an ROC curve, also known as Swets A measure, that can be used as a single metric of the systems ability to discriminate between good and bad recommendations. Moreover, to complete our analysis, we compute, besides MAE and ROC curve, also the Customer ROC (CROC) curve, another metric introduced in [Schein A.I., Popescul A., Ungar L.H. and Pennock D.M. 2005], and we use as synthetic evaluation parameter the area under the CROC curve. Rank accuracy metrics measure the ability of a recommendation algorithm to yield a recommended ordering of items that matches how the user would have ordered the same items. It is important to point out that ranking metrics do not attempt to measure the ability of an algorithm to accurately predict the rating for a single item, therefore in our case, where we display to the user predicted rating values, it is important to additionally evaluate the system using a predictive accuracy metric as described above. We use the Normalized Distance-based Performance Measure (NDPM) as rank accuracy metric, that is computed as follows:

$$NDPM = \frac{2 \cdot C^{-} + C^{u}}{2 \cdot C^{i}} \tag{6}$$

where C^{-} is the number of contradictory preference relations between the system ranking and the user ranking. A contradictory preference relation happens when the system says that item 1 will be preferred to item 2, and the user ranking says the opposite. C^{u} is the number of compatible preference relations, where the user rates item 1 higher than item 2, but the system ranking has item 1 and item 2 at equal preference levels. C^{i} is the total number of preferred relationships in the users ranking (i.e. pairs of items rated by the user for which one is rated higher

than the other). NPDM is a value ranging in [0.0..1.0], where 0.0 means best recommendations and 1.0 means worst recommendations. In our experiments, we have computed the average of NPDM on all the users.

6.4.2 *Results.* Figure 9-(A) shows how the average MAE of the considered recommender systems varies for different sizes of the set of users. We see that MUADDIB presents a MAE that is always smaller than the other systems, and the advantage increases when the size of the set of users increases too. This is due to the fact that MUADDIB increases its MAE very slightly with the number of users, while the other systems present a more drastic increase. We can see that MUADDIB outperforms MASHA, that is the second best performer, from 15 percent (for a size of 50 users) to 28 percent (for a size of 200 users). The good quality of the MUADDIB recommendations is confirmed by the analysis of the Swet's A measure related to the ROC-4 curve (see Figure 9-(B)), where the advantage of using our recommender with respect to the second-best performer recommender (MASHA) is comprised between 14 percent and 22 percent, while the advantage with respect to the other two recommenders is, in average, about 30 percent. Analogous considerations can be done considering the area under the CROC curve in Figure 9-(C). The combined analysis of MAE and ROC shows that MUADDIB performs significantly better than the other three systems both in predicting the rates of the users and in providing recommendations judged as good by the users. The analysis of the NPDM measure, represented in Figure 9-(D) shows that MUADDIB, X-COMPASS and EC-XAMAS presents performances close enough in providing a recommended ordering of items that matches how the user would have ordered the same items. However, also in this case, MUADDIB is the system that presents the best performance, with an advantage of about 11 percent with respect to MASHA (which is the second best performer). It is also important to point out that X-COMPASS, that does not take into account the effect of the device in the generation of the recommendations (it only considers the device in the construction of the user's profile) shows always worst results than the other three systems.

6.5 Evaluation of the Efficiency

Finally, we have compared the impact of the different recommendation algorithms on the performances of the Web sites. To this purpose, we have considered the user's *access time*, i.e. the system response to the user's request of accessing to a given Web page. Figures 10-(A),10-(B), reports the average access time of the users when accessing a Web site considered in the experiment above, using a desktop PC or a cellular phone, respectively. The average value has been computed on all the Web sites and for different number of users in the system. The experiment shows that MUADDIB introduces an access time significantly smaller than the other systems, and this positive gain in terms of cost increases when the number of users increases too. This result is accordingly with the theoretical considerations discussed in Section 5. In particular, Figure 10-(A) shows that, for a user accessing with a desktop PC, the access time with MUADDIB is almost the same (under than 1 second) for all the sizes of the set of users, while the cost with MASHA, EC-XAMAS and X-COMPASS increases almost quadratically. In this situation, the average access time using MUADDIB with a set of 200 users is 1,10 seconds,

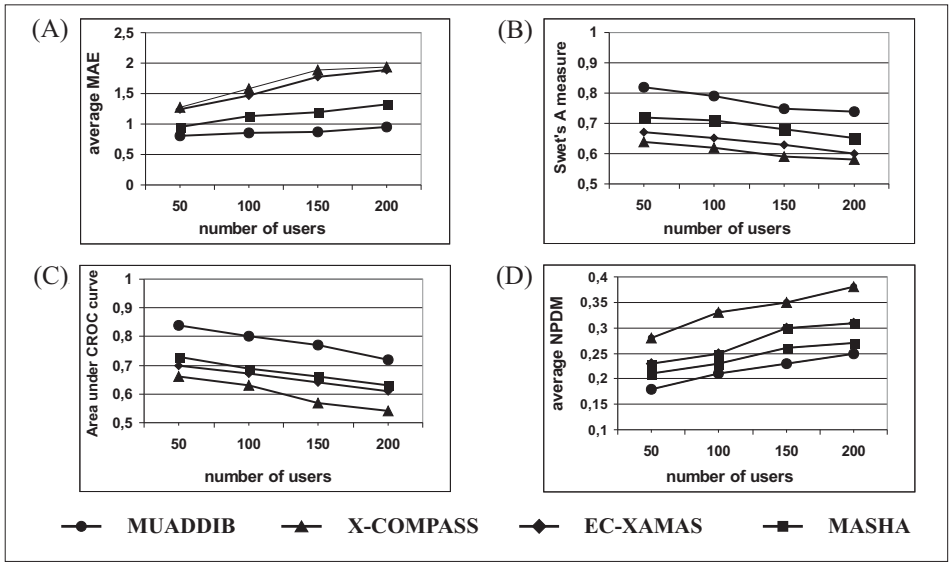

Fig. 9. (A) Average mean absolute error (MAE), (B) Swet's A measure related to the Receiver Operating Characteristic ROC-4, (C) area under the Customer ROC (CROC) curve and (D) average Normalized Distance-based Performance Measure (NPDM) of the recommender systems for different sizes of the set of users

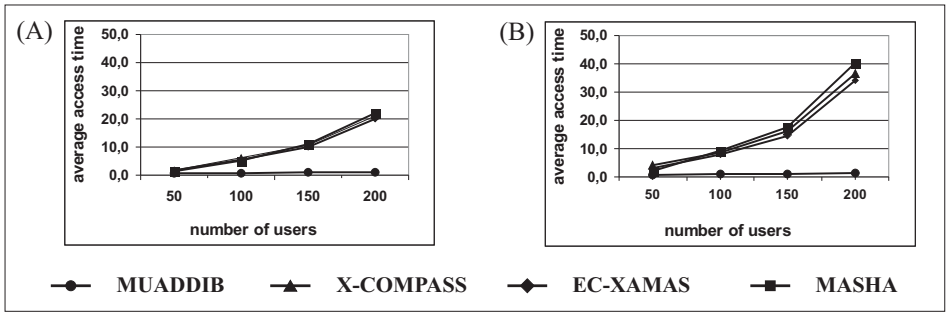

Fig. 10. Average access time of different recommender systems: (A) for a PC desktop; (B) for a cellular phone

while using the other systems the access time is about 20 seconds. The advantage of using MUADDIB becomes more evident if the user exploits a cellular phone (Figure 10-(B)). In this case, the access time of MUADDIB for 200 users is 1,31 seconds, while with the other systems ranges from 35 to 40 seconds. The generally high cost of the network connection for a cellular phones makes practically unapplicable the other systems in presence of a large number of users.

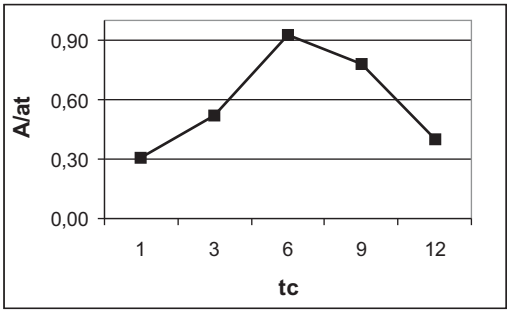

Fig. 11. The ratio A/at (A=average Swet's A measure, at=average access time) for different values of tc (hours)

6.6 Sensitivity to the parameter tc

In the experiments above, we have set the period of re-clustering, denoted as tc, equal to 6 hours. This value is derived from a preliminary analysis of sensitivity we have performed with respect to tc. Accordingly to the discussion reported in section 4.3, when tc increases the quality of recommendations decreases since the recommendations are generated on too old data while, on the contrary, the efficiency of the recommender system improves since the recommender components are less involved in computation and answer more quickly to the request of the site managers. As a consequence, it is necessary to maximize the ratio between the quality of the recommendations and the price for obtaining it. To determine such a threshold, we have considered the average Swet's A measure as a measure of the quality, and the average access time at as a measure of the price, and we have repeated the experiments described in the previous subsections for different values of tc. In Figure 11 we have plotted the value of A/at for different values of tc, so empirically determining the best ratio for tc equal to 6 hours, a value that we have next adopted in the experiments described in the previous subsections to compute all the measures of efficiency and effectiveness. It is worth to point out that the result of this analysis is related to the particular case of using four recommenders. If we change the number of recommenders, it is necessary to redo the analysis of sensitivity.

7. CONCLUSIONS

In this paper we have presented a recommender system architecture, called MUAD-DIB, designed to maintain multi-dimensional information on a distributed environment, in order to generate recommendations on the basis of both user's profile and exploited device. More specifically, our system is based on the following two ideas. The first is that a device assistant monitors a user that is exploiting a fixed device to build a light profile just for that device, while a profiler constructs off-line a complete user's profile. This leads to make more simple the task of the device assistant, that often has limited resources and, on the other hand, to take into account the different exploited devices in constructing the user's profile. The second

is that each group of similar users is associated with a recommender. Such a software component computes off line the similarity between the users and stores the users'behaviour in order to support both content-based and collaborative filtering recommendations. This leads to generate very effective recommendations, taking into account also the exploited devices, and leaving to the site manager the only task of generating a graphical presentation. We have performed some experiments to evaluate the performances of our system, in comparison with other four distributed recommender systems, and the obtained results show a significant improvements of the suggestions and a significant reduction of the time costs. It is important to point out that the improvements introduced by MUADDIB in the efficiency of the recommendation is a theoretical result, while the better quality of the generated recommendations is derived by the experimental results. These results seem promising, but they need to be confirmed by further analytical studies. This is a subject of our ongoing research.

REFERENCES

Aberer K., Alima L.O., Ghodsi A., Girdzijauskas S., Haridi S. and Hauswirth M. 2005. The Essence of P2P: A Reference Architecture for Overlay Networks. In *Proc. of the 5th IEEE Int. Conf. on Peer-to-Peer Computing (P2P '05)*. Vol. 0. IEEE Comp. Soc., Washington, DC, USA, 11–20.

Aberer K. and Hauswirth M. 2004. *P2P Systems: Practical Handbook of Internet Computing*. CRC Press, Boca Raton, FL, USA.

Adomavicius G. and Tuzhilin A. 2001. Multidimensional Recommender Systems: a Data Warehousing Approach. In *Proc. of the 2nd Int. Work. on Electronic Com. (WELCOM01). LNCS, vol. 2232*. Springer-Verlag, London, UK, 80–192.

Adomavicius G., Sankaranarayanan R., Sen S. and Tuzhilin A. 2005. Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach. *Trans. Inf. Syst. 23,* 1, 103–145.

Al Halabi W.S., Kubat M. and Tapia M. 2007. Time Spent on a Web Page is Sufficient to Infer a User's Interest. In *Proc. of the IASTED Europ. Conf. IASTED European Conference (IMSA'07)*. ACTA Press, Anaheim, CA, USA, 41–46.

Badi R., Bae S., Moore J.M. Meintanis K., Zacchi A., Hsieh H., Shipman F. and Marshall C.C. 2006. Recognizing User Interest and Document Value from Reading and Organizing Activities in Document Triage. In *Proc. of the 11th Int. Conf. on Intel. User Interfaces (IUI '06)*. ACM, New York, NY, USA, 218–225.

Baraglia R. and Silvestri F. 2007. Dynamic Personalization of Web Sites Without User Intervention. *Commun. ACM 50,* 2, 63–67.

Baus J. and Krüger A. and Wahlster W. 2002. A Resource-Adaptive Mobile Navigation Nystem. In *Proc. of the 7th Int. Conf. on Intelligent User Interfaces (IUI '02)*. ACM, New York, NY, USA, 15–22.

Berkhin P. 2002. *Survey of Clustering Data Mining Techniques*. Accrue Soft., San Jose, CA, USA.

Billsus D. and Pazzani M.J. 2000. User Modeling for Adaptive News Access. *User Modeling and User-Adapted Interaction 10,* 2-3, 147–180.

Birukov A., Blanzieri E. and Giorgini P. 2005. Implicit: A Recommender System that Uses Implicit Knowledge to Produce Suggestions. In *Work. on Multi-Agent Information Retrieval and Recommender Systems at the 19th Int. Joint Conf. on AI (IJCAI-05)*. Edinburgh, UK.

Breese J., Heckerman D. and Kadie C. 1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proc. of the 14th Int. Conf. on Uncertainty in Artificial Intelligence (UAI'98)*. Morgan Kaufmann, San Fransisco, CA, USA, 43–52.

Brin S. and Page L. 1998. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks 30,* 1, 107–117.

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

Chai W. and Vercoe B. 2000. Using User Models in Music Information Retrieval Systems. In *Proc. of the 1st Int. Symp. on Music Information Retrieval (ISMIR 2000).*

Chi E. H. and Mytkowicz T. 2007. Understanding Navigability of Social Tagging Systems. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI'07).* ACM, New York, NY, USA.

Claypool M., Le P., Wased M. and Brown D. 2001. Implicit Interest Indicators. In *Proc. of the 6th Int. Conf. on Intel. User Interfaces (IUI '01).* ACM, New York, NY, USA, 33–40.

De Meo P., Rosaci D., Sarné G.M.L., Ursino D. and Terracina G. 2007. EC-XAMAS: Supporting E-Commerce Activities by an XML-Based Adaptive Multi-Agent System. *Appl. Artif. Intell. 21,* 6, 529–562.

Díez J., del Coz J.J. Luaces O. and Bahamonde A. 2008. Clustering People According to Their Preference Criteria. *Expert Syst. Appl. 34,* 2, 1274–1284.

Eirinaki M. and Vazirgiannis M. 2005. Usage-Based PageRank for Web Personalization. In *Proc. of the 5th IEEE Int. Conf. on Data Mining (ICDM '05).* IEEE Comp. Soc., Washington, DC, USA, 130–137.

Eirinaki M. and Vazirgiannis M. 2007. Web Site Personalization Based on Link Analysis and Navigational Patterns. *ACM Trans. Interet Technol. 7,* 4, 21.

Garruzzo S., Modafferi S., Rosaci D. and Ursino D. 2002. X-Compass: An XML Agent for Supporting User Navigation on the Web. In *Proc. of the 5th Int. Conf. on Flexible Query Answering Systems (FQAS '02).* Springer-Verlag, London, UK, 197–211.

Goecks J. and Shavlik J. 2000. Learning Users' Interests by Unobtrusively Observing Their Normal Behavior. In *Proc. of the 5th Int. Conf. on Intel. User Interfaces (IUI '0O).* ACM, New York, NY, USA, 129–132.

Goldberg K., Roeder T., Gupta G. and Perkins C. 2001. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Information Retrieval 4,* 2, 133–151.

Grefenstette G. 1994. *Explorations in Authomatic Thesaurus Construction.* Kluwer Academic Pub., Hingham, MA, USA.

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, 5–53.

Jacobsson M., Rost M. and Holmquist L.H. 2006. When Media Gets Wise: Collaborative Filtering with Mobile Media Agents. In *Proc. of the 11th Int. Conf. on Intel. User Interfaces (IUI '06).* ACM, New York, NY, USA, 291–293.

JADE URL. 2007. http://jade.tilab.com/.

Joerding T. 1999. Temporary User Modeling for Adaptive Product Presentations in the Web. In *Proc. of the 7th Int. Conf. on User Modeling (UM '99).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 333–334.

Kelly D. and Belkin N.J. 2004. Display Time as Implicit Feedback: Understanding Task Effects. In *Proc. of the 27th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR '04).* ACM, New York, NY, USA, 377–384.

Kim H.-R. and Chan P.K. 2008. Learning Implicit User Interest Hierarchy for Context in Personalization. *Applied Intell. 28,* 2, 153–166.

Kim K.-J. and Ahn H. 2008. A Recommender System Using GA K-means Clustering in an Online Shopping Market. *Expert Syst. Appl. 34,* 2, 1200–1209.

Lee W.-P. 2004. Deploying Personalized Mobile Services in an Agent-based Environment. *Expert Systems with Appl. 32,* 4, 1194–1207.

Li Q., Kim B.M. and Myaeng S.H. 2005. Clustering for Probabilistic Model Estimation for CF. In *Proc. of the 14th Int. Conf. on World Wide Web: Special Interest Tracks and Posters (WWW '05).* ACM, New York, NY, USA, 1104–1105.

Linden G., Smith B. and York J. 2003. Amazon.com Recommendations. Item-to-Item Collaborative Filtering. *IEEE Internet Computing 7,* 1, 76–80.

Melville P., Mooney R.J. and Nagarajan R. 2002. Content-boosted Collaborative Filtering for Improved Recommendations. In *Proc of the 18th National Conf. on AI.* American Ass. for AI, Menlo Park, CA, USA, 187–192.

Middleton S.E., Shadbolt N.R. and De Roure D.C. 2003. Capturing Interest Through Inference and Visualization: Ontological User Profiling in Recommender Systems. In *Proce. of the 2nd Int. Conf. on Knowledge capture (K-CAP '03).* ACM, New York, NY, USA, 62–69.

Miller B.N., Konstan J.A. and Riedl J. 2004. PocketLens: Toward a Personal Recommender System. *ACM Trans. Inf. Syst. 22,* 3, 437–476.

Mourland C. and Germanakos P. (Eds). 2007. *Intelligent User Interfaces: Adaptation and Personalization Systems and Technologies.* Information Science Reference.

MUADDIB site URL. 2008. http://www.muad.altervista.org.

Ng R.T. and Han J. 2002. CLARANS: A Method for Clustering Objects for Spatial Data Mining. *IEEE Trans. on Knowl. and Data Eng. 14,* 5, 1003–1016.

Rosaci D. and Sarné G.M.L. 2006. MASHA: A Multi-Agent System Handling User and Device Adaptivity of Web Sites. *User Modeling and User-Adapted Interaction 16,* 5, 435–462.

Sarwar B., Karypis G., Konstan J. and Riedl J. 2000. Analysis of Recommendation Algorithms for e-Commerce. In *Proc. of the 2nd ACM Conf. on Electronic Comm. (EC '00).* ACM, vNew York, NY, USA, 158–167.

Schafer J.B., Konstan J.A. and Riedl J. 2001. E-Commerce Recommendation Applications. *Data Min. Knowl. Discov. 5,* 1-2, 115–153.

Schein A.I., Popescul A., Ungar L.H. and Pennock D.M. 2005. CROC: A New Evaluation Criterion for Recommender Systems. *Electronic Commerce Research 5,* 1, 51–74.

Symeonidis P. , Nanopoulos A., Papadopoulos A.N. and Manolopoulos Y. 2008. Nearest-Biclusters Collaborative Filtering based on Constant and Coherent Values. *Inf. Retr. 11,* 1, 51–75.

Taghipour N., Kardan A. and Ghidary S.S. 2007. Usage-based Web Recommendations: a Reinforcement Learning Approach. In *Proc of the 2007 ACM Conf. on Recommender systems (RecSys '07).* ACM, New York, NY, USA, 113–120.

Tanenbaum A.S. and Van Steen M. 2001. *Distributed Systems: Principles and Paradigms.* Prentice Hall PTR, Upper Saddle River, NJ, USA.