2016 4th Intl Conf on Applied Computing and Information Technology/3rd Intl Conf on Computational Science/Intelligence and Applied Informatics/1st Intl Conf on Big Data, Cloud Computing, Data Science & Engineering

Toward Obliging Bots for Supporting Next Actions

Akira Inokuchi*, Haruaki Tamada*, Hideaki Hata†, Masateru Tsunoda‡,
* Faculty of Computer Science and Engineering, Kyoto Sangyo University.
† Graduate School of Information Science, Nara Institute of Science and Technology.
‡ Department of Informatics, Kindai University.

***Abstract*—ChatOps is an industry practice of conversation-driven development using instant messaging to automate common development or monitoring tasks. It makes team members communicate with each other efficiently to work, learn and create together. We expect ChatOps to have intelligent agents, which can perform predictive operations based on users' activities, in the near future. To enable such intelligent agents, we worked on building prediction models of users' next actions. We collected action histories and their objectives and then applied sequential pattern mining to predict next actions. The result shows that some rules that meet some conditions could apply in 99% rate . Even if data do not meet the conditions, excluding rules in unnecessary actions improved the quality of the rules. Then, we analyzed generality of rules. To relate different actions as the same action based on an interpretation, the quality of the rules was improved.**

I. Introduction

ChatOps supports software development by performing operations by sending or receiving messages through a chat service. Recently, ChatOps is widely used in software development[1]. Developers use common chat service as a usual communication tool. Therefore, ChatOps reduce the learning cost to introduce a novel tool to perform a new task. ChatOps performs various operations. A bot is a computer program which performs various predetermined operations. The bot receives certain inputs and performs predetermined certain outputs. To performing operations, a bot is often active in a chat service. Figure 1 describes the role of a bot in general ChatOps. At first, (1) a user sends a command as a message to a bot on a chat service, and the bot receives the message. Then, (2) the bot performs operations based on the message. Finally, (3) the bot obtains the results and (4) sends it to a user. Moreover, every operation is logged, since operations are performed through a chat service.

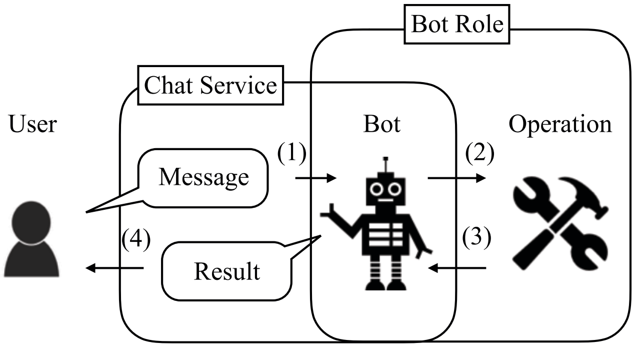

Fig. 1: A role of a bot in general ChatOps

ChatOps is introduced in various offices, however, operations through a bot on a chat service are mostly fixed[1]. Typical operations are deployment, reservation of meeting rooms, notification from a system, confirmation of a schedule and etc. However, ChatOps potentially has many possibilities. The paper enhances possibilities of ChatOps by collecting other data, for example, action histories of a user, in addition to chat logs.

Then, we tackle to recommend an action to a user or to perform instead of the user by predicting next actions from histories. We named our bot Kirin and describe the role of Kirin in Figure 1.

At first, (1) Kirin collects action histories from a user. Next, (2) Kirin predicts next actions of the user from the collected histories. Then, (3) Kirin performs operations based on the predicted actions and (4) sends the result to the user.

A typical scenario is that a user always performs storing source codes, then, compiling them. Kirin performs a compilation after detecting the store action of the user. That is, Kirin performs predicted operations before a user action.

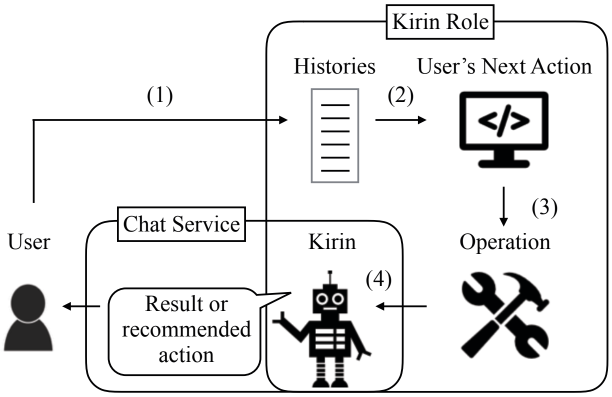

Fig. 2: A role of bot in Kirin

[1] ChatOps at GitHub —https://speakerdeck.com/jnewland/chatops-at-github

978-1-5090-4871-7/16 $31.00 © 2016 IEEE
DOI 10.1109/ACIT-CSII-BCD.2016.43

Therefore, the paper examines the prediction of next actions of a user. Furthermore, the paper examines whether we can predict next actions of a user by histories of a different user. If the prediction is well performed, we will use a lot of histories from many users for prediction.

The rest of the paper is organized as follows. Section II describes our method, then we examine our method in Section III and IV. Next, Section V shows related works. Finally, we conclude the paper in Section VI.

II. Proposed Method

A. Overview

To predict next actions of a user, rules that explain the order of occurrence actions are necessary. The goal of this paper is to obtain rules and to improve qualities of rules based on user's past actions. Therefore, we collect action histories of a user and analyze them. In the analysis, we extract several rules which contain an antecedent part and a consequent part. The antecedent part shows user's action combination in time series. Also, the consequent part represents the action that is performed after actions in the antecedent part.

We show the overview of the proposed method in Figure 3. At first, (1) we collect action histories and its objectives from a user. Next, (2) we prepare a dataset from histories and objectives. Finally, (3) we extract rules from the action histories.

A user performs various actions, even if we collect only actions on a PC. Typical examples of the actions are an operation with IDE, typing in word-processing software, switching applications and etc. On the other hand, the actions and their frequencies might be different following objectives of the user. For example, IDE is frequently used in development, however, not used for writing a manuscript. Contrary, some actions might be performed constantly, regardless of objectives. Therefore, we focus on both of the dependent actions and the independent actions from the objectives.

For collecting action histories of a user, we should consider what types of actions can be collected. When an action is conducted on a particular application, we must monitor the application. However, since actions across applications, we must monitor OS. In this paper, we collect actions on Terminal application and transitions between applications. In Terminal, we collect input command histories as actions of a user. Also, we collect transitions between applications as actions of the user by monitoring OS.

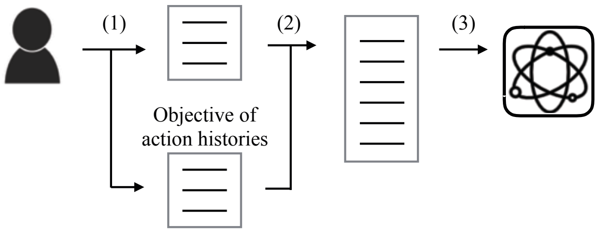

Fig. 3: Overview of proposed method for one user

The actions from both sources reflect behaviors of a user. Terminal is used skillfully by experienced developers since various commands are installed by default. Also, to achieve tasks generally requires to use several applications. Therefore, a user frequently switches applications to achieve their tasks.

We analyze collected action histories with SPM (sequential pattern mining) [2]. The algorithm of SPM extracts rules which show an action after a particular action sequence. For obtaining high-quality rules, we associate action histories and objectives and generate a new action histories.

B. Data Collection

This section gives collection method and its data. Collected action histories are command histories on Terminal and transitions between applications with date time. Command histories were collected by `history` command in bash[2] and zsh[3] (built-in command). Transitions between applications were collected by MALIMO[4] since the user environment was OS X. Besides, if the user environment was Windows, we can use Taskpit[5] or ManicTime[6] for collecting transitions. Then, we merge the all of the collected action data into the one file as a dataset with sorting by date time. Objectives of a user are collected by Toggl[7]. For our work, the user registers an objective to Toggl before starting a task.

C. Data Analysis

We analyze collected data to predict next actions after particular action sequence within a specified time. We employed the analysis method as SPM[2][8].

In this paper, we collect action histories $H = \{h_1, h_2, ..., h_n\}$ of a user. Each element of action histories is an action $h_i = \{t_i, C_i\}(1 \leq i \leq n)$ and shows an action of a user. t_i is date time of an action, and C_i means a content set of an action. Specification of C_i is given by a user. Then, we extract work histories $\mathcal{W} = \{W_1, W_2, ..., W_m\}$ by splitting H into each work. W_j means a work in the sequence of actions ($W_j = \{h_j, h_{j+1}, ..., h_{j+k-1}\}$). When an interval between h_j and h_{j+1} is greater than τ, we split the action histories H at h_j and h_{j+1} and generate $\mathcal{W}$, since we consider that the task was switched. An action h_i belongs to only one work W_j.

A rule set $R = \{r_1, r_2, ..., r_l\}$ is extracted from $\mathcal{W}$. A rule r_i represents in $\{a_i \rightarrow o_i\}(1 \leq i \leq l)$. a_i is called antecedent part and consists of $\{h_i, h_{i+1}, ..., h_{i+k}\}(1 \leq k \leq 4)$. Then, we call o_i consequent part, and o_i consists of an action h_o. Note that, each interval among elements of a_i and between h_{i+k} and h_o is less than δ. Some action may add between the elements of a_i and h_{i+k} and o_i, within satisfying the condition.

[2] https://www.gnu.org/software/bash/

[3] http://zsh.sourceforge.net/

[4] http://tamalab.bitbucket.org/malimo/

[5] http://taskpit.jpn.org/

[6] http://www.manictime.com/

[7] https://toggl.com/

[8] https://cran.r-project.org/web/packages/arulesSequences/index.html

Next, we define the evaluation metrics for extracted rules. The rules R^S are extracted from train data H^S. Then, we apply R^S to test data H^T and evaluate R^S. We develop three metrics in the following formula, rule counts C_R, occurrence probability of antecedent P_R and accuracy of the rule A_R.

$$\begin{aligned}
p(\mathcal{W}^T, r_i) &= \begin{cases} 1 & (a_i \textit{ is in } W_j^T (1 \leq j \leq m)) \\ 0 & (\textit{otherwise}) \end{cases} \\
s(\mathcal{W}^T, r_i) &= \begin{cases} 1 & (r_i \textit{ can be applied to } W_j^T (1 \leq j \leq m)) \\ 0 & (\textit{otherwise}) \end{cases} \\
C_R &= \text{count}(R^S) = |R^S| \\
P_R &= \text{probability}(\mathcal{W}^T, R^S) = \frac{\sum_{i=1}^{l} p(\mathcal{W}^T, r_i)}{|R^S|} \\
A_R &= \text{accuracy}(\mathcal{W}^T, R^S) = \frac{\sum_{i=1}^{l} \frac{s(\mathcal{W}^T, r_i)}{p(\mathcal{W}^T, r_i)}}{\sum_{i=1}^{l} p(\mathcal{W}^T, r_i)}
\end{aligned}$$

Moreover, we collect the objectives of actions ($P = \{p_1, p_2, ..., p_u\}$). An objective represents $p_x = \{s_x, e_x, \rho_x\} (1 \leq x \leq u)$. s_x shows start time, e_x means end time, and ρ_x is objective in natural language. We split the action histories H into objective action histories $H_{p_i} = \{h_j, h_{j+1}, ..., h_{j+k-1}\}$ by satisfying $s_i \leq t_j, ..., t_{j+k-1} \leq e_i$.

III. Experimental Evaluation

In this section, we investigate rules in the case that we extract rules from a user and apply it to the same user. We show the overview of the proposed method in Figure 4. At first, (1) we create a dataset from action histories and objectives of a user. Next, (2) we split the dataset into train data and test data. Then, (3) we extract rules of actions from the train data. Finally, (4) we evaluate the rules by applying the rules to the test data. Thus, we reveal a quality and a condition to be a higher quality of the rules.

A. Extraction Rules

The input data of this experiment is action histories in a week of a university professor. $|H|$ is 2,833, and the number of action kind is 181. We split collected data into train data H^S and test data H^T. H^S is former 80% of H, and H^T is rest. Furthermore, τ is 10 minutes, and δ is 15 seconds.

We extract the rules R^S from W^S and evaluate the result of applying R^S to W^T for two cases. The cases are using all of the input data W^S and each W_i^S by splitting W in objectives. We show the number of the actions and works of six each W_i^S in Table I. Seminar A (Graduate school), Career courses and Seminar B (Java) mean lectures and their preparations, study meeting means meeting with laboratory members and writing paper A and B mean writing and revising the papers.

Actions in Seminar A (Graduate School) of the user are preparation lecture materials with PowerPoint, developing assignments with Xcode and managing the version of these contents with git.

B. Evaluation of Rules

The result of the analysis shows in Table II. The first column represents objective of the actions. The following columns represent metrics showed in Section II-C. The row of W represents all of the action histories, and the row of W having an objective represents action histories excluded unknown objective. The following rows mean the result metrics of the applied rules based on the objective. The bar labeled `./hoge` command on Terminal means the user execute `hoge` file. The user confirmed and tested the result of execution since the user was developing `hoge` then. The table shows that A_R is greater than90% in using all data (W), W having an objective and career course. Therefore, we consider that the extracted rules have high accuracy. Especially, in the career course, the quality of the extracted rules is quite a high since A_R is 99%, and P_R is 91%. On the other hand, in writing paper A and B and seminar B (Java), we could not extract high-quality rules. The results are affected by a small amount of the actions ($|H|$). $|H|$ of career course is less than about 100 actions than Seminar A (Graduate school). Nevertheless, the accuracy of extracted rules from career course are better in all metrics C_R, P_R and A_R. Therefore, h_i in $H_{\text{Career course}}$ repeats similar actions. That is, $H^S_{\text{Career course}}$ and $H^T_{\text{Career course}}$ have similar actions. Hence, the result demonstrated that, when action histories are consistent, a quality of extracted rules increases.

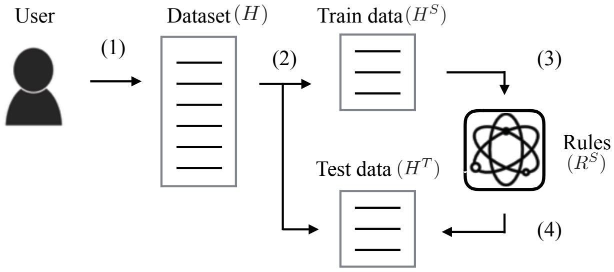

Fig. 4: Overview of proposed method

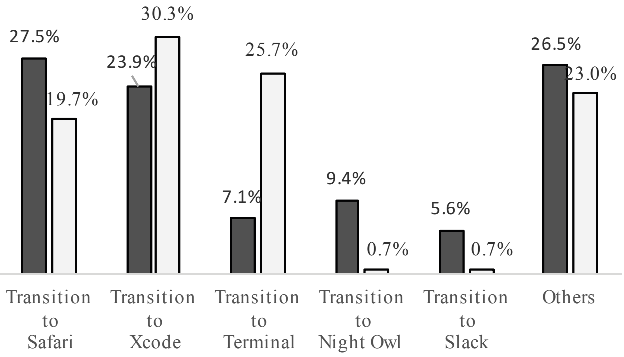

Fig. 5: Actions rate of Seminar A (Graduate school)

TABLE I: Metrics of the input data

	$\|H^S\|$	$\|H^T\|$	$\|H\|$	$\|W^S\|$	$\|W^T\|$	$\|W\|$
All data	2,266	567	2,833	64	24	87
Data with objectives	1,721	434	2,155	35	21	50
Seminar A (Graduate school)	607	152	759	9	1	9
Career course	501	126	627	5	1	5
Study meeting	297	75	372	10	8	17
Writing paper A	180	46	226	7	4	10
Writing paper B	72	18	90	2	3	4
Seminar B (Java)	64	17	81	2	4	5
Unknown objective			678			37

TABLE II: Metrics of the applied rules based on the objective

	C_R	P_R	A_R
W	395	41.5%	92.9%
W **having an objective**	391	67.0%	95.9%
Seminar A (Graduate school)	129	51.2%	87.4%
Career course	245	91.4%	99.1%
Study meeting	35	20.0%	37.5%
Writing paper A	8	12.5%	0.0%
Writing paper B	0	N/A	N/A
Seminar B (Java)	0	N/A	N/A

C. Detail Evaluation of the Rules

From Table II, $|R|$ of seminar A (Graduate school) is 129. The consequent parts of 52 rules include the transition to Night Owl (application of twitter client). The typical rule is to activate Night Owl, Xcode, then, Night Owl, again.

$$\{ \text{Night Owl, Xcode} \} \rightarrow \{ \text{Night Owl} \}$$

The transition is included 57 times in $H^S_{\text{seminar A (Graduate school)}}$. However, in $H^T_{\text{seminar A (Graduate school)}}$, the transition is included only 1 times. Therefore, $H^S_{\text{seminar A (Graduate school)}}$ and $H^T_{\text{seminar A (Graduate school)}}$ have the same objective, however, do not have same actions. For this reason, we exclude transition rule of Night Owl from $R^S_{\text{Seminar A (Graduate school)}}$, generate new rules and apply the new rules to $H^T_{\text{Seminar A (Graduate school)}}$. As a result, the quality of the rules is improved since P_R increases from 51% to 79%, and also A_R increases from 87% to 91%.

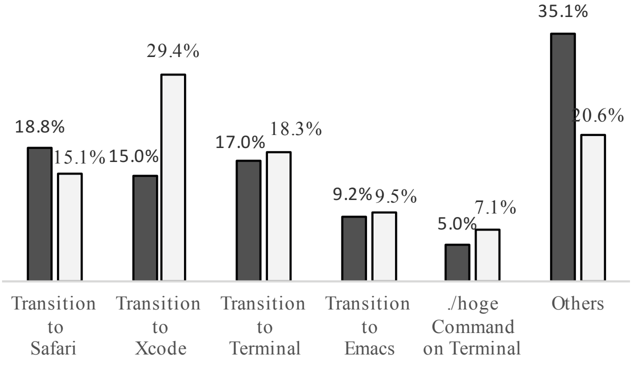

Fig. 6: Actions rate of Career courses

Hence, we found that, if action histories have the same objective, however, do not consist same contents of the actions, a quantity and a quality of the extracted rules are worse. To improve a quality of the rules, the action histories should consist the same objective and contents of actions.

D. Comparison of Train Data and Test Data

We examine the difference between H^S and H^T in seminar A (Graduate school) and career courses. Figure 5 and 6 show containing rate of the highest frequent five actions in H^S and H^T. The vertical axis shows the rate of frequencies, and the horizontal axis shows the actions by sorting frequencies. Additionally, the bar labeled other means the actions that not in top five frequencies. The black bar shows the rate of action in H^S, and the white one shows the rate in H^T.

From seminar A (Graduate school) in Figure 5, differences are greater than 5% between H^S and H^T except for transition to Xcode and PowerPoint. On the other hand, from career course in Figure 6, the difference is less than 5% except for transition Xcode and Finder. Therefore, in career course, there are no significant differences between H^S and H^T. Hence, the difference between H^S and H^T affects the quality of rules.

IV. Applying Rules for a Different User

A. Overview

In this experiment, we investigate the rules for applying to a different user. The procedure of the experiment is shown in Figure 7. The subjects of the experiment are two university students. We collect action histories and objectives from two subjects in two weeks. Then, we conduct the experiment with the collected as follows.

(1) We create a dataset from action histories H_A and H_B and objectives P_A and P_B of users A and B.
(2) We regard H_A as H^S and H_B as H^T (respectively denotes H^S_A and H^T_B).
(3) The rules R^S_A are extracted from the train data H^S_A of a user.
(4) We evaluate the rules R^S_A by applying the rules to the test data H^T_B.
(5) We swap the train data and the test data and perform from (2) to (4).

The common objectives from the subjects were making a poster and writing a paper. Since $|H|$ of making poster are small ($|H_A| = 50$, $|H_B| = 31$), we exclude the actions of making a poster. Therefore, we focus on the actions of writing

TABLE III: Metrics of input data and extracted rules R_A, and R_B

	Input data		Extracted rules		
	$\lvert H\rvert$	$\lvert W\rvert$	C_R	P_R	A_R
R_A	484	19	14	7.1%	0.0%
R_B	2,035	29	238	0.4%	11.1%

TABLE IV: Actions rates of H_A and H_B

H_A **of subject** A		H_B **of subject** B	
Actions	**Rate**	**Actions**	**Rate**
Transition to TeXShop	38.4%	Transition to Terminal	30.0%
Transition to Firefox	35.3%	Transition to Google Chrome	22.9%
Transition to Terminal	7.4%	Transition to Preview	11.5%
Transition to Preview	3.3%	Transition to TweetDeck	6.2%
Transition to Finder	2.5%	`exit` command on Terminal	4.6%
Others	13.1%	Others	24.8%

TABLE V: Related actions rates of H'_A and H'_B

H'_A **of subject** A		H'_B **of subject** B	
Actions	**Rate**	**Actions**	**Rate**
Transition to Terminal	45.8%	Transition to Terminal	30.0%
Transition to Browser	35.3%	Transition to Browser	29.1%
Transition to Preview	3.3%	Transition to Preview	11.5%
Transition to Finder	2.5%	`exit` command on Terminal	4.6%
Transition to Skype	1.9%	Transition to Keynote	3.3%
Others	11.2%	Others	21.5%

TABLE VI: Metrics of R'_A, and R'_B

	C_R	P_R	A_R
R'_A	12	66.7%	90.9%
R'_B	241	2.1%	86.7%

a paper. The main actions in writing a paper of two subjects are writing, referring documents and dictionary on the web and managing versions of their contents with git.

B. Metrics of the extracted rules

The result of the analysis and metrics of the input data are shown in Table III. The first and second columns show metrics of the input data. The following columns represent metrics showed in Section II-C. The columns of extracted rules are obtained by applying R_A^S to H_B^T (row of subject A) and R_B^S to H_A^T (row of subject B). From Table III, the maximum value of P_R is 7.1% and of A_R is 11.1%. The metrics show that the qualities of rules are very low. Therefore, even if users have the same objective, extracted rules could not share among users.

As well as Section III-D, we focus on the highest frequent five actions in H_A and H_B. The resultant actions of the subjects are shown in Table IV. From Table IV, subject A performed to transition to TeXShop and Firefox, however, subject B did not perform the actions. As well as, the transition to Google Chrome and TweetDeck are in subject B, however, not in subject A. Therefore, actions of subject A and B are quite different.

C. Interpretation of the Actions

Subject A uses Firefox, and subject B uses Google Chrome from Table IV. The applications are different, however, browsers. Therefore, we relate the transitions to Firefox and Google Chrome to the same action. Also, we relate other actions of both subjects. Finally, we regard the transitions to Firefox, Google Chrome, and TweetDeck as the transition to a browser and the transition to TexShop as the transition to Terminal.

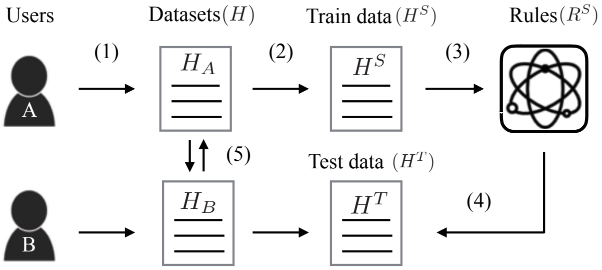

Fig. 7: Overview of proposed method for multiple users

Table V shows the resultant actions of relations H'_A and H'_B from subject A and B. From Table V, the transition of Terminal, Browser and Preview are within top three in the same order in actions of both subjects. While the difference is greater than 5%, the generality of actions between users is improved in comparison with before relations.

Then, we conduct the experiment shown in Section IV-A with changing the H_A to H'_A, and H_B to H'_B. Table VI shows the result. Since the maximum value of P_R is 66.7% and of A_R is 90.9% from Table VI, the qualities of the rules were improved. Relations between actions of users improve the generality of the rules. That is, improving a generality of rules requires an interpretation of the rules.

D. Limitation

We focused on top five frequent actions in Section IV-C. However, many actions still exist in the collected actions H_A and H_B. For example, on Terminal, `cd` command which changes the current directory are logged as a different command when arguments are specified in an absolute path and a relative path. If the commands were interpreted, we can regard them as the same command.

Also, in the Section IV-C, we related the transition to TeXShop as the transition to Terminal. However, the transition to Terminal has multiple purposes. For example, version control with `git` command is conducted on Terminal. Therefore, it is quite a tough task to understand the intention of the user from the action histories. If we could understand the intention of the user from the action histories, general rules might be extracted. Extracting more general and meaningful rules is left to our future work.

V. Related Works

Monden et al. tackled to improve a quality in personal software process (PSP) by collected actions[3]. PSP means improving the process by analyzing spent time for works (e.g., implementation, test) and revealing differences between estimation and measurement and causes of differences[4].

Since the paper also uses action histories of a user, the paper will contribute to the development of PSP.

Furthermore, to facilitate the measurement of actions, Monden et al. developed TaskPit and have improved it[5]. TaskPit measures transition of applications and date time as well as MALIMO.

Ohashi et al. estimated objectives of an action from collected action histories with random forest[6]. Differences of the paper are objective variable, explanatory variables and analysis method. We use data from their explanatory variables in addition to objectives and command history. And we predict next actions using sequential pattern mining.

VI. Conclusion

In this paper, we tackled obtaining rules and improving qualities of rules based on user's past actions. We collected action histories of users and their objectives. Then, we extract rules from collected actions by using sequential pattern mining Also, we define the metrics for evaluating the extracted rules. The extracted rules were evaluated by the metrics. The results were that probability of antecedent P_R is greater than 90%, and accuracy of the rule A_R is also greater than 99%.

Next, we investigate a generality of the rules. We conducted the experiment that rules from subject A were applied to different subject B. The result shows that probability and accuracy were quite low since $P_R = 7.1\%$, and $A_R = 11.1\%$. However, to relate actions as the same action based on an interpretation, the metric values were improved. The probability P_R was improved to 66.7% from 7.1%, and the accuracy A_R was improved to 90.9% from 11.1%. To evaluate the quality of interpretation is left to our future work.

References

[1] E. Sigler, "What is chatops? and how do i get started?" https://www.pagerduty.com/blog/what-is-chatops/, December 2014.

[2] F. Masseglia, M. Teisseire, and P. Poncelet, "Sequential pattern mining: A survey on issues and approaches," *Encyclopedia of Data Warehousing and Mining nformation Science Publishing*, pp. 1028–1032, 2005.

[3] A. Monden, Y. Kamei, H. Uwano, and K. Matsumoto, "A software development task measurement system for process improvement," in *15th Foundation of Software Engineering (FOSE2008)*, 2008, pp. 123–128.

[4] W. S. Humphrey, *Introduction to the Personal Software Process*. Addison-Wesley Professional, 1996.

[5] A. Monden, H. Uwano, K. Araki, K. Yamada, and K. Matsumoto, "Automatic measurement of software development tasks in software companies," in *20th Foundation of Software Engineering (FOSE2013)*, 2013, pp. 257–262.

[6] R. Ohashi, H. Uwano, A. Monden, K. Araki, K. Yamada, and K. Matsumoto, "Task purpose estimation in software development based on automatic measurement data and machine learning," *Computer Software*, vol. 33, no. 2, pp. 2_139–2_150, 2016, (In Japanese).