Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 35 (2014) 1547 – 1556

Procedia Computer Science

18th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems - KES2014

A considerate application prediction system with artificial neural network

Daichi Hasumi, Eiji Kamioka*

Shibaura Institute of Technology, 3-7-5 Toyosu, Koto-ku, Tokyo 135-8548, Japan

Abstract

Personal computer is one of the indispensable tools at work and in everyday life. Some of application programs in the computer are habitually used or launched in a particular time. Even though their invocations can be predicted in advance, they are executed manually in each time, hence, this results in a deterioration of the usability in computer operation. In this paper, a considerable application prediction system with Artificial Neural Network, which recommends a useful application for the user at the time, will be proposed. It refers to an application ontology and uses an application log obtained from the user's personal computer. Moreover, the effectiveness of the proposed system will be discussed showing the prediction accuracy of about 90% in recommending useful applications when the user utilizes the computer in daily life.

Keywords: Recommendation System; Machine Learning; Application Log; Artificial Neural Network

1. Introduction

A lot of tasks in daily life are processed by application software installed on a personal computer (hereafter, referred to as PC). The tasks include a wide variety of types from business tasks to private ones. It is easy to imagine that there are some characteristic tendencies of application type necessary in individual task and in each time period. For example, an email client is launched to check emails when a user arrives at work, a text editor and a web-browser are executed at the same time when a student writes a report, and so forth, depending on the user's work and life style. Therefore, it may be possible to predict the execution timing of a necessary application program based on

* Corresponding author. Tel.: +81-3-5859-8266 ; fax: +81-3-5858-8266.
E-mail address: kamioka@shibaura-it.ac.jp

doi:10.1016/j.procs.2014.08.238

the tendency of application usage in the past. Nevertheless, those applications are usually executed manually in each time.

Considering recent high functional computers, the cost in computer operation by the user, such as launching several applications at the same time cannot be ignored.

This study proposes a considerate application prediction system which recommends the most appropriate application for the user at the time, aiming at alleviating the user's stress in computer operation.

The keys to embody this system are 1) application log, 2) application ontology and 3) artificial neural network (ANN)[5]. The application log is a historical data about application execution including the time period when each application is running on a computer. The application ontology is a database that associates an application function with an application name. The artificial neural network outputs the most considerate application function that the user may want to use based on the application log, the application ontology and the current time. This paper will discuss the effectiveness of the proposed system in terms of the accuracy of recommended application function.

The remainder of this paper is organized as follows. Section 2 introduces some existing recommendation systems as related work. The details of the proposed system architecture and the key technologies mentioned above are described in Section 3. Section 4 states the performance evaluation of the proposed system. The evaluation results are discussed in Section 5 and Section 6 concludes this paper and states the future work.

2. Related works

A lot of studies on prediction of human behaviors, gestures and situations using data mining technique[1] have been reported. In this section, some recommendation systems which provide specific resources or services depending on the user's status are stated.

Osawa et al.[2] discussed a recommendation system that suggests several web applications suitable for a user's task by referring to web applications and ontology database. This system performance was evaluated in terms of the user's task completion time. As a result, actually the task completion time was reduced. However, since the system recommends suitable web applications, not by using machine learning technique but by using their own evaluation formula, the task dependency of the evaluation formula must be taken into consideration.

Butoianu et al.[3,4] developed a system which recommends documents suitable for an application being used by a student aiming at alleviating the user's stress in e-learning. The information of the application used by the student is sent to the system, and then the user profile is updated. This system also determines the documents to recommend to the user using an evaluation formula with contextualized attention metadata. Hence, the same problem of the previous research remains.

The above investigation shows that it is not a good way to use task sensitive formula in building a generic framework for recommendation systems. One of the promising approaches for recommendation systems is collaborative filtering. The collaborative filtering is an algorism by which some items in a specific category are expressed as a matrix form. It calculates the similarity between users from the matrix and recommends items to the user.

Ohsugi et al.[6] proposed a recommendation system using collaborative filtering technique to suggest application functions, which are usually not used, to the users. This system applies a similarity calculation algorithm of the collaborative filtering to the history of software function execution. They conclude that their proposed system is better than the random recommendation algorithm.

Wakiyama et al.[7] proposed a recommendation system using the user's line of sight to create the user profile which shows the user's interest. This system recommends paintings to the user by collaborative filtering without explicit user input. They showed that this proposed system enables the creation of an interest model of the user and the recommendation through the evaluation experiment.

Christina Christakou and Andreas Stafylopatis[8] discussed a hybrid movie recommendation system, which uses content-based filtering with neural network and collaborative filtering. They realized a high accurate recommendation system incorporating the movies evaluated by a user and the movies evaluated by the others into the recommendation algorithm.

As mentioned above, the effectiveness of the recommendation systems using collaborative filtering or hybrid system is high, and thus the algorithms are used for the existing recommendation services such as Amazon.com[9].

However, there are some issues. The first issue is security problem. It is necessary to share user preference to calculate the similarity between the users. User preference is a private information, hence, sharing this information implicitly could provoke the disapproval of the users and would be at substantial risk for the information leak to the third party. Furthermore, this algorithm does not work well in the field where there is little similarity to each user. The second issue is computational problem. Hybrid systems and ensemble systems which uses several prediction algorithms tend to increase the computational load of the recommendation system as the number of users increases. When the accuracy of recommendation system is high, the value of the system is also high. The computational resources, however, are finite, therefore this could be a significant problem in developing prediction systems.
To solve these problems, the proposed system uses machine learning technique. The machine learning can be customized in the training data sets so that the users should not share private information with each other and so that the computational load can be controlled.

3. Considerate application prediction system

The final goal of this study is to recommend the most appropriate application that the user wants to utilize. To realize this goal, an application prediction system is proposed using three key components which are Application Log, Application Ontology and Artificial Neural Network. Figure 1 shows the brief system architecture. The following subsections will describe the above three components in detail.

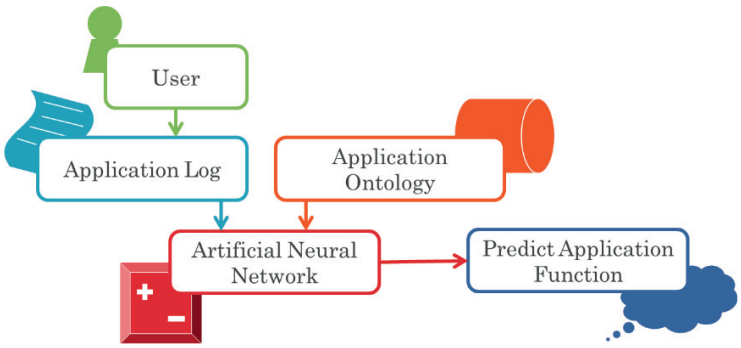

Fig. 1. Proposed system architecture.

3.1. Application log

The application log is a historical data about application execution including the time period when each application is running on a computer. It is automatically obtained on the user's laptop computer by a resident program every 10 minutes. This program has newly been developed for the purpose of this study. The application log is composed of the date and time when an application is launched, the application name and the active window name. The active window means the foreground application window on the desktop of the computer. It is used to distinguish the currently being used application from just running applications. Figure 2 shows an example of the application log. The application log used in this study has been recorded in the user's laptop computer for three month.

```
20121024-1040
Active:Word.app
Word.app
iTunes.app
Finder.app
```

Fig. 2. An example of application log.

The first line in Fig.2 means the date and time log recorded and the second line indicates that Microsoft word is running as the active window application. Subsequently, the names of applications, which are running on the computer, are listed. The “Active” flag in the second line is used to distinguish the application currently being used by the user from the applications just being launched. Note that “nolaunch” is recorded in the application log when the computer is in sleep mode or when no applications are running. It means that the user is not using the computer.

3.2. Application ontology

The proposed system finally recommends an application name, but, before that, an application function, which the user wants to use, is predicted based on currently being used application functions. As described in the previous sub-section, the application log provides only application names, not application functions. Therefore, an application ontology is introduced here to associate an application name with an application function. Figure 3 illustrates a portion of the hierarchical structure of the application ontology database created in OWL language.

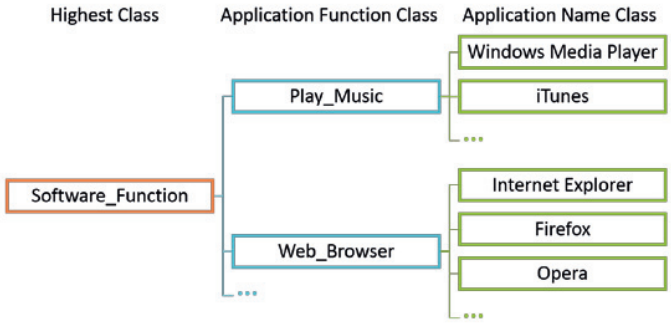

Fig. 3. A portion of hierarchical structure of application ontology database.

In the application ontology database, Software_Function is defined as the highest class and 13 application functions, which are regarded as daily use application functions, are defined as the subclass components. Table 1 shows the 13 application functions used in this study. The lowest class is composed of application name instances.

Table 1. Application functions.

Develop_C	Develop_HTML
Develop_Java	Graphic_Edit
Play_Music	Send_Mail
Text_Editor	Web_Browser
File_Browser	Object_Drawing
Presentation	Spreadsheet
SNS_Client	

3.3. Artificial neural network

The proposed system predicts the most considerate application function by machine learning technique. The machine learning technique used in this study is the most popular neural network, called the multi-layer feed-forward neural network, trained with a back-propagation learning algorithm. It is composed of neurons which are ordered into layers. Hereafter, this is called ANN in this paper.

In this study, the information obtained from the application log and the application ontology is used as data sets which are trained for the ANN. The application log is analyzed and an application function that the user wants to use is predicted in each prediction unit. Here, the prediction unit means a time interval of one hour in the day of the week. For instance, if an application, which the user wants to use, at 8p.m. on Monday is predicted, then the application log data during 7.30p.m. to 8.30p.m. on Mondays are referred to. In this case, the prediction unit is the time interval from 7.30p.m. to 8.30p.m. on Monday. The reason for using the prediction unit is the tendency of application usage for the user may be based on the user's life rhythm. For instance, the user has a life rhythm to start writing a report at 8p.m. every Monday, thus Microsoft word and a web browser are usually running at the time on Mondays. Therefore, drawing attention to a specific time interval and the day of the week is useful to predict an application that the user wants to use.

$$\text{Launch Probability} = \frac{L_b}{L_a}$$

L_a: the number of application log data
L_b: the number of application log data which shows a considered application is running

Fig. 4. Formula of launch probability.

Here, the launch probability of application is introduced due to the above hypothesis. It shows how often an application is launched in each prediction unit as shown in Fig.4, where L_a is the number of application log data and L_b is the number of application log data which shows the application is running. More concretely, when the launch probability of Microsoft word at 8p.m. on Mondays is calculated, the number of log data of the prediction unit, that is, from 7.30p.m. to 8.30p.m. on Mondays is counted as L_a. Subsequently, the number of log data which shows Microsoft word is running in the same prediction unit is calculated as L_b. Note that the launch probability is calculated not at the specific time but in the prediction unit such as from 7.30p.m. to 8.30p.m. because the user's life rhythm usually has some fluctuation.

Table 2. An example of input data set for the ANN.

Application Function	Status of PC	Launch Probability
develop_c	yes	1
file_browser	yes	0.714285714
graphic_edit	no	0.714285714
play_music	yes	0.285714286
presentation	yes	0.428571429
send_mail	no	0.714285714
spreadsheet	yes	1
text_editor	yes	1
twitter	yes	0.571428571
twitter	yes	0.142857143
web_browser	no	0.714285714

When an application function is predicted using the ANN, the launch probability, the application functions extracted from the application ontology and the PC usage status are used as the training data sets. Here, the PC usage status can be obtained from the application log as stated in section 3.1. Basically, one application function is predicted as an output of the ANN. However, if the reliability of the predicted application function is low enough, the output of the ANN shows "no recommended application function". The reliability is based on the distribution probability which is calculated by the ANN.

Table 2 shows an example of training data sets for the ANN. A data set has three attributes such as application function, PC usage status and launch probability. This type of training data sets are prepared for each prediction unit.

4. Evaluation

In this section, the effectiveness of the proposed system is evaluated in terms of prediction accuracy. This evaluation was conducted to the application log for three months. The estimation accuracy of application function was calculated in each prediction unit in the case where no application was launched and in the case where a specific application was launched. If the output result from the ANN is "no recommended application functions" and no application function is really executed in the targeted prediction unit, this prediction result is regarded as correct. The details are described in the following sub-sections.

4.1. Evaluation 1: Estimation accuracy when no application was launched

If no application is launched, the estimation of application function relies on the user's life rhythm. Figure 5 shows the prediction accuracy in each prediction unit. In Fig. 5, "Using: yes" means that no application is not launched but the user was using the PC until just before the considered prediction unit. On the other hand, "Using: no" means that no application is not launched and the PC is in sleep mode or the user did not touch the PC during the prediction unit. As shown in Fig. 5, the accuracy increases from evening to midnight, and then it decreases from early in the morning to noon in the situation of "Using: yes". This is obvious considering the user's life rhythm in general. In the situation of "Using: no", the tendency of the accuracy transition seems opposite to the "Using: yes" situation but it is not trustable since there is not enough information in the application log to predict application functions. The important thing is to focus on the prediction unit when the user is using the PC because the prediction during no PC usage is meaningless in this study.

Figure 6 presents the prediction accuracy when the user is using the PC frequently, and thus shows the effectiveness of the proposed system in reality. The prediction units considered in Fig. 6 include less than 5% of "nolaunch" log data. As shown in Fig.6, the proposed system predicted application functions with high accuracy of more than 85% on average.

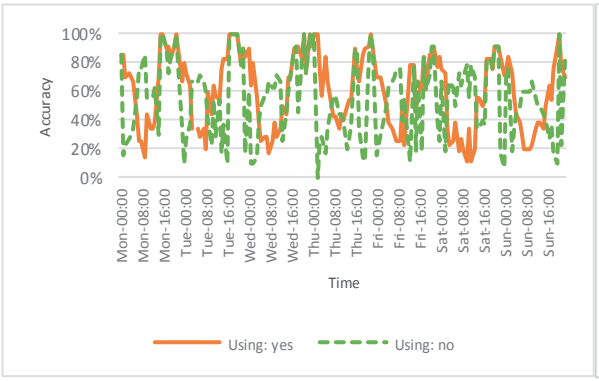

Fig. 5. Prediction accuracy when no application was launched.

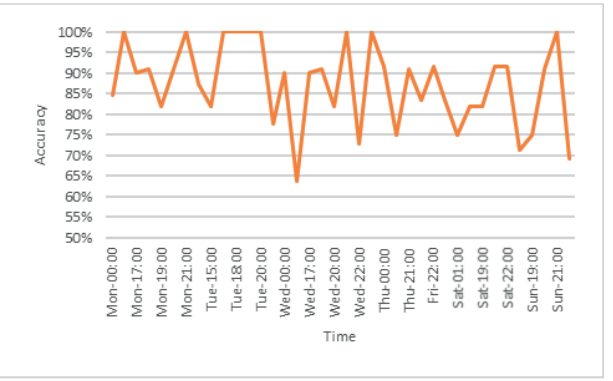

Fig. 6. Prediction accuracy when user utilized PC frequently.

4.2. Evaluation 2: Estimation accuracy when a specific application was launched

In the previous sub-section, the prediction accuracy based on only the user's life rhythm was discussed. It is obvious that the prediction accuracy would improve if the information of application functions currently being used is utilized with the user's life rhythm. The accuracy graph is here in the situation of the one of the top three frequency application function is already launched.

Web_Browser and File_Browser are the top two application functions that the user utilizes frequently among the application functions defined in the application ontology. In this sub-section, the prediction accuracy when each above application function is running singularly is discussed. Note that only the case of "Using: yes" is considered as mentioned in the previous sub-section.

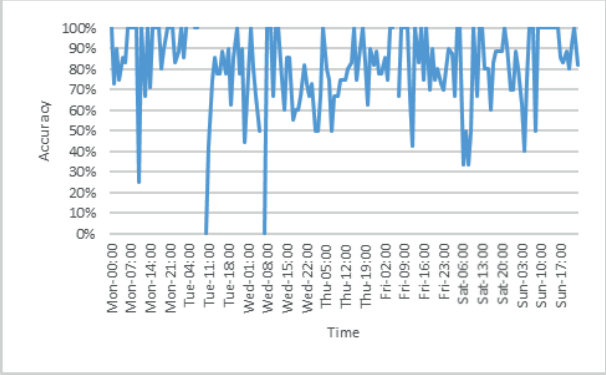

Fig. 7. Prediction accuracy when Web_Browser was running.

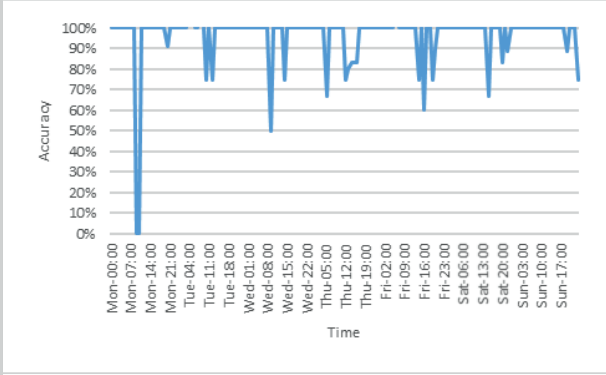

Fig. 8. Prediction accuracy when File_Browser was running.

Figure 7 shows the prediction accuracy when Web_Browser function was running singularly. In some prediction unit, the accuracy is missing because the ANN was not able to predict due to the lack of application log data. It is understandable from the fact that the prediction accuracy is 0% around the prediction unit with missing accuracy. This result is not really better than the one in Fig.6. This is because Web_Browser is a quite common application function, hence, it is always running. It means that Web_Browser does not have a special relation with the other application functions.

Figure 8 presents the prediction accuracy when File_Browser was running singularly. Compared to the result in Fig. 8, the prediction accuracy is much better. There are the prediction accuracy of 100% in many prediction units. It means that File_Browser tends to be used with other specific functions.

5. Discussion

The amount of information on application log is critically important to predict application functions with high accuracy. This issue can be divided into two cases. One is the case where the number of application log data in "Using: yes" situation is small. The other is the case where the number of application log data with "nolaunch" is large. These two cases seem to be similar but actually they are different. The former indicates that the prediction accuracy deteriorates since the number of training data sets for the ANN is small. The latter means that the useful information to predict application functions is not much but it can output the result of no prediction, in other words, the result of no recommended application functions, hence, this may result in improving the prediction accuracy. In this section, the influence of the application log data on the prediction accuracy is discussed.

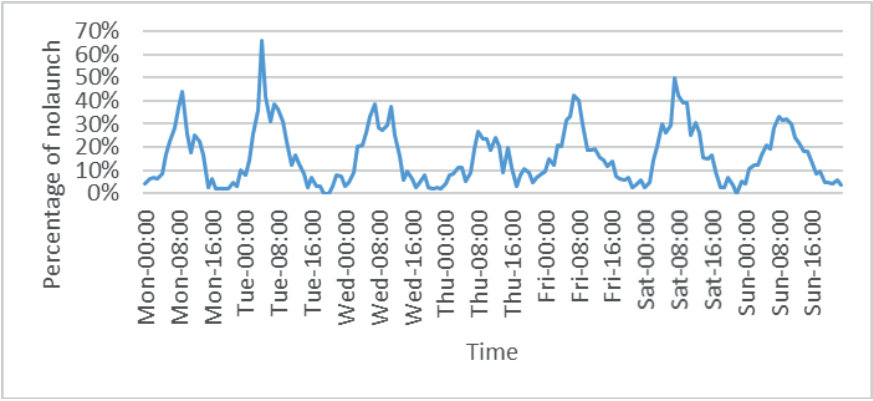

Fig. 9. Percentage of "nolaunch" log data among total log data in each prediction unit.

Figure 9 presents how many "nolaunch" log data exist among the total log data in each prediction unit. The vertical and horizontal axes indicate the percentage of "nolaunch" log data and the prediction unit, respectively. In fig. 9, there are 7 peaks in a week. These prediction units correspond to the time when the accuracy of "Using: yes" decreases and the accuracy of "Using: no" increases in Fig. 5.

Let us look at the relation between "nolaunch" rate and the prediction accuracy. Figure 10 shows the correlation between the "nolaunch" ratio and the prediction accuracy in all the time period of application log data. As you can see, there are two types of tendencies in this graph. One is the tendency where the accuracy increases quickly as the percentage of "nolaunch" decreases, the other is the accuracy increase gradually as the percentage of "nolaunch" increases. According to the investigation of the application log, the former tendency becomes strong when the number of log data is large. This is because the larger the number of training data sets for the ANN is, the higher the prediction accuracy is. On the other hand, the latter tendency becomes strong when the number of log data is small. This is because the proposed system has an answer of "no recommended application functions" as an output of the ANN. This decision is much better than wrong predictions due to the lack of training data sets. In the evaluation, if the output result from the ANN is "no recommended application functions" and no application function is really executed in the targeted prediction unit, this prediction result is regarded as correct. Therefore, the prediction accuracy increases even though the percentage of "nolaunch" increases. The significant point is how much the high percentage of "nolaunch" log data contributes to the prediction accuracy improvement.

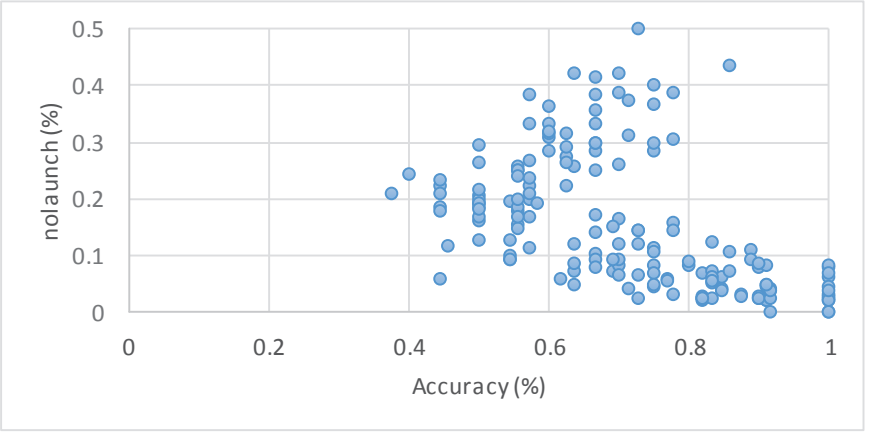

Fig. 10. Correlation between percentage of "nolaunch" log data and prediction accuracy.

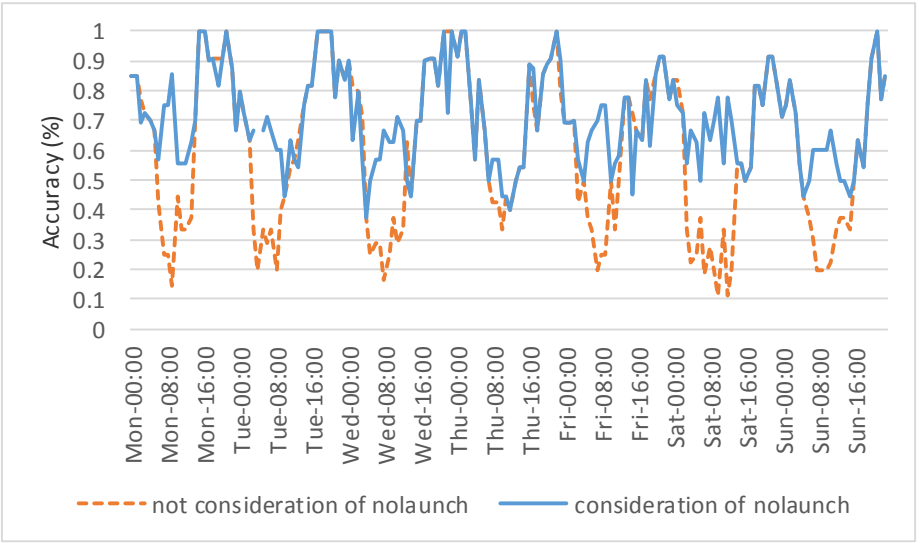

Fig. 11. Prediction accuracy taking "nolaunch" log data into consideration.

Figure 11 shows the influence of considering "nolaunch" log data on the prediction accuracy. The prediction accuracy was drastically improved. This is not just the prediction accuracy improvement but also the user satisfaction improvement. This is also a strength of the proposed system.

6. Summary and future work

In this paper, a considerate application prediction system was proposed in order to recommend an application for the user when he/her is using a PC for some work. The keys to embody this system are the application log, the application ontology and the artificial neural network, which were fully explained.

Moreover, the performance evaluation of the proposed system was stated in terms of prediction accuracy to show its effectiveness. The evaluation results showed the high prediction accuracy of about 90% on average.

In addition, the influence of "nolaunch" log data, which is the status when no application functions are running, on the prediction accuracy was discussed. The introduction of "nolaunch" tag data is useful to improve the prediction accuracy and the user satisfaction.

However, there are some issues left. First, the application log used in this study was obtained from only one subject, hence, it is not clear if the prediction accuracy could be the same for different subjects. It will be verified with more subjects' application logs in the next step. Second, the proposed system does not work until sufficient amount of application log is obtained, so-called cold start problem. One of the solution of this is non-supervisor approach. It will also be explored. Third, the predicted application functions must be investigated in terms of the user satisfaction. The main goal of this study is to reduce the stress on the user's computer operations. It will also be investigated through some subjective evaluation experiments in the future.

References

1. Usama Fayyad, Gregory Piatetsky Shapiro, Padhraic Smyth. *From Data Mining to Knowledge Discovery in Databases.* MIT Press Cambridge, 1996, pp.1-89.
2. Tetuya Osawa, Naoki Fukuta, Tadashi Iijima, Takahira Yamaguchi. *Ranking Web Application Comparison Based on Ontology*. The Institute of electronics Information and Comunication Engineers Technical Report of IEICE, KBSE 2004-43104(588), January 2005, pp.49-54.

3. Valentin Butoianu,Philippe Vidal, Katrien Verbert, Erik Duval, Julien Broisin. *User Context and Personalized Learning: a Federation of Contextualized Attention Metadata.* Journal of Universal Computer Science, Vol.16, No.16, 2010, pp.2252-2271.
4. Broisin J, Brut M, Butoianu V, Sedes F, Vidal P. *A Personalized Recommendation Framework based on CAM and Document Annotations.* Procedia Computer Science, Vol.1, Issue 2, 2010, pp.2839-2848.
5. Rumelhart DE, Widrow B, Lehr MA. *The basic ideas in neural networks.* Communications of the ACM, Vol.37, No.3, pp.87-92, March 1994.
6. Naoki Ohsugi, Akito Monden, Shuuji Morisaki, Ken-Ichi Mathumoto. *Software Function Recommender System Based on Cllaborative Filtering.* Journal of Information Processing, Vol.45, No1, 2004, pp.267-278.
7. Kouki Wakiyama, Athuo Yoshitaka, Tsukasa Hirashima. *Information Recommendation by collaborative filtering incorporated with gaze detection.* the 13th Workshop on InteractiveSystems and Software (WISS2005), 2005.
8. Christina Christakou, Andreas Stafylopatis. *A Hybrid Movie Recommender System Based on Neural Networks.* Intelligent Systems Design and Applications ISDA '05, September 2005, pp.500-505.
9. Greg Linden, Brent Smith, Jeremy York. *Amazon.com Recommendations: Item-to-Item Collaborative Filtering.* IEEE Internet Computing, Vol.7, Issue 1, January 2003, pp.76-80.