Leveraging Users for Efficient Interruption Management in Agent–User Systems

Tammar Shrot*, Avi Rosenfeld† and Sarit Kraus*
*Dept. of Computer Science
Bar-Ilan University
Ramat-Gan 52900, Israel
Email: {machnet, sarit}@cs.biu.ac.il
†Jerusalem College of Technology
Jerusalem 91160, Israel
Email: rosenfa@jct.ac.il

Abstract

In collaborative systems involving a user and an agent 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 create costly user interruptions. One of the most important issue concerning the initiation of information sharing in collaborative systems is the ability to accurately estimate the cost and benefit arising from those interruptions. While cost estimation of interruptions has been previously investigated, these works assumed either a large amount of information was available about each user, or only a small number of states needed consideration. This paper presents a novel synthesis between Collaborative Filtering methods with classification algorithms tools to create a fast learning algorithm, MICU. MICU *exploits the similarities between users in order to learn from known users to new but similar users and therefore requires less information on each user in compare to other methods. Experimental results indicate the algorithm significantly improves system performance even with a small amount of data on each user.*

1. Introduction

An important aspect of collaborative systems is efficient agent–user communication [1]. When working together toward a joint task it is important to share information in order to coordinate consequent actions [2]. In addition, different agents in the group often possess information required by others [3]. The need for efficient communication arises in mixed human-computer groups as well as in homogeneous computer-agent environments [4]. 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 hours of observations and human categorization, 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 this very long training period can incur a very high cost. In addition, some applications, such as on–line bidding agents have a limited time and only small number of iterations with each user making this approach impractical.

We focus on developing novel algorithms that can quickly and accurately model user's preferences. Our research was specifically motivated by the growing desire to create automated decision support units that better focus a user's attention in dynamic environments. We consider a team framework where agents and human operators teams together to share their utilities values. The idea is that the agents and humans operators might work on completely different tasks [3]. However, these tasks will eventually lead to a final joint goal. Agents and human operators have unique capabilities that be used to best contribute to the group in different ways (via different tasks). Agents can quickly process large amounts of information, a characteristic that can be critical, especially in dynamic and time sensitive environments. 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 group incurs a cost every time the agent interrupts the person to obtain the information she may have. 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. This decision is complicated by the fact that 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 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 Collaborative Filtering 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. Collaborative 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 [5], [6].

We propose a new approach MICU (Managing Interruption via Clustering Users). MICU is the approach of combining components of Collaborative Filtering algorithms with basic classification algorithms such as the *C4.5 Decision Tree* algorithm [7] or the *k-nearest neighbor ($k-NN$)* algorithm [8]. This novel synthesis between Collaborative Filtering and traditional methods represents significant differences from these base methods.

The advantage of MICU 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. Additionally, in contrast to machine learning algorithms, MICU uses two different phases to decide about the situations' classification using different type of data in each phase.

MICU is also significantly different from traditional Collaborative Filtering algorithms in three major ways. 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 machine learning tools are used instead in order to determine "similarity" among users and interruptions. Second, several traditional Collaborative Filtering algorithms work by identifying the precise value of a current parameter state (e.g. the genre of a movie) [6]. MICU allows a range of possible parameter values. The result is an algorithm that can effectively make decisions without precise state information. Third, MICU has no need for personal information (demographic information, economic and marital status, age, etc.) typically used in Collaborative Filtering algorithms [6]. MICU focuses exclusively on the user's interaction with the agent, and thus does not require sharing any personal information.

2. Related Work

In this paper we propose MICU, a new method for managing interruption management in mixed human–agent groups based on combining Collaborative Filtering methods with traditional machine learning methods. In this section we present related work from Human–agent interruption management, collaborative filtering, and machine learning methods.

2.1. Cost Estimation of Interruptions

One of the most important issues concerning the initiation of 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 impact on the group's performance [9]. Interruptions have two ways to negatively affect the users, a long term effect and a short term effect. Both the long and short term effects must be taken into account when calculating the cost of the interruptions.

Previous works have investigated how to estimate the cost of interruptions [10], [11]. Fleming and Cohen [9] 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. However, they assume having statistical data about the users' knowledge and utilities values. Horvitz and Apacible [10] studied the *user's* benefit from receiving information from a computer agent, while our study focus on the opposite situation, in our study the *agent* wishes to receive information from the user. Tambe et al. [11] focused on when to turn control to the user, instead of information transfer. Sarne and Grosz [2] offered an efficient model to manage the agent's information and to decide when to interrupt the user. Kamar et al. [4] 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. Iqbal and Bailey [12] developed OASIS, a system to identify breakpoints in users activities and delay the interruptions till these breakpoints.

The key difference between our research and previous works is that 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. In contrast, work by Iqbal and Bailey [12] required long and expensive training periods as each subject had to use the system for at least 90 minutes and later manually reviewed a video and marked the appropriate times for interruptions. Similarly, work by Sarne and Grosz [2] needed a large amount of information on each user before it can begin to operate at an efficient level. Work by Kamar et al. [4] 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 [13], [14], [6]. 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 [15], [16], [17]. That is, if user u_1 and user u_2 shown similar taste then items that u_1 likes will be offered to u_2 and vise verses. Users' similarity is calculated by comparing users' history and identifying similar ranks to the same items. Item Based models recommend items that are most similar to the set of items the active user has rated [18], [19], [20]. That is, if user u liked item i_1 then items found to be similar to i_1 will be offered to u. Items' similarity is calculated based on the items' purchased history.

Karypis [21] 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. [22] 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).

2.3. Machine Learning

In this paper we use two machine learning classification algorithms, the *C4.5 Decision Tree* algorithm [7] and the *k-nearest neighbor ($k-NN$)* algorithm [8].

The *C4.5* algorithm [7] is an algorithm used to generate a decision tree. *C4.5* builds decision trees from a set of training data in the labeled database, using the concept of information entropy. *C4.5* is a model–based learning where most calculations, building the model, are done a priori.

The *k-nearest neighbors ($k-NN$)* algorithm [8] is a method for classifying objects based on closest training examples in the given labeled database. *k-nearest neighbors* is a type of instance–based learning, or lazy learning where the function is only approximated locally and the computation is done only at classification time.

Hundreds of past works combine several machine learning algorithms in order to achieve better results than each of them achived individually. Huang et al.'s [23] approach was to use both machine learning tools that discover global structure in the data with machine learning tools that discover local structures in the data and combine the results in order to achieve the best in both world and learn as much as possible from the given data. This approach is the closest to ours. However, while hybrid approaches generally use different algorithms in each phase, they use exactly same data in both phase. Our approach differs as we collect different types of information, the user's task information and general domain information, and thus use different information in each phase.

In this paper we present MICU, a novel variation of traditional learning algorithms that applies the tools and principles defined in the hybrid User and Item Based Collaborative Filtering methods in order to incorporate them into our learning algorithm. By leveraging these approaches we can quickly learn about new users from known users. In the next section, we present specifics of the Interruption Management Domain we studied, as well as specifics of our algorithms.

3. MICU – A Synthesis between CF and Machine Learning

We generally model the Interruption Management's joint agent-human 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 subtask entitles the group to a certain gain. Different types of task have different gain values based on the relative worth of these tasks to the group. In addition, successfully transferring desired user information to the agent entitles the group to a certain different gain, this value changes according to the information accuracy.

Our proposed algorithm, MICU, (Figure 1) is motivated by the Collaborative Filtering hybrid approach. Just like in the hybrid Collaborative Filtering algorithm ([21], [22]), MICU has two phases: a "user" phase (Lines 1-3 in the algorithm), and an "item" phase (Lines 4-6 in the algorithm). The first phase uses user specific data in order to identify similar users in the database and construct an environment of similar users (user's neighborhood). 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 "similar" users that were discover in the first phase (defined herein).

While the algorithm's structure is motivated by Collaborative Filtering methods, the tools actually used to decide the user's neighborhood (similar users) and if an interruption should take place are machine learning tools. Collaborative Filtering methods cannot feasibly be implemented on the given data because in mixed agent–user domains, unlike in Collaborative Filtering, users cannot be compared according to shared habits and items (interruptions profiles) cannot be compared according to shared history, since this information does not exist. Consequently, different methods must be found to model new users's and items's similarities. MICU (Figure 1) contains information from old and different situations that were already examine. In these situations the outcome of the interruption is already known. Therefore, it is possible to label these interruptions as either good or bad interruption timing. Since our data can be 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 database of all users' characteristics or history.

The basic element of information used is *user's state* (u_s). A *user's state* is a vector that describes the state the user is in a given time. The u_s's vector contains numerical or other discrete values for different attributes. These attributes include information such as the user's location, percentage of the task accomplished, time left to complete the task, or the location of the nearest subtask to be accomplished. All values in the vectors' attributes are normalized to the same scale. Different domains might have different attributes associated with the users' states. However, all vectors in a given domain will have the same attributes. A *time* element is an element $t \in \{0...T_{max}\}$ that represents the time that has passed since the beginning of the task (t was in seconds in our experiments). A *timed state* (ts) is a pair (t, u_s) that represents the user's state at time t.

As mentioned earlier, we use two types of data – user's specific data and state's specific data. The user's specific data is the *user's latest history* data (h). User's history h is a collection of k *timed states* (k was equal 5 in our experiments) gathered within a short period of time defined as T_{samp} (20 seconds in our experiments). The second data type - the state specific data is the *interruption profile* (ip). ip is a *timed state* that represents the user's state immediately prior to the time of the interruption.

A *situation* (s) is a pair of user's latest history h and the interruption profile ip that immediately follows it. That is, $s = (h, ip)$ s.t. $ip = (t^*, u_s^*)$ and $\forall (t', u_s') \in h \ (t^* - T_{samp}) \leq t' < t^*$. Therefore, a situation is constructed from two distinct data types, both user's specific and state specific data, and gives us a wider point of view about the interruption's influences. A *labeled situation* is a pair $(s, label)$ that match a situation with the label of whether it is a "good" situation for interruption or not. Namely, whether the group's gain from this interruption higher than the cost of the interruption.

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

Figure 1. **MICU. An algorithm for deciding whether it is a good or bad timing to interrupt the user right now.**

MICU's (Figure 1) input is a small database of labeled situations (db) and a non labeled situation s, that we wish to discover whether it is a "good timing" or not. In the first phase ("user" phase) the algorithm builds a "user's" similarity model between the new given situation (s) and the given labeled situations in the database (db) (Lines 2 and 3 in the algorithm). This phase uses only the first data type in the situations (h) - the historical data. The users' similarity model is built according to the similarity between the "user's latest history" data type in the situations. Specifically, h is a set of k vectors that represents the user's behavior in the short period (T_{samp}) sampled before the interruption (20 seconds in our experiments). For each situation, the algorithm calculates the average changes in the user's attributes value throughout the latest history measuring time (Line 2 in the algorithm). Meaning, it calculates how each attribute's value changed (on average) between the sampling in h. The user's profile is the vector of averages changes. Namely, the algorithm uses h in order to calculate the average change in the user's behavior (values) in this time sequence. Then, the user's similarity between two situations 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 (s) and the given labeled situations. Once the user's similarity model is completed, the algorithm takes the l most similar situations (user's similarity) in the database as the new situation's neighborhood, and uses this neighborhood in the algorithm's next stage. Notice, that this data is calculated based on the *k-nearest neighbor ($k-NN$)* algorithm [8] 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 data type in the situation

(ip). 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 situation's neighborhood in the first phase it uses a machine learning classification algorithm on the items that belongs to the neighborhood's situations and returns the calculated classification.

The net result is that once a new situation arrives, the algorithm needs 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 the second phase of MICU is meant to work with any machine learning algorithms, we specifically considered two classification algorithms: the *C4.5 Decision Tree* algorithm [7] and the *k-nearest neighbor ($k-NN$)* algorithm [8].

4. Experimental Results

In order to study MICU and its abilities we conducted a number of experiments. We investigated if the algorithm is indeed efficient; how the size of the initial database effects its performances, and whether changing the level of heterogeneity in the users population reduces its effectiveness over naive classification algorithms. In this section we provide details about these experiments, as well as the description of the environment used for conducting our experiments.

4.1. Experimental Environment

We are currently conducting experiments with a game setting called "Final Frontier Trader"[1]. The goal of the game is to destroy all enemy ships and asteroids. For every asteroid or ship destroyed the group (one human operator and one agent) 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 number of points she earns from successful task completion. Additionally, the user can increase the group's gain by answering questions from the agent ($comGain(d)$). 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 $comGain(d)$ rewarded to the group is reduced. The joint final goal of the group is to increasing the group's profit, the human's individual main goals is to play the game and the agent's individual main is to get answers to as many question as possible. 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 as different people will take different lengths of time to correctly answer a question, and will suffer from different levels of distractions from their main goal. 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 group's 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 (u_s) either for user's history (h) or interruption profile (ip).

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 questions.
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 the user's performances and make sure that they interrupted the users. Scenario 3 was the experimental scenario used to evaluate the effectiveness of MICU.

Each subject performed the entire experimental protocol as described above. The subject's state and status were sampled every 5 seconds (user's 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

1. http://fftrader.sourceforge.net

the differences in the information gather before and after the interruptions as well as the final score the user achieved in the game. For example, if the group's gain from the communication was lower than the bonus the group would have got from completing the goals earlier - this was a "bad" interruption. Other examples, interruptions that cause a user to loss life and she eventually died, or interruptions that got a user far away from her subgoals and she lost due to timeout.

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 labeled database. The rest of the data was used as unlabeled situations in order to test algorithm. The situations from the test data were given as new situations that must be analyzed. The algorithm's decisions were compared to the situations' labels. Four algorithms were examined:

- A naive $k-NN$ classification model.
- A naive $C4.5$ decision tree classification model.
- The MICU algorithm using $k-NN$ in the second phase ($MICU+KNN$).
- The MICU algorithm using $C4.5$ decision tree in the second phase ($MICU+C4.5$).

4.2. Testing the Algorithms

We posit that MICU 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 65 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 16 interruptions per game). Therefore, we had approximately 1000 situations ($s=(h,ip)$) in our database. For each situation we had a short history from the user's activity in the previous 20 seconds ($h=\cup_{0<i<5}ts_i$) which was used to identify users' similarities, and an interruption's profiles (ip). The games 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 Clustering" approach in MICU. This is due to the fact that if all situations were similar – we could use the naive classification algorithms.

The results, as presented in Figure 2, support our claims. They show that in a heterogeneous environments users using MICU significantly improves the system performances. We can see that there is a significant change ($p<0.01$) between the naive algorithms and the MICU algorithms. In addition, it seems that there is no significant difference ($p>0.05$) between the naive $C4.5$ decision tree algorithm and the naive k-nearest neighbor ($k-NN$) classification algorithm. It is interesting that both naive algorithms ($C4.5$ and $k-NN$ algorithms) were improved by a very similar factor with no significant difference ($p>0.05$) between MICU using $k-NN$ and MICU using $C4.5$ algorithms. It shows that both algorithms can significantly benefit from using the "User Clustering" approach in heterogeneous environments.

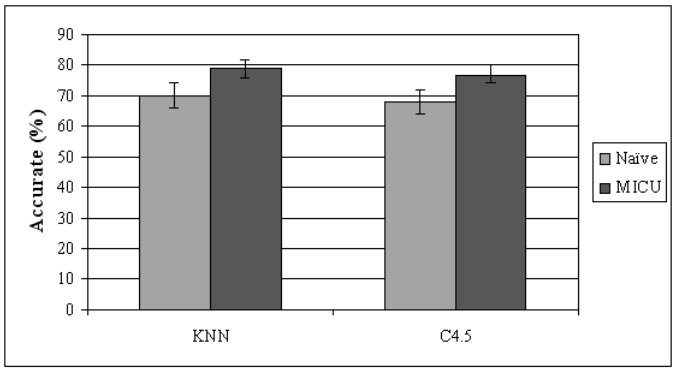

Figure 2. The algorithm accuracy percentage as a function of the different algorithms.

4.3. The Effect of Adding Labeled Situations to the System

All the algorithms discussed in this paper assume that there is an initial labeled database from past interactions that can be used in the classification algorithms. However, questions arise as to how much initial training is needed before the system can be used. Therefore, we studied how the size of the 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 change the number of situations ($s=(h,ip)$) used in the experiments.

Several interesting phenomenon appear in Figure 3. First, as stated in Section 4.2, using "User Clustering" 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 contains 100 situations. Second, it seems that while the number of situations was smaller than 200 the algorithms were less stable. However, when the number of situations exceeds 200 all algorithms became stable, including the MICU algorithms that use less data in order to build their final decision model (since they only use data from similar users found in the first phase). Apparently, the similarity between users is strong enough to achieve a significant improvement over the naive algorithm even with a small amount of data.

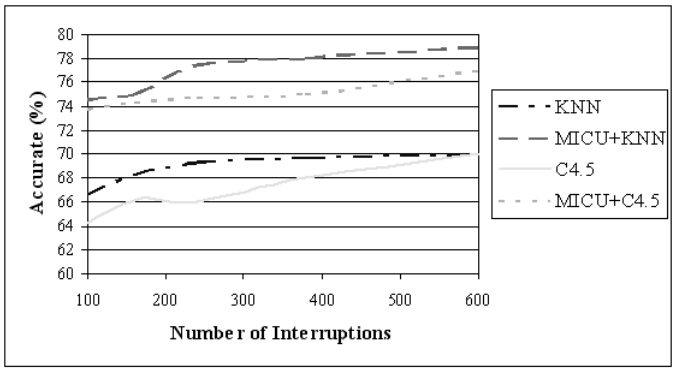

Figure 3. The algorithms accurate percentage as a function of the number of situations.

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 MICU algorithms. This is because the Collaborative Filtering "User Clustering" approach learns based on user's differences.

To understand differences between users, we divided them into the following four groups: the "gamer" group (the best players); the "skilled" group; the "fair" group; and the "hopeless" group (worse players). 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 (two subjects total) were removed from all our experiments, leaving us with 65 subjects.

Since each subject donated a different number of situations (s) to the database the number of subjects was ignored and the number of situations was kept constant during this experiment. The "level of heterogeneous" was modified by controlling the ratio of situations that came from each group.

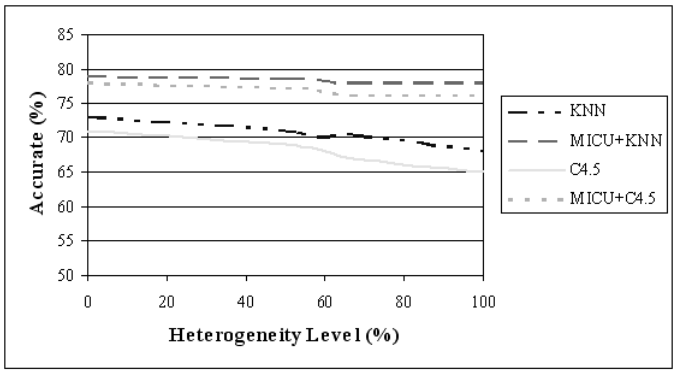

Figure 4. The algorithms accurate percentage as a function of the heterogeneity's level.

The results in Figure 4 show that the MICU algorithms perform 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 MICU 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 MICU 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. The interesting result is that changing the heterogeneity level barely disturbs MICU.

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 200 situations, MICU was 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 perform "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 MICU 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 MICU. 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 group's value, but rather provides the group 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, i.e., in a mix–initiative environment where the user and agent share the same goal and the interruptions are questions regarding common actions. Finally, interesting questions may address what is the sufficient level of model accuracy needed to improve performance. Is the level of accuracy MICU achieved in 20 seconds sufficient, or should the aim be for a more accurate model, but with a longer training period as in [12]? 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.

6. Acknowledgments

This research is based upon work supported in part under NSF grant 0705587 and ISF grant #1357/07. Sarit Kraus is also affiliated with UMIACS.

References

[1] P. Adamczyk and B. Bailey, "If not now, when?: the effects of interruption at different moments within task execution," *In CHI 04*, pp. 271–278, 2004.

[2] 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.

[3] L. Dabbish and R. Kraut, "Controlling interruptions: Awareness displays and social motivation for coordination," in *In Proc of CSCW 2004, ACM Press*. ACM Press, 2004, pp. 182–191.

[4] E. Kamar, B. J. Grosz, and D. Sarne, "Modeling user perception of interaction opportunities in collaborative human-computer settings," in *AAAI 2007*, 2007, pp. 1872–1873.

[5] 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.

[6] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. 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*. New York, NY, USA: ACM Press, 1999, pp. 230–237.

[7] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[8] B. Dasarathy, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. Computer Society Press, 1991.

[9] M. Fleming and R. Cohen, "A user modeling approach to determining system initiative in mixed-initiative ai systems," *In UM 01*, pp. 54–63, 2001.

[10] E. Horvitz and J. Apacible, "Learning and reasoning about interruption," *In ICMI 03*, pp. 20–27, 2003.

[11] M. Tambe, E. Bowring, J. Pearce, P. Varakantham, P. Scerri, and F. Pynadath, "Electric elves: What went wrong and why," *In AI Magazine*, 2007.

[12] S. T. Iqbal and B. P. Bailey, "Effects of intelligent notification management on users and their tasks," *In CHI 08*, pp. 93–102, 2008.

[13] J. Xu, L.-J. Zhang, H. Lu, and Y. Li, "The development and prospect of personalized tv program recommendation systems," in *IEEE Fourth International Symposium on Multimedia Software Engineering (MSE'02)*, 2002, pp. 82–89.

[14] Breese, Heckermen, and Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Technical report, Microsoft Research*, 1998.

[15] C. C. Aggarwal, J. L. Wolf, K. lung Wu, and P. 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*. ACM Press, 1999, pp. 201–212.

[16] R. Lin, S. Kraus, and J. Tew, "Osgs - a personalized online store for e-commerce environments," *Information Retrieval journal*, vol. vol.7, no. 3–4, pp. 369–394, 2004.

[17] S. E. Middleton, N. R. Shadbolt, and D. C. D. Roure, "Ontological user profiling in recommender systems," *ACM Transactions on Information Systems (TOIS) 22(1)*, pp. 54–88, 2004.

[18] C.-S. Hwang and P.-J. Tsai, "A collaborative recommender system based on user association clusters," *Lecture Note on Computer Science*, vol. vol. 3806, pp. 463–469, 2005.

[19] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," *Proc. 10th International Conference on the World Wide Web*, pp. 285–295, 2001.

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

[21] G. Karypis, "Evaluation of item-based top-n recommendation algorithms," in *CIKM*, 2001, pp. 247–254.

[22] 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.

[23] K.-z. Huang, H.-q. Yang, I. King, and M. Lyu, *Machine Learning: Modeling Data Locally and Globally*. Springer-Verlag New York Inc, 2008.