Automatic Creation and Updation of User Behavior Profile

Y.Sushma [1], J.Ramesh [2]

[1]M.Tech Student, Dept of CSE, QIS College of Engineering & Technology, Ongole, Prakasam Dist, A.P, India
[2]Assistant Professor, Dept of CSE, QIS College of Engineering & Technology, Ongole, Prakasam Dist, A.P, India

ABSTRACT

Now a day's there is a huge requirement for the knowledge on the computer. Therefore there should be an minimum knowledge on the system is very much beneficiary. Here the main strategy of the system is easily and efficiently identifying the behavior of the user in such a way that which is very robust for the implementation of the system respectively. Environments equipped with intelligent sensors can be of much help if they can recognize the actions or activities of their users. If this activity recognition is done automatically, it can be very useful for different tasks such as future action prediction. Although there are several approaches for recognizing activities, most of them do not consider the changes in how a human performs a specific activity. We present an approach used for creating and updating automatically the profile of a computer user is used called evolving agent behavior Classification based on distributions of relevant events(EVABCD). This approach depends upon the observed behavior of the user. It changes such interest specified in the profile of the user automatically when the user's search interest changes and to retrieve search results based on that interest. The behavior of the user is characterized by the commands that the user types. The EVABCD approach along with proposed features is highly efficient to adapt to the evolving user profiles and return better search results according to Evolving user's interest.

KEYWORDS: *User profile, activity recognition, Evolving fuzzy rule based Classifiers*

1. INTRODUCTION

Would it not be interesting to recognize a computer user and to know how (s)he will behave after (s)he types a few commands? Recognizing the behavior of others in real time is significant in different tasks, such as to predict their future behavior, to coordinate with them or to assist them. In order to act efficiently, humans usually try to recognize the behavior of others. New theories claim that a high percentage of the human brain capacity is used for predicting the future, including the behavior of other humans[1].

Specifically, computer user modeling is the process of learning about ordinary computer users by observing the way they use the computer. This process needs the creation of a *user profile* that contains information that characterizes the usage behavior of a computer user. Experience has shown that users themselves do not know how to articulate that what they do, especially if they are very familiar with the task they perform. Computer users, like all of us, leave out activities that they do not even notice they are doing. Thus, only by observing users we can model his/her behavior correctly [2]. However, the construction of effective computer user profiles is difficult problem because of the following aspects: human behavior is usually erratic, and sometimes humans behave differently because of a change in their goals.

In recent years, significant work has been carried out for profiling computer users. In this research, an approach for profiling and recognizing *general* user behavior profiles is proposed. This approach is called ***EVABCD*** (Evolving Agent Behaviour Classification based on Distributions of relevant subsequences of commands) and can be applied for creating and recognizing any behavior represented by a sequence of commands(or events). EVABCD creates a user profile as a distribution of a relevant subsequeces and then statistical

methods are applied for recognizing a given sequence of commands.

However, for evaluating EVABCD, the UNIX operating system environment is used. The creation of the unix user profiles from a sequence of unix commands should consider the sequentiality of the commands typed by the user and temporal dependencies. In a human computer interaction by commands, the sequentiality of these commands is essential for the result of the interaction. This aspect motivates the idea of automated sequence learning for computer user behavior classification; if we do not know the features that influence the behavior of a user, we can consider a sequence of past actions to incorporate some of the historical context of the user.

Therefore there is a huge challenge in order to satisfy the above phenomena respectively and also not only this one factor there are several lot of the factors based on this particular system respectively in an well equipped fashion basis. Therefore that strategy of the system is to recognize the behavior based on the priority, interest, behavior in an order stipulated basis.

2. RELATED WORK

Over the last decade there has been growing interest in the interpretation of human behavior, including in video [6]. However, most of these works are limited because human activity is a complex process which is difficult to model, especially its dynamic aspect. In this research, we present an approach to recognize a human activity directly from the sensors readings collected in an intelligent home environment, without complex.

There are many research studies about activity recognition in intelligent environments which can be applied to many real-life, humans centric problems.[7][8] However, several challenges need to be addressed before intelligent home environments technologies can be deployed. One of these main challenges is the design of powerful activity recognition algorithms.

Most of the current research studies focus on recognizing simple human activities and the recognition of complex activities are only beginning to emerge. Some of the recent approaches to activity recognition are based on probabilistic models, such as hidden Markov models [13] are Bayesian networks [9], [10] because sensor reading are noisy and activities are usually performed in a non deterministic way. Other models for recognizing activities are logic based [11] or hand-crafted [12].

There are different methods to find out relevant information under the human behavior in many different areas: Macedo et al. [3] propose a system (WebMemex) that provides recommended information based on the captured history of navigation from a list of known users. Pepyne et al.[4] describe a method using queuing theory and logistic regression modeling methods for profiling computer users based on simple temporal aspects of their behavior. Gody and Amandi [5] present a technique to generate readable user profiles that accurately capture.

The evolving user behavior classifier is based on evolving fuzzy systems and it takes into account the fact that the behavior of any user is not fixed, but is rather changing. In [5] Amandi and Godoy proposed an approach can be applied to any behavior represented by a sequence of events.

- In order to classify an observed behavior, as many other agent modeling methods, creates a library which contains the different expected behaviors. This library is not prefixed, but it is evolving and learning from the observations of users behaviors and it will be filled from scratch by assigning temporarily to the library the first observed user behavior as prototype.

- The Evolving Profile Library (EPLib), is used to store the different expected user's behaviors and it is continuously changing. Trie based approach with the classifier is being used.

The following are the two main action proposed by Gong and Pepyne [4]:

1. Creating and evolving the classifier. This action involves 2 sub actions.
 a. Creating the user behavior profiles. This sub action analyzes sequences of commands typed by the users online and creates respective profiles for the users.
 b. Evolving the classifier. This sub action includes update of the classifier. The profiles of the user changes automatically whenever search interest of the user changes.
2. User classification. The user profiles created are associated with one of the prototypes from the Evolving Profile Library. New prototypes are added and existing prototypes are removed.

3. PROPOSED APPROACH

The proposed system is designed with an well equipped fashion where the system is to completely overcome the drawback of the several previous existing techniques respectively.

3.1 Administrator Process

In this administrator process the administrator or authorized user adds dataset or webpage details in the server database. The administrator add website details, all details should be valid format like title should contain the strings, numbers, special symbols. Administrator will be able to view the profile of the user and such that able to detect any abnormalities in the user behavior. The administrator will authorize the user by validating their user name and password.

3.2 User Process

This section introduces the proposed approach for clustering, classifier design, and classifies the behavior profiles of user that includes concept based and history based search facility. The history based search enable the user to navigate frequently accessed pages easily. It updates the profile of the user automatically when his/ her preference of search changes. And also it deletes the unused prototypes for long time and updates with new prototypes in the evolving profile library.

Although EVABCD can be applied for creating and recognizing any behavior profile represented by a sequence of commands, this research is focused on creating computer user profiles from command-line interface. Specifically, EVABCD is detailed using the unix command environment. EVABCD as other behavior modeling methods [14], uses a library in which all the different user profiles recognized are stored. Then a matching of the sequence to classify with the *Profile-Library* is done. Thus, EVABCD is divided into 2 phases:

1. Creation of the User Behavior Profiles:
In this phase, the sequence of commands typed by the users are analyzed and the corresponding profiles are created and stored in the *Profile-Library*. This process is detailed in section4.

2. User Classification: The goal of this phase is to classify a new sequence of commands typed by a user into one of the profiles created in the previous phase. Section 5 explains the proposed statistical classification method.

4. USER BEHAVIOUR PROFILES CREATION

In this phase, the first step is to extract the significant pieces of the sequence of commands that can represent a pattern of behavior. When a user types a command, it usually depends on the previous typed commands and it is related to the following commands. According to this aspect and as it was used in [15], in order to get the representative set of sub sequences from the acquired sequence, the use of a trie data structure[16] is proposed.

The construction of a user profile from a single sequence of commands is done by a three steps process:
1. Segmentation of the sequence of commands,
2. Storage of the subsequences in a trie, and
3. Creation of the user profile.

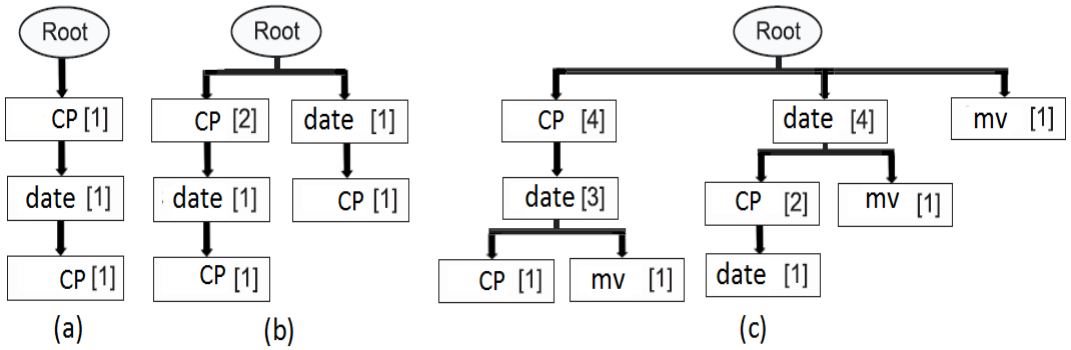

Fig.1.Steps of creating an example trie

These steps are detailed in the following 3 sub sections.

In order to clarify the process for creating a UNIX user profile, let us consider the following sequence as example: {cp → date → cp → date → mv}.

Segmentation of the sequence of commands: In the first step the given sequence is segmented into subsequences of equal length from the first to the last element. Suppose consider the sequence A= A_1A_2….An (where 'n' is the number of commands of the sequence) will be segmented into the subsequences described by Ai…..Ai+ length $\forall$ *i*, i=[1,n-length+1], here length is the size of the subsequences created and determines how many commands are considered as dependent. In the rest of the paper we will use the term subsequence length to denote the value of this length.

In the proposed sample sequence (*{ cp → date → cp → date → mv}*), let 3 be the subsequence length, then it is obtained: *{cp → date → cp}* and *{date → cp → date}* and *{cp → date → mv}*.

Storage of the Subsequences in *trie*: The sequences generated in the previous step are stored in trie. They are stored in a way that all possible subsequences are accessible. In the proposed trie, each node represents a command and its children represent command that follow it. Also, each node keeps track of the number of times a command has been inserted on to it. The subsequence suffixes (Subsequences that extend to the end of the given sequence) are also inserted

Considering the previous example, the first sub sequence (*{cp → date → cp}*)is added as the first branch of the empty trie (Figure 1a). Each node is labled with the number 1 (in square brackets) which indicates that the command has been inserted into the node once. Then, the suffixes of the sub sequence is added as the first branch of the empty *trie* (Figure 1a). Each node is labeled with the number 1 (in square brackets) which indicates that the command has been inserted in the node once. Then, the suffixes of (*{date → cp}* and *{cp}*) are also inserted (Figure 1b). Finally, after inserting the 3 subsequences and its corresponding suffixes, the completed trie is obtained (Figure 1c).

Creation of the user profile: For this purpose, frequency-based methods are used. Specifically, to evaluate the relevance of a subsequence using EVABCD, its relative frequency or support [18] is calculated. In this case, the support of a sub sequence is defined as the ratio of the number of times the subsequence has been inserted into the trie to the total number of the subsequences of equal size inserted. Calculating this value, the trie is transformed into a set of subsequences labeled with its corresponding support value. This structure represented as a distribution of relevant subsequences. Once a user behavior profile has been created, it is stored in the *profile-Library* with an identification name. In the previous example, the trie consists of 9 nodes; therefore, the profile consists of 9 different subsequences which are labeled with its support (figure 2).

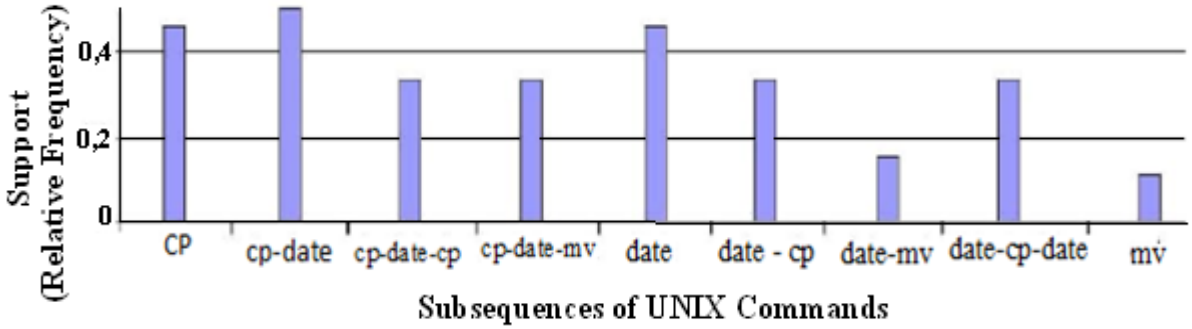

Fig. 2. Distribution of subsequences.

5. USER CLASSIFICATION

In this second phase, a new sequence of commands typed by one of the users previously analyzed must be classified. It means that given an observed sequence E typed by a user and a set of user behavior profiles *P = {up*1, *up*2,..., *upn}* stored in the profile-Library, the goal this phase is to determine into which profile *upi € P* the sequence E belongs to.

Firstly, the distribution of relevant subsequences of the new sequence (input) is created by applying the process explained in the previous section. Then, it is matched with all the profiles stored in the Profile-Library.

Based on the command typed by the user, the profile of the user will be updated automatically by evolving classifier [17]. The incremental learning algorithm used satisfies the following criteria's.

1. It should be able to learn additional information from new commands typed by the user.
2. If any access to the original data already exist, then it should train the existing classifier.
3. It should pressure the previously acquired knowledge .
4. It should be able to accommodate new classes that may be introduced with new data.

The sample spread is determined based on the scattered data. The equation to get spread of the Kth data sample is defined as,

$$\sigma i(k) = \sqrt{1/k \; \Sigma \text{cosDist}(\text{Proti},zk) nk=0}$$

Where $\sigma i(0)=1$

Here K is the number of data samples inserted in the data space; cosDist(proti,zk) is the cosine distance between the new data sample (zk) and ith prototype.
To update recursively,

$$\sigma i(k)=\sqrt{[\sigma i(k-1)]2+1/k \; \text{cosDist}(\text{Proti},zk)-[\sigma i(k-1)]2}$$

where k is the number of data samples inserted.

Once the corresponding distribution has been extracted from the data output stream, then it is processed by the evolving classifier approach.

6. EXPERIMENTAL SETUP AND RESULTS

To evaluate EVABCD in the UNIX environment, we can use a data set with the UNIX commands typed by 168 real users and labeled in the 4 different groups. Therefore, in these experiments we will use supervised learning.

Data Set

For evaluating EvABCD in the Unix environment, we have used the command-line data collected by greenberg [19] using unix csh command interpreter. In these data, 4 target groups were identified, representing a total of 168 male and female users with a wide cross section of computer experience and needs. Salient features, the size of the data stream (the number of people observed) the command lines of each group are described below.

Novice Programmers: The users of this group had little or no previous exposure to programming, operating systems, or Unix-like command-based interfaces.
Sample: 55 Users and 77423 command lines.

Experienced Programmers: In this group, the members were senior computer science under graduates expected to have a fair knowledge of the Unix environment.
Sample: 36 users and 74906 command lines.
Computer Scientist: This group had varying experience with unix, although all were experts with computers.
Sample: 52 Users and 125691 command lines.
Non-Programmers: Document preparation was the dominant of the activity of the members of this group. Knowledge of unix was the minimum necessary to get the job done.
Sample: 25 users and 25608 command lines.

The number of UNIX commands analyzed per user is very relevant for the result of the classification. Using EvABCD in a real application, after a user has typed a particular number of commands, its behavior can be classified and the evolving behavior library updated. However, in order to use all the data we have, in this experiment all the commands the user has typed during long period of time are used. For this reason a distribution (which represent a behavior) is represented by a very large number of subsequences. And if the number of users increases, the number of different subsequences increases, too.

In the phase of behavior model creation, The length of subsequences in which the original sequence is segmented (used for creating the trie) is a relevant parameter; using longer sub sequences, the time consumed for creating the trie and the number of relevant sub sequences in the corresponding distribution increase drastically. In the experiments presented in this paper, the subsequeces length value was selected to be 3.

Results

Although the sequence length is small 93 commands), the number of commands typed per user is large; thus, the number of different sub sequences of commands created per user is very large.The number of different sub sequences is shown per group as follows; novice programmers: 25614, experienced programmers:43049, Computer scientists:66490, non programmers: 10572. Also, the number of different sub sequences of commands typed by the 168 users is 135317.

According to this data, after applying EvABCD using the explained experimental design, the percentage of users correctly classified into its corresponding group is: 100% validation data! Therefore, this result shows that the proposed classifier works excellent in this kind of environments.

The number of prototypes created per group is important, too. As we have used 1—fold cross validation, the number of different prototypes created in each of the 10 runs shown in table 1. This number varies depending on the heterogeneity of the data.

	Prototypes in each of the 10 runs									
Group	1	2	3	4	5	6	7	8	9	10
Novice Progr.	3	2	7	2	2	2	3	4	2	2
Exp. Progr.	2	3	3	2	2	2	2	2	2	2
Comp. Scientists	2	2	1	2	1	1	1	1	3	3
Non-Progr.	2	2	1	2	2	2	2	2	2	2

Table.1. Number of prototypes created per group

The result obtained in this experiment shows that the proposed classifier can be very useful to classify user behaviors in a dynamic environment for the example of the UNIX user profiles with a great amount of data (in this case, commands per users). In order to compare these results we consider two well established classifiers.-the algorithm C4.5 used to generate a decision tree and the K-nearest neighbor algorithm (K-NN) used to classify objects based on closest training examples in the feature space.

However, In order to make a comparison, we reduced the number of subsequences of commands per user using this support value. In this case, we consider that the sub sequences with a higher support are more relevant. The percentage of sub sequences reduced is very high and only around the 3% of the initial data were used. In this reduced dimension experiment, again the proposed EvABCD evolving classifier outperformed the well established half-line classifiers and the results are tabulated in Table 2.

Classifier	Rate of unknown users correctly classified
EvABCD	**81,54%**
C4.5	73,80%
3-Nearest Neighbor	44,64%

Table .2. COMPARATIVE RESULTS

7. CONCLUSION

A lot of analysis has been made on the present system where the system is efficient and effective in terms of the implementation based scenario. Therefore it has been applied on the large number of the datasets where the evaluation of the performance takes place. The present system is designed in such a way that it completely overcome the drawbacks of the several previous existing techniques in an effective manner and also accurate in terms of the analysis based strategy in a well respective fashion in terms of the implementation.

Here in the present method an advanced technique is designed in an well efficient way where the performance of the system is evaluated in an effective way in an accurate analysis approach. Where it completely overcome the previous problems related to the classification followed by the data extraction scenario and finally on the comparison oriented with the fusion based on the similarity score in an well respective fashion.

REFERENCES

[1]. Mulcahy, N.J., Call, J.: Apes save tools for future use. Science 312(5776), 1038–1040 (2006)
[2]. Hackos, J.T., Redish, J.C.: User and Task Analysis for Interface Design. Wiley, Chichester (1998)
[3] A.A. Macedo, K. N. Truong, J. A. Camacho-Guerrero, and M. da GraC¸ a Pimentel, "Automatically sharing web experiences through a hype rdocument recommender system," in HYPERTEXT 2003. New York, NY, USA: ACM, 2003, pp. 48–56.
[4] D. L. Pepyne, J. Hu, and W. Gong, "User profiling for computer security," in Proc. American Control Conference, 2004, pp. 982–987.
[5] D. Godoy and A. Amandi, "User profiling for web page filtering," IEEE Internet Computing, vol. 9, no. 4, pp. 56–64, 2005.
[6]. J. Hoey and J. J. Little, Value-directed human behavior analysis from video using partially observable markov decision processes, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (2007) 1118–1132.
[7]. J. Sarkar, Y.-K. Lee and S. Lee, A smoothed naïve bayes based classifier for activity recognition, IETE Technical Review (SCIE) 27(2) (2010) 109–119.
[8]. E. Kim, S. Helal and D. Cook, Human activity recognition and pattern discovery, *IEEE Pervasive Computing* 9 (2010) 48–53.
[9]. D. H. Wilson and C. Atkeson, Simultaneous tracking and activity recognition (star) using many anonymous, binary sensors (2005) 62–79.
[10]. E. Castillo, I. Gallego, J. M. Menendez and S. Sanchez-Cambronero, Traffic estimation and optimal counting location without path enumeration using bayesian networks, Computer-Aided Civil and Infrastructure Engineering 23(2) (2008) 198–207.
[11]. N. Landwehr, B. Gutmann, I. Thon, L. De Raedt and M. Philipose, Relational transformation-based tagging for activity recognition, Fundamenta Informaticae 89(1) (2008) 111–129.
[12]. J. Fogarty, Sensing from the basement: A feasibility study of unobtrusive and low-cost home activity recognition, in Symposium on User interface Software and Technology (2006) 91–100.
[13]. D. Sanchez,M. Tentori and J. Favela, Activity recognition for the smart hospital, *IEEE Intelligent Systems* **23**(2) (2008) 50–57.
[14]. Riley, P., Veloso, M.M.: On behavior classification in adversarial environments. In: DARS, pp. 371–380
[15]. Iglesias, J.A., Ledezma, A., Sanchis, A.: Sequence classification using statistical pattern recognition. In: Berthold, M.R., Shawe-Taylor, J., Lavraˇc, N. (eds.) IDA 2007. LNCS, vol. 4723, pp. 207–218. Springer, Heidelberg (2007)
[16]. Fredkin, E.: Trie memory. Comm. ACM 3(9), 490–499 (1960)
[17]. Branch J.W, Breimer E., Coull S.E and Szymanski.B.K. (2003), 'Intrusion Detection: A Bioinformatics Approach', Proc. Ann. Computer Security Applications Conf. (ACSAC), pp. 24-33.
[18]. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Eleventh International Conference on Data Engineering, Taipei, Taiwan, pp. 3–14 (1995)
[19] S. Greenberg, "Using unix: Collected traces of 168 users," Master's thesis, Department of Computer Science, University of Calgary, Alberta, Canada, 1988.