Quickly Learning User Characteristics for Efficient Interruptability in Agent–User Collaborative Systems

Tammar Shrot[1], Avi Rosenfeld[2], and Sarit Kraus[1]

[1] Bar-Ilan University, Ramat-Gan 52900, Israel
{machnet, sarit}@cs.biu.ac.il
[2] Jerusalem College of Technology, Jerusalem 91160, Israel
rosenfa@jct.ac.il

Abstract. In teamwork when a user and an agent are working together on a joint task it may be important to share information in order to determine the appropriate course of action. However, communication between agents and users can constitute costly user interruptions. One of the most important issue concerning the initiation of information sharing in teamwork is the ability to accurately estimate the cost and benefit arising from those interruptions. While cost estimation of interruptions has been investigated in prior works, all of those works assumed either a large amount of information existed about each user, or only a small number of states needed consideration. This paper presents a novel synthesis between Collaborative Filtering methods together with classification algorithms tools in order to create a fast learning algorithm. This algorithm exploits the similarities between users in order to learn from known users to new but similar users and therefore demands less information on each user. Experimental results indicate the algorithm significantly improves system performance even with a small amount of data on each user.

Key words: Cost Estimation, Collaborative Filtering, Classification Algorithm

1 Introduction

An important aspect of teamwork is efficient inner-team communication [1]. When working together on a joint task it is important to share information in order to coordinate consequent actions. In addition, different agents on a team often possess information required by others. The need for efficient communication arises in mixed human-computer teams as well as in homogeneous computer-agent environments [10]. However, it is important to appropriately time the communication, since poorly timed interruptions can lead to averse effects on task performance and on a human agent's emotional state [1].

Cost estimation of interruptions has been investigated in prior works (see Section 2.1). However, these methods require many repetitive interactions or a very strict domain with a small number of states. In the domain we studied,

as well as in many other domains, we assume that these repetitive iterations incur a very high cost. In addition, some applications, such as on–line bidding agents have a limited number of iterations with each user making this approach impractical.

We focus on developing novel algorithms that can quickly and accurately model user preferences. Our research was specifically motivated by DARPA's Coordinators program[3]. The basic goal of this program is to create automated decision support units that better focus a user's attention in dynamic environments. We consider a teamwork framework where agents and human operators work together on a joint task and have unique capabilities. Human operators are assumed to have access to more complete domain knowledge or expert knowledge external to their agents. However, people's time is valuable, and thus the team incurs a cost every time the agent interrupts the person to obtain the information she may have. The profile of the specific user receiving the query can quickly change as the user's availability to provide information is subject to dynamics based on the person's ability to be interrupted, her current activity or the current state of the environment. The agent must decide if it should initiate a query to a human operator based on evaluating whether the cost of interrupting the user is outweighed by the value gained from her knowledge.

The basis of our solution is the assumption that users can be clustered such that the behavior of one user will be similar to the other users in her cluster. Thus, once we have knowledge about some users we can generally estimate the value and cost associated with an agent–human interaction of new, but similar users as well. Specifically, we propose the use of a new user modeling approach with elements of Collaborative Filtering (CF) algorithms. Traditional CF algorithms are typically applied to recommend a given product (book, movie, game, etc.) to a user based on information gleaned about general users' buying behavior. Collaborate Filtering analyzes the relationships between users and interdependencies among products, in order to identify new user–item associations. In addition, to avoid the need for extensive data collection about items or users, Collaborative Filtering requires no domain knowledge [3].

We propose a new approach of combining components of Collaborate Filtering algorithms with basic classification algorithms such as the *C4.5 Decision Tree* algorithm [15] or the *k-nearest neighbor* ($k - NN$) algorithm [5]. The advantage of this synthesis over traditional learning methods is its significant reduction in the learning time needed to model a given user. This allows us to quickly decide about the efficiency of an interruption with only limited data and can avoid pitfalls such as protracted learning periods and elicitation of private user data. Our approach is also significantly distinguished from traditional Collaborate Filtering algorithms and Machine learning algorithms. Three major differences exist between our approach and Collaborate Filtering algorithms. First, in Collaborative Filtering algorithms the similarity between users is decided by shared habits, and the similarity between items is decided by shared history. In our domain such shared information does not exist and other, machine

[3] http://www.darpa.mil/ipto/programs/coor/coor.asp

learning, tools are used. Second, several traditional Collaborate Filtering algorithms work by identifying the precise value of a current parameter state (e.g. the genre of a movie) and use this information to decide whether to recommend an item [7]. In our approach, we allow a range of possible parameter values. The result is an algorithm that can effectively make decisions without precise state information. Third, our approach has no need for personal information (demographic information, economic and marital status, age, etc.) typically used in Collaborate Filtering algorithms [7]. Many users are reluctant to divulge private information, leading to many people refraining from these systems. In contrast, our model focuses exclusively on the user's interaction with the agent, and thus does not require sharing any personal information. In addition, our approach significantly distinguished from machine learning algorithms. In contract two machine learning algorithms, our approach uses two different phases to decide about the situations' similarity level and each phase uses different type of data regarding the situation.

2 Related Work

We propose a new method for cost estimation with is motivated by Collaborative Filtering methods.

2.1 Cost Estimation of Interruptions

One of the most important issues concerning the initiation of teamwork interruptions is the ability to accurately estimate the cost and benefit arising from the interruption. Accurate estimation will enable interruption only when it will have a positive ipmact on the team's performance.

Previous works have investigated how to estimate the cost of interruptions [8, 18]. Fleming and Cohen [6] were the first to build a user-specific model which generally takes the user's specific factors into account. They used cost estimation to create a decision making mechanism in order to decide when to initiate communication. Horvitz and Apacible [8] studied the user's benefit from receiving information from the computer agent, while our study focus on the opposite situation — the agent wishes to receive information from the user. Tambe et al. [18] focused on when to turn control to the user, instead of information transfer. Sarne and Grosz [16] offered an efficient model to manage the agent's information and to decide when to interrupt the user. Kamar et al. [10] used POMDP and MDP models to evaluate the cost of interruption while taking into account the possible mismatch between the computer's calculation of utility and the person's perception of it. The key difference in our research is we study cost-estimations that can work in dynamic domains in which the environment's conditions rapidly change, actions occur quickly, user's abilities change over time and decisions must be made within tightly constrained time frames. For example, Nonetheless, work by Sarne and Grosz [16] needs a large amount of information on each user before it can begin to operate at an efficient level. Work by Kamar et al. [10] considers a

limited domain with a small number of states and is computationally–infeasible for domains with many dynamic states.

2.2 Collaborative Filtering

Collaborative Filtering (CF) is a method of making automatic predictions (filtering) about the preferences of a user by collecting data on the preferences of many users (collaborating). There are hundreds of examples of recommendation systems via Collaborative Filtering. Surveys of recommendation systems and collaborative systems are presented in [20, 4, 7]. User profiling and matching mechanisms are illustrated, especially on Collaborative Filtering techniques. In addition, a number of Collaborative Filtering algorithms are compared for accuracy of available test data.

Collaborate Filtering models can be built based on users or items. User Based collaborative filtering systems find other users that have displayed similar tastes to the active user and recommend the items similar users have preferred [2, 12, 14]. Item Based models recommend items that are most similar to the set of items the active user has rated [9, 17, 13].

Karypis [11] was the first to recommend an approach that combines the best of the Item Based and the User Based (classic Collaborate Filtering) algorithms, by first identifying a reasonably large neighborhood of similar users and then using this subset to derive the Item Based recommendation model. Vozalis et al. [19] have developed a hybrid method consisting of a number of steps. In the first step, the algorithm creates a neighborhood of users most similar to the selected active user; it first calculates the similarity level between the users and the active user and then chooses the k-nearest neighbor who rated a large number of items. In the second step, the rating of the k users is used to calculate item-similarity between these users. In the final step, the algorithm recommends the active user items similar to the items chosen (based on the first two steps).

In our research we leverage these approaches to quickly learn about new users from known users. We present a novel variation of traditional learning algorithms that applies the tools and principles defined in the hybrid User and Item Based Collaborative Filtering method in order to incorporate them into our learning algorithm. In the next section, we present specifics of the Coordinators Domain we studied, as well as specifics of our algorithms.

3 A Synthesis between CF and Machine Learning

We generally model the Coordinator's joint agent-human teamwork domain problem as follows: Assume a user has to complete a task within a limited time. The task consists of many sub-tasks. Each successful completion of a sub-task entitles the team to a certain gain, referred to as "*taskGain*", different types of task have different gain value. In addition, successfully transferring desired user information to the agent entitles the team to a certain different gain, referred to as "*comGain*", this value changes according to the information accuracy.

1. Accept a new non labeled situation $s = (i, u)$.
2. Use u to create a user profile p for s.
3. Use the profile p to build a neighborhood NGB of users that were found to be similar to u.
4. $Items = \emptyset$
5. $\forall s' = (i', u')$ *s.t.* $u' \in NGB \quad Items = Items \bigcup \{i'\}$
6. Build a *classification* model CM between i and $Items$ using a classification algorithm.
7. Label the new situation s according to the classification decision of CM.

Fig. 1. Algorithm for deciding whether the timing is good or bad.

Our proposed algorithm (Figure 1) is motivated by the Collaborative Filtering hybrid approach. Just like in the Collaborative Filtering algorithms our method has two phases: a "user" phase, and a "item" phase. The first phase uses user specific data in order to identify similar users in the database and construct an environment of similar users. The second phase of the algorithm uses state specific data in order to decide about the state. However, the second phase only takes its information from states that belong to users in the "similar environment" built in the first phase (defined herein).

Each data point in our database (situation) is constructed from two distinct data types: user latest history and interruption profile. User latest history is a collection of vectors with information gathered within a short period of time (20 seconds in our experiments) before the agent considers if it should interrupt the user. This historical information shows how the user behaves and enables the algorithm to construct a profile of the user's behavior. The second data type - the interruption profile is a vector that describes the user's state immediately prior to the time of the proposed interruption. Each of these vectors (from the user's profile and interruption profile) contains numerical and/or other discrete values for different attributes. The user's history an interruption profiles vectors' attributes, include for example information regarding the user's location, percentage of the task accomplished, time from the beginning of the task, location of the nearest disturbing elements, etc. All values in all vectors' attributes are normalized to the same scale.

While the algorithm's structure is motivated by Collaborative Filtering methods, the tools actually used to decide the user's neighborhood and if an interruption should take place are machine learning tools. Collaborative Filtering methods cannot feasibly be implemented on the given data. In teamwork domains, unlike in Collaborative Filtering, users cannot be compared according to shared habits and items (states) cannot be compared according to shared history. Consequently, different methods must be found to quickly model new users's and items's similarities. Since our data is labeled, our solution is to use traditional machine learning classification algorithms. The classification algorithms are used to quickly compare the users (in the first phase) and items (in the second phase) without resorting to a shared habits or shared history.

In the first phase ("user" phase) the algorithm builds a "user's" similarity model between the new given situation and the given labeled situations in the database (Lines 2 and 3 in the algorithm). This phase uses only the first data type in the situation (user's latest history). The users' similarity model is built according to the similarity between the "user latest history" data type in the situations. Specifically, the user's latest history is a set of x vectors that represents the user's behavior in the short period sampled before the interruption (20 seconds in our experiments). For each user, the algorithm calculates the changes in the user's attributes value throughout the latest history measuring time. Then, for each attribute it is calculating the average change in the user's attributes values (Line 2 in the algorithm). The user's profile is the vector of averages changes. Namely, the algorithm uses the user's latest history in order to calculate the average change in the user behavior (values) in this time sequence. Then, the similarity between two users is measured as the distance between the two calculated profiles (Line 3 in the algorithm). The assumption behind this approach is that users with similar profiles (same average change in values) undergo the same process and therefore most probably act in similar ways. This model represents the user's similarity level between the new situation and the given labeled situations. Once the similarity model is complete, the algorithm takes the l situations of the most similar users in the database as the new situation's neighborhood, and uses this neighborhood in the algorithm's next stage. Notice, that this phase is using tools taken from the *k-nearest neighbor* $(k-NN)$ algorithm [5] in order to locate the new situation's neighborhood.

The next stage of the algorithm (Lines 4 to 7 in the algorithm) uses only the second date type in the situation (interruption profile). Once the user's profile and neighborhood are constructed, the next stage of the algorithm is to build an "item's" (interruption profiles) similarity model between the situations that belong in the neighborhood. Once the above algorithm identifies the user's neighborhood in the first phase it uses a machine learning classification algorithm on the items that belongs to the neighborhood's users and returns the calculated classification. The net result is that once a new situation arrives, the need only a very short time to gather enough data in order to decide how to treat it. This allows for a faster and more accurate classification than the base machine learning algorithms alone could provide. While our approach is meant to work with any machine learning algorithms, we specifically considered two classification algorithms: the *C4.5 Decision Tree* algorithm [15] and the *k-nearest neighbor* $(k-NN)$ algorithm [5].

4 Experimental Results

In order to study our new algorithm and its ability we conducted a number of experiments. This section contains the experiments we did and the rational behind them, as well as the description of the environment used for our experiments.

4.1 Experimental Environment

We are currently conducting experiments with a game setting called "Final Frontier Trader"[4]. The goal of the game is to destroy all enemy ships and asteroids. For every asteroid or ship destroyed the team earns a certain gain. We refer to the constant score obtained from destroying an asteroid as *astGain*, and the constant gain from destroying a ship as *shipGain*. The game has a time limit (ten minutes) and the human operator (user) must destroy all enemy ships and asteroids before the end of the game. A user who dies during the game (e.g. shot by enemy ships) or the time limit passed loses a large number of points. On the other hand, the faster the user accomplishes her main goal the larger the gain she earns from successful task completion. Additionally, the user can increase the team's gain by answering questions from the agent (*comGain*). We assume that while the agent asks a question, the user cannot simultaneously perform her previous subtask. Thus, while the user is prompted for information the game continues, the user's ship continues moving, and enemy ships can potentially shoot at and even destroy the user's ship. Control is returned to the user only after she correctly answers the question. Moreover, for each wrong answer the user provides, the value of *comGain* rewarded to the team is reduced. The agent's questions are simple mathematical questions which are presented as multiple choice (*SAT*) questions. We assume that these questions may hurt the human's performance. Our goal is for the agent to ask a given question only when the cost to the user for answering the question will be lower than the gain from answering the question.

The agent collects information regarding the user's state in regular intervals (life status, location, percentage of the task accomplished, time from the beginning of the task, location of nearest enemies, etc.). This information is necessary to construct the user's state model used by our algorithms.

In our research protocol each subject played three games:

1. A simple scenario with simple questions – users' game learning session.
2. A complex scenario without questions – experiments without question.
3. The complex scenario with questions – the experiment session.

All subjects played scenario 1 first, as a training session about the game's controls. The order in which they played scenarios 2 and 3 varied. 50% of the users played the experimental scenario without questions before playing it with questions, and 50% played it with questions before playing it without questions. This was done to negate any impact on results from the order these scenarios were played in.

The information gathered from scenario 1 was omitted from the analysis of the results, since these games were only used to teach the users about the environment. The information gathered from scenario 2 was compared with the information from scenario 3 in order to isolate the effect the questions had on

[4] http://fftrader.sourceforge.net

the user performances and make sure that they interrupted the users. This information was used while labeling the questions in the database. Scenario 3 was the experimental scenario used to evaluate our algorithm.

Each subject performed the entire experimental protocol as described above. The subject's state and status were sampled every 5 seconds (user's latest history) and also before (interruption profile) and after each interruption in the scenarios with questions. At the end of the research protocol each question (interruption) was labeled as either "good" or "bad" according to the effects it had on the score and according to the amount of interruption it caused the user (the amount of life lost, if the user was set out off course, etc.). The labels were determined based on an analysis of the difference between the "before" and "after" information as well as the final score the user achieved in the game.

We ran a 10–fold cross–validation test on the data collected from the third scenario. In each of these 10 iterations a subset of the data (90%) was randomly chosen, and used in order to build the model. The rest of the data was used to test the model. The situations from the test data were inserted into the model as new situations that must be analyzed. The model decisions were compared to the situations' labels. Four decision models were examined:

- A naive $k-NN$ classification model.
- A naive $C4.5$ decision tree classification model.
- A "User Based" + $k-NN$ model ($UB+KNN$).
- A "User Based" + $C4.5$ decision tree model ($UB+C4.5$).

4.2 Testing the Algorithms

We posit that our algorithm can learn from similar subjects to new subjects and correctly decide whether it is currently a good or a bad time to interrupt the subject. To test this hypothesis, we ran the above research protocol on a group of 56 people who varied in age, occupation and level of computer knowledge. Each person had between 4 to 25 interruptions in his game (average value of 20 interruptions per game). Therefore, we had approximately 1120 situations (user's latest history + interruption's profile) in our database. For each situation we had a short history from the user's activity in the previous 20 seconds (about 4 - 5 vectors) which was used to identify users' similarities, and an interruption's profiles used to identify items' (states') similarity. The scenarios used have a large number of enemies and their aggressive level varied from low to high. Therefore, the number of possible states the users could encounter was enormous. We deliberately studied a large variety of different users and states so we can examine the true influence of the "User Based" approach.

The results, as presented in Figure 2, support our claims. They show that in a heterogeneous environments users using the "User Based" approach significantly improves the system performances. We can see that there is a significant change ($p < 0.01$) between the naive algorithms and the enhanced "User Based" algorithms. In addition, it is seem that there is no significant difference ($p > 0.05$) between the naive $C4.5$ decision tree algorithm and the naive k-nearest neighbor

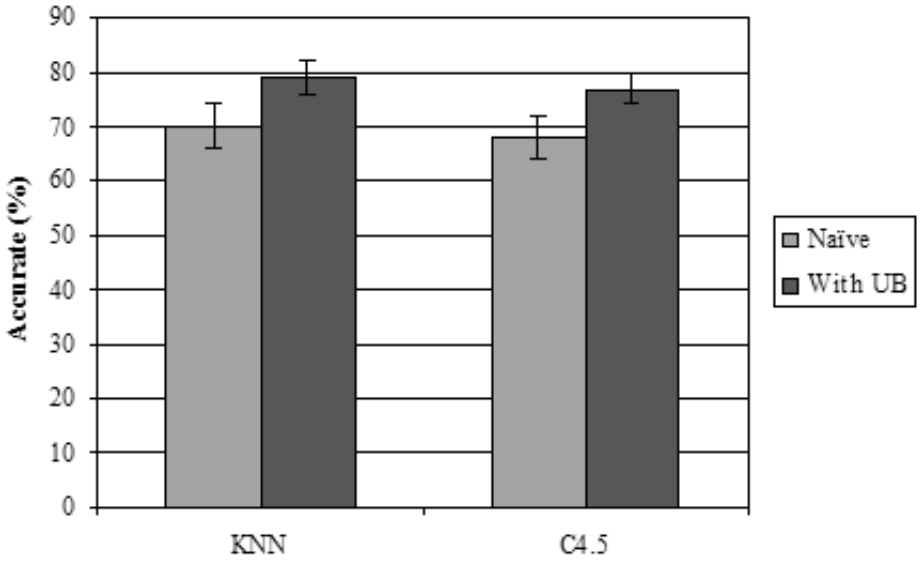

Fig. 2. The algorithm accuracy percentage as a function of the different algorithms.

$(k-NN)$ classification algorithm. It is interesting that the both naive algorithms ($C4.5$ and $k-NN$ algorithms) were improved by a very similar factor with no significant difference ($p > 0.05$) between "User Based" enhance $k-NN$ and the "User Based" enhance $C4.5$ algorithms. It shows that both algorithms can significantly benefit from using the "User Based" approach in heterogeneous environments.

4.3 The Effect of Adding Users to the System

All the algorithms discussed in this paper assume that there is an initial labeled database that can be used in the classification algorithms. However, questions arise as to how much initial training is needed before the system can begin to be used. Therefore, we studied how the size of the initial database effects each of the algorithms, and attempted to reveal the minimal amount of training data needed for each algorithm. During this experiment it was important to make sure that while enlarging the number of users, we are not changing the level of heterogeneity among the users. This was done by using the procedure described in section 4.4.

In order to simulate a larger database we enlarge the number of subjects used in the experiments. On average, each subject added 20 new situations to the database. Namely, when we have 30 subjects in the experiment we have 600 situations in the database.

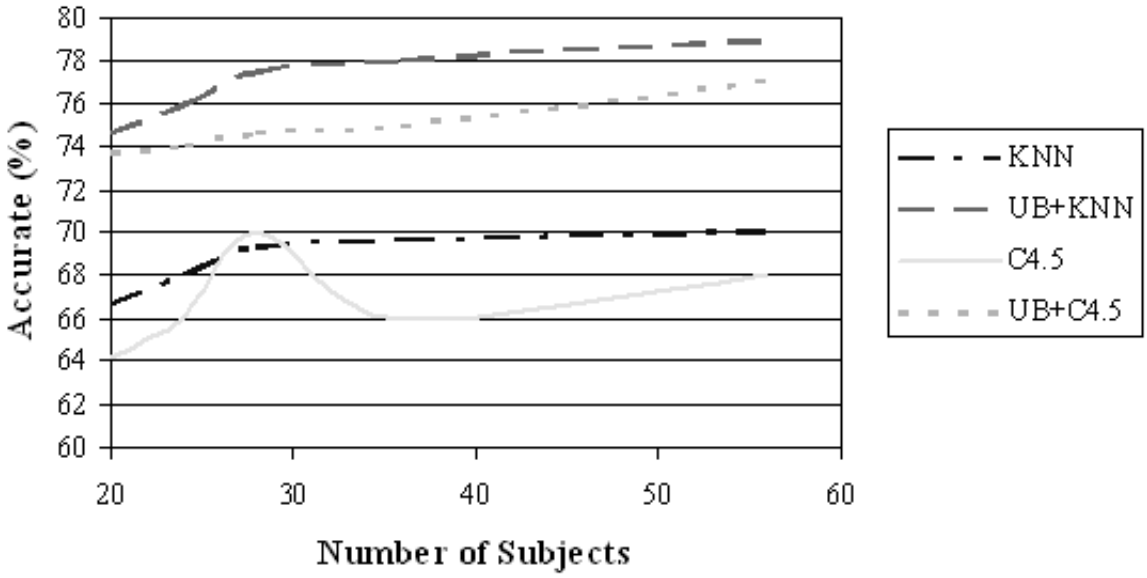

Fig. 3. The algorithms accurate percentage as a function of the number of subjects.

Several interesting phenomenon appear in Figure 3. First, as stated in Section 4.2, using "User Based" aspects from Collaborative Filtering improves both naive machine learning algorithms by the same factor. This significant improvement is found even in a small database that contain 20 subjects ($\sim$ 400 situations). Second, it seems that while the number of users was smaller than 30 (around 600 situations in the database) the algorithms were not stable. However, when the number of users exceeds 30 (more than 600 situations) all algorithms became stable, including the our algorithms that use less data in order to build their model ($l = |database| * 0.1 \approx 60$ situations). Apparently, the similarity between users is strong enough to achieve a significant improvement over the naive algorithm even with a small amount of data.

4.4 The Effect of Changing the Heterogeneity Level of the Subjects

One of our hypothesis is that as the users' level of heterogeneity increases the larger the gain we have from using the "User Based" approach. This is because the Collaborative Filtering "User Based" approach learns based on user differences.

To understand differences between users, we divided them into the following four groups: the "gamer" group; the "skilled" group; the "fair" group; and the

"hopeless" group. We asked user's to self-describe which group they believed they would fall in. Subjects that their self definition did not match their performances were removed from all our experiments, leaving us with 56 subjects.

The number of situations were kept constant during this experiment, and the "subjects' level of heterogeneous" was modified by controlling the ratio of subjects that came from each group.

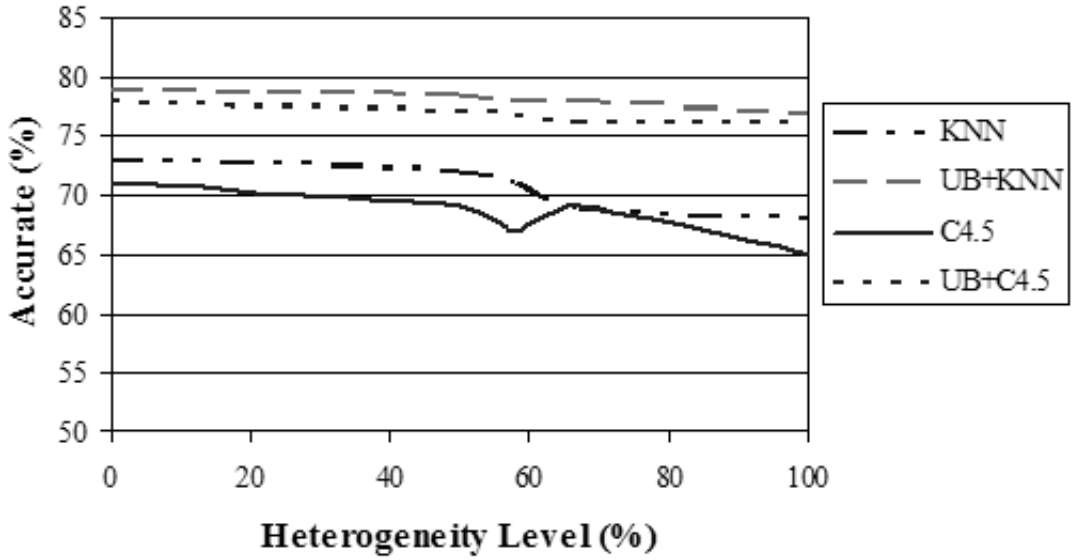

Fig. 4. The algorithms accurate percentage as a function of the heterogeneity's level.

The results in Figure 4 show that the "User Based" approach performs better in heterogeneous environments. We define that heterogeneous level means the ratio between the groups in the general population. Namely, heterogeneity of 0% means that all situations in the database are from the same group; heterogeneity of 100% means that each group has an equal number of situations in the database. Notice, that while the "User Based" algorithms are negligibly effected by the reduction in the user's heterogeneity level, the naive algorithms perform much better. Therefore, as the heterogeneity level decreases the "User Based" approach shows less improvement compared to the naive algorithms.

This result is not surprising. As the heterogeneity level in the database increases, the learning task becomes harder as more problem instances are increasingly less similar. Therefore, it is obvious that naive classification algorithm will produce less suitable results.

5 Conclusions and Future Work

This paper introduces a novel approach to combine Collaborative Filtering methods with classification algorithms tools in order to create a new fast user characteristic algorithm for mix agent–user system. This approach significantly improves system performances even in the absence of sufficient information for learning or user modeling. We found that after a small initial database was created with as little as sparse data from 30 subjects, our algorithms were able to accurately model the subject's interruption preferences with nearly 75% success in less than 20 seconds!

We conclude that the algorithms we present can provide a good solution for semi–tailored applications that have a large number of users but only a short dialog with each user. These applications cannot realistically gather enough information on a single user in order to grant her the personalized service she requires. However, an application can use the offered approach in order to learn from past users and quickly profile new users.

There are many directions we would like to pursue in our future work. First and foremost, we would like to preform "on-line" experiments. The experiments preformed in this paper used an "off-line" experimental procedural. Namely, the subjects were given questions in fixed time intervals, and the analysis on the data using our new algorithm was done off-line. We would like to conduct experiments where the decision whether to ask a question in a given time will be decided on-line using our proposed algorithm. Another direction for future research is to investigate other domains and problems. For example, we hope to study situations where providing information to agents does not directly increase the team's value, but rather provides the team with additional abilities that would enable it to perform new tasks, or improve their performance in existing tasks. Similarly, we currently studied user-agent interruption manager only as modeled through answering mathematical questions. We would like to examine different types of interruptions. In addition, we would like to investigate the influence causes by the measuring time duration of the user's latest history. We would like to find out if longer measuring time will enable better representation of the user's profile. Finally, interesting questions may address what is the sufficient level of model accuracy needed to improve performance. Is the 75% level of accuracy our model achieved in 20 seconds sufficient, or should the aim be for a more accurate model, but with a longer training period? We are confident that the novel synthesis of Collaborative Filtering and traditional learning methods presented in this paper will stimulate interesting innovations in the future.

References

1. P. Adamczyk and B Bailey. If not now, when?: the effects of interruption at different moments within task execution. *In CHI 04*, pages 271–278, 2004.
2. Charu C. Aggarwal, Joel L. Wolf, Kun lung Wu, and Philip S. Yu. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *In Proceedings*

of the Fifth ACM SIGKDD International Conference on Knowledge discovery and data mining, pages 201–212. ACM Press, 1999.
3. R. M. Bell and Y. Koren. Improved neighborhood-based collaborative filtering. In *Proc. KDD-Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.
4. Breese, Heckermen, and Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Technical report, Microsoft Research*, 1998.
5. B.V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. Computer Society Press, 1991.
6. M. Fleming and R. Cohen. A user modeling approach to determining system initiative in mixed-initiative ai systems. *In UM 01*, pages 54–63, 2001.
7. Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, New York, NY, USA, 1999. ACM Press.
8. E. Horvitz and J. Apacible. Learning and reasoning about interruption. *In ICMI 03*, pages 20–27, 2003.
9. C.-S. Hwang and P.-J. Tsai. A collaborative recommender system based on user association clusters. *Lecture Note on Computer Science*, vol. 3806:463–469, 2005.
10. Ece Kamar, Barbara J. Grosz, and David Sarne. Modeling user perception of interaction opportunities in collaborative human-computer settings. In *AAAI 2007*, pages 1872–1873, 2007.
11. George Karypis. Evaluation of item-based top-n recommendation algorithms. In *CIKM*, pages 247–254, 2001.
12. R. Lin, S. Kraus, and J. Tew. Osgs - a personalized online store for e-commerce environments. *Information Retrieval journal*, vol.7(3–4):369–394, 2004.
13. G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing 7*, pages 76–80, 2003.
14. S. E. Middleton, N. R. Shadbolt, and D. C. De Roure. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS) 22(1)*, pages 54–88, 2004.
15. John Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
16. D. Sarne and B. J. Grosz. Timing interruptions for better human-computer coordinated planning. *In AAAI Spring Symp. on Distributed Plan and Schedule Management*, 2006.
17. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. *Proc. 10th International Conference on the World Wide Web*, pages 285–295, 2001.
18. M. Tambe, E. Bowring, J. Pearce, P. Varakantham, P. Scerri, and F. Pynadath. Electric elves: What went wrong and why. *In AI Magazine*, 2007.
19. M. Vozalis and K. G. Margaritis. On the combination of collaborative and item-based filtering. In *presented at the 3rd Hellenic Conference on Artificial Intelligence (SETN 04)*, Samos, Greece, 2004.
20. Jiangshan Xu, Liang-Jie Zhang, Haiming Lu, and Yanda Li. The development and prospect of personalized tv program recommendation systems. In *IEEE Fourth International Symposium on Multimedia Software Engineering (MSE'02)*, pages 82–89, 2002.