Mining System-User Interaction Logs for Interaction Patterns

Mohammad El-Ramly
Department of Computer Science,
University of Leicester, UK
mer1@le.ac.uk

Eleni Stroulia
Department of Computing Science,
University of Alberta, Canada.
stroulia@cs.ualberta.ca

Abstract

Many software systems collect or can be instrumented to collect data about how users use them. The type of data collected depends on the system. Other than logging purposes, such data can be used for different purposes. For example, sequential data mining can be applied to discover interesting patterns of user activities. We developed a process for discovering a special type of sequential patterns, called interaction patterns. These are sequences of events with randomly distributed noise, in the form of spurious user activities

We successfully applied our interaction pattern mining process in two contexts. First, we mined recorded traces of the system-user dialog with the user interface of a legacy system to recover the de-facto requirements of the legacy system and reengineer its user interface. Second, we mined Web server logs of a focused Web site, which is a site that is used consistently by the users to support an ongoing process, e.g., a university course Web site. The discovered interaction patterns of previous recent users can be used to generate Web page recommendation on-the-fly. Interaction pattern mining is promising and can be generalized to other software runtime repositories.

1. Introduction

Many systems can be instrumented to generate various types of runtime data while they are operating. These data rang from memory usage of a software system to alarms in a telecommunications system, etc. Dynamic behavior analysis is the process of reasoning over the runtime behavior of a system using such data. In its broad sense, dynamic analysis can be applied to a wide range of systems using a wide range of techniques. It can be offline, i.e., operating on recorded traces of the system's behavior, or online, i.e., operating on the data generated while the system is running. Dynamic behavior analysis methods are established and often used for program comprehension, debugging, reverse engineering, etc. [13].

We focus on dynamic analysis of software systems using one type of runtime data; that is sequential data. Sequential runtime data consists of temporal sequences of events that took place while the system is running. These data can be the sequences of procedure calls occurring during the execution of a procedural system, the entries of Web server logs, etc. We call such sequences "interaction traces". This type of data is common and usually includes some interesting patterns of the system behavior and user activities, especially if the data is directly related to the user's input. Finding these patterns helps understanding what tasks are of interest to the system user or alternatively what system services are frequently used by the users and how. Then, these patterns can be used for reengineering and improving the system-user interaction, program comprehension, etc.

We have developed a process for mining interaction traces and successfully used it to mine runtime data of software systems. In this paper we demonstrate two application of our "interaction pattern mining" process. First, we applied it to partially recover the software requirements of a legacy system from traces of the users' dialog with the system user interface (UI), recorded while the users were doing their regular jobs. The recovered requirements model the currently active and popular services of the legacy system as accessed through its UI interface, rather than the original specifications that led to system design, which are often lost or outdated for legacy systems. These service models are used for reengineering the legacy user interface and wrapping it with a Web or graphical user interface (GUI). Second, interaction pattern mining was used to discover the interesting navigation patterns in the Web server logs of a focused Web site. A focused Web site supports an ongoing evolving process and users use it in a consistent way, e.g., navigation activities of different users during a period of time are more or less similar. Examples of such sites are Web sites of university courses. They evolve according to the course schedule. Students access the Web site in similar ways depending on the course-work progress. The discovered pattern can be used for online URL recommendations for current users to make their navigation easier and faster, by suggesting to them the consequent pages accessed by enough recent users who had similar navigation history.

The rest of the paper is organized as follows: Section Two describes the "interaction pattern mining" problem. Sections Three and Four describe the two applications. Section Five is the summary and conclusions.

2. Interaction Pattern Mining

Mining sequences of data for recurring patterns is a generic well-studied problem with instances in a range of domains that are similar in that the data to be mined is represented by sequences, but different in the type of patterns of interest to the analysts in each domain. Examples of such problems are sequential pattern mining of data from the retail industry [1], "discovery of frequent episodes in event sequences" [11], "discovering patterns in DNA and protein sequences [4]. Available algorithms to solve these problems emerged from data mining [e.g., 1, 2, 3, 11] and bio-informatics [e.g., 8, 10] communities

Interaction pattern mining is similar to other sequential pattern mining problems in that the input data is ordered sequences of event Ids (or URLs or protein labels, etc.), but it is different in terms of the type of patterns sought. This is because interaction pattern mining constrains the maximum level of noise permitted in a pattern instance, but does not care where the noise occurs. This gives flexibility in discovering tasks that include user mistakes, trivial events and/or alternative paths for some subtasks.

Interaction pattern mining has a variety of applications in software engineering. For example, it can be used to analyze, understand and model user-browsing behavior on the Web, or interaction behavior between users and interactive applications. The objective in this family of applications is to discover coherent sequences of user actions that are likely to correspond to the accomplishment of a task. Such episodes can be used in several activities, e.g., application requirements recovery, user interface modeling and behavior prediction and recommendation.

To formally define interaction pattern mining,

1. Let A be the ***alphabet*** of events.
2. Let $S = \{s_1, s_2, \ldots, s_n\}$ be a set of sequences. Each ***sequence*** s_i is an ordered set of Ids drawn from A and it represents a recorded trace of runtime behavior of the system under analysis.
3. An ***episode*** e, is an ordered set of events occurring together in a given sequence.
4. A ***pattern*** p is an ordered set of events that exists in every episode $e \in E$, where E is a set of episodes of interest according to some user-defined criterion c. E and e are said to "support" p. The individual Ids in an episode e or a pattern p are referred to using square brackets, e.g., $e[1]$ is the first Id of e. Also, $|e|$ and $|p|$ are the number of items in e and p respectively.
5. If a set of episodes E supports a pattern p, then the first and last Ids in p must be the first and last Ids of any episode $e \in E$, respectively, and all Ids in p should exist in the same order in e, but e may contain extra Ids, i.e., $|p| \leq |e| \; \forall \; e \in E$. Formally,
   - $p[1] = e[1] \quad \forall \; e \in E$,
   - $p[|p|] = e[|e|] \quad \forall \; e \in E$, and
   - $\forall$ pair of positive integers (i, j), where $i \leq |p|$, $j \leq |p|$ and $i < j$, $\exists \; e[k] = p[i]$ and $e[l] = p[j]$ such that $k < l$.

   The above predicate defines the class of patterns that we are interested in, namely, ***approximate interaction patterns*** with at most a predefined number of insertion errors. For example, the episodes {2,***4***,3,4}, {2,***4***,3,***2***,4} and {2,3,4} support the pattern {2,3,4} with at most 2 insertions per episode, which are shown in bold italic font.
6. An ***exact interaction pattern*** q is a pattern supported by a set of episodes E such that none of its instances has insertion errors
   - $q[i] = e[i] \quad \forall \; e \in E$ and $1 \leq i \leq |q|$
7. The ***support*** of a pattern p, written as $support\ (p)$, is the number of episodes in S that support p.
8. **A *qualification criterion*** c, or simply ***criterion***, is a user defined quadruplet ($minLen$, $minSupp$, $maxError$, $minScore$). Given a pattern p, the ***minimum length*** $minLen$ is a threshold for $|p|$. The ***minimum support*** $minSupp$ is a threshold for $support\ (p)$. The ***maximum error*** $maxError$ is the maximum number of insertion errors allowed in any episode $e \in E$. This implies that $|e| \leq |p| + maxError \; \forall \; e \in E$. The ***minimum score*** $minScore$ is a threshold for the scoring function used to rank the discovered patterns. A scoring function can be defined depending on the application in hand and the nature of the patterns sought.
9. A ***maximal pattern*** is a pattern that is not a sub-pattern of any other pattern with the same support.
10. A ***qualified pattern*** is a pattern that meets the user-defined criterion, c.

Given the above definitions, the problem of interaction pattern mining can be formulated as follows:

Given:

(a) an alphabet *A*,

(b) a set of sequences *S*, and

(c) a user criterion *c*

Find all the qualified maximal patterns in *S*.

Solving interaction pattern mining problems is a three steps process. The first is a pre-processing step, needed for cleansing the data in the input sequences, depending on the problem in hand. In the second step, our novel algorithm for interaction pattern mining, Interaction Pattern Miner 2 (IPM2) [7], is applied to the cleansed sequences to discover the patterns that meet a user-defined criterion. The last step is the post processing analysis and comprehension of the discovered patterns. The first and third steps are application and data dependent.

3. Application 1: Requirements Recovery from Legacy System-User Dialog Traces

We applied interaction pattern mining to recover the software requirements of interactive legacy systems in the form of frequent usage scenarios of the system. These scenarios are in fact patterns of interaction between the legacy system UI and its users. Such patterns are buried in the huge amount of data exchanged in dialog between the users and the system via its UI. If this dialog is recorded, it can be mined for these patterns. This can be done to comprehend and/or reverse engineer the legacy system. It is particularly useful if comprehending or reveres engineering the legacy code is expensive or impossible.

In particular, for lightweight wrapping of the legacy system with a Web or graphical user interface, or lightweight integration of the legacy system user interface with other systems, it is imperative to understand how the system is actually currently used. In other words, to ensure that the desired services of the legacy system are properly wrapped or integrated, it is important to model these services. The interaction patterns representing these services can be enriched with additional semantic information and used for reengineering the legacy user interface, re-documenting the legacy system, creating a help system, or other migration and reengineering tasks

We applied interaction pattern mining for traces of interaction with legacy systems user interfaces as part of CelLEST project [5,9,12] for semi-automated legacy system user interface reengineering. In this project sub-task, we used a host-access middleware that serves as an emulator and recorder to access legacy IBM 3270 systems. Through the middleware, legacy system users can open a session with the legacy application and do their regular jobs. The middleware records their dialog with the system UI in the form of sequences of legacy screen snapshots forwarded to the user' terminal, interleaved with the user actions in response, in the form of keystrokes. Screens and snapshots are like classes and objects; the later are instances of the former. Analysis methods, including feature extraction, clustering and others are used to derive a model of each legacy screen from all its instances in the traces. This model includes, for example, any distinguishing features of the screen like a title or a certain visual distribution of the screen content that occurs on all its recorded instances. In this step, each screen is given a unique Id. Hence, the interaction traces can be abstracted by sequences of Ids as shown in Figure 1, which shows part of a real interaction trace taken from navigating a legacy library catalog system.

To explain what type of interaction patterns can be found in these traces, Figure 1 shows two very similar segments of the user dialog with the legacy library system (in the dashed polygons) that occurred apart from each other in the trace. These segments suggest that the user was doing two similar runs of the same task, or alternatively, s/he was using the same legacy system service twice, although the two runs differ in the number of instances of screens 6 and 9 accessed. In the actual recorded trace, screen 4 displays the results of issuing a *browse* command to browse the relevant part of the library catalog file. Then, the user decides which items s/he wanted to retrieve from the catalog by issuing a *retrieve* command and s/he receives screen 5. Then, s/he displays brief information about the items using *display* command that displays screen 6. Finally, s/he selects an item using the *display item* command to display its full or partial information (screen 7). After selecting an option from screen 7 (e.g., full details, summary, etc.), screens 8, 9 and 10 display the first, intermediate and last pages of the required details, respectively. The number of instances of screens 6 and 9 retrieved varies depending on the item checked. The navigation segment of Figure 1 shows that this task of item information retrieval was done twice.

We have applied interaction pattern mining to recorded system-user dialog traces of a number of systems [5]. In one experiment, we collected 5 traces of user interaction with the IBM 3270 connection of public library system. Each trace represented a session of interaction between a user and the Library catalog, during which s/he retrieved information about the library items of interest. The traces had 1657 screen snapshots in total and 27 screens (recall the classes and object analogy). The segment of Figure 1 is taken from one of the traces after identifying screens and labelling snapshots with their screen Ids. The traces were pre- processed and analyzed using IPM2 algorithm. Then manually, an analyst post-processed the discovered patterns by filtering out trivial patterns. Full details of this experiment are in [5]. In this experiment, three patterns of interaction with the library catalog were

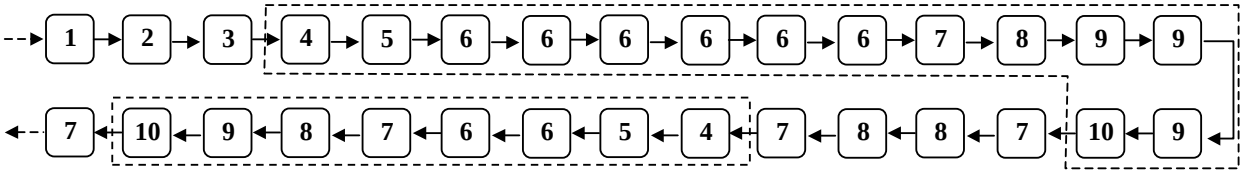

Figure 1. A Segment of a Trace of Interaction with a Legacy Library System

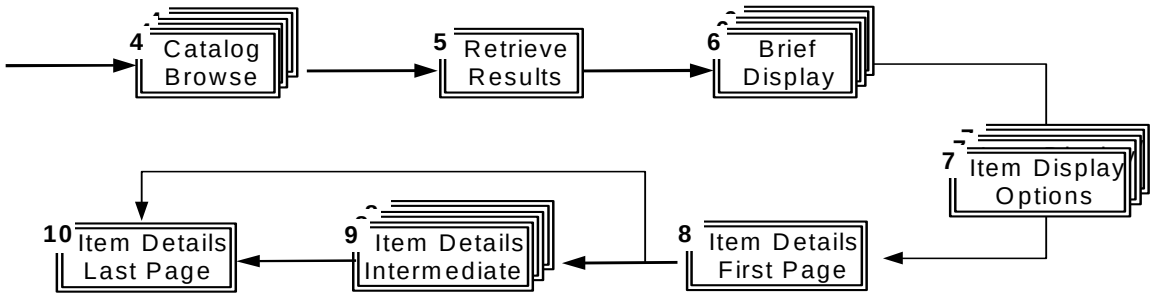

Figure 2. A Diagrammatic Representation of The Pattern 4^{+}-5-6^{+}-7^{+}-8^{+}-9^{*}-10, Corresponding to The Information Retrieval Task Repeated Twice in The Trace Segment of Figure 1.

discovered; each represented an information retrieval task, corresponding to a service of the legacy system. Figure 2 shows one of the interaction patterns discovered, which is {4^{+}-5-6^{+}-7^{+}-8^{+}-9^{*}-10}. n^{+} means one or more instance of screen n and n^{*} means zero or more. The two dashed polygons of Figure 1 represent instances of this pattern.

In CelLEST project for lightweight user interface reengineering, each pattern was enriched with semantic information and then translated to abstract user interface specifications that are executable on multiple platforms, e.g., XHTML-enabled (Extensible Hypertext Markup Language) platforms and WAP (Wireless Application Protocol) devices [9]. Hence, a quick Web wrapper can be generated semi-automatically for the legacy services of interest. Additionally, the enriched patterns can be represented in alternative formats to serve other purposes. For example, to integrate the legacy system services with new object oriented applications designed using UML, it could be useful to represent the recovered legacy system requirements as use cases. Figure 3 shows a use case representation of the pattern of Figure 2. These patterns can also be used for other re-documentation, program comprehension and requirement recovery tasks. .

Use Case name: Retrieving Information on a Library Item
Participating actor: Library System User
Entry condition: The user issues a *browse* command
Flow of events:

1- Flip the catalog pages until the relevant page.
2- Issue a *retrieve* command to construct a results-set for the chosen catalog entry.
3- Display the results set using *display* command and turn its pages until the required item is found.
4- Issue a *display item* command.
5- Specify a display option.
6- Display the item details.

Exit condition: The user retrieves the required Information about the item s/he wants.

Figure 3. A Use Case Model Representation of The Interaction Pattern of Figure 2.

4. Application 2: Analysis of Web-usage Behavior for Focused Web Sites

We applied our interaction pattern mining process for Web-usage behavior analysis of focused Web sites. A focused web site supports an ongoing evolving process and is usually navigated in a task-driven as opposed to data-driven way. The users of the Web site usually navigate it to accomplish the same task(s) during a certain period of time, and as the process evolves, they shift together to other tasks. Models of these tasks, in the form of interaction patterns, are buried in the Web server logs.

We used a Web site of a computer science university course as an example of focused Web sites. The users of this site usually do similar tasks related to their course work every week, e.g., read and/or download new materials, lab instructions and assignments and related code, etc. Ideally, the discovered interaction patterns correspond to the frequent user tasks of interest, not just to interesting associations of Web pages. In other words, the sequence of navigation matters. Since interaction pattern mining tolerates noise, in the form of insertion errors, it can discover patterns of sequential user navigation with spurious navigation or slight differences. These patterns can serve as basis for online URL recommendations for current users to make their navigation easier and faster, by suggesting to them the further pages accessed by recent users who had similar navigation sequences. Full details of this application are in [6], but highlights are briefed below.

Mining Web logs for interaction patterns is a three steps process. First, preprocessing cleans and standardizes the Web server logs and divides them into sessions. A session is a coherent sequence of Web site navigation activities of the same user. Then, IPM2 is applied to the sequences of URL Ids representing sessions, with a user defined interestingness criterion.

The extracted patterns can be used in a variety of ways, which decide what post-processing is needed. First, they can be used as means of understanding the navigation behavior of the Web site users. For example, the existence

of many long patterns might imply usability problems: in principle, users of focused Web sites are focused too and want to be able to accomplish their tasks in shorter rather than longer visits. Alternatively, patterns can be used as the basis for generating runtime recommendations for pages that new users of the Web site may want to visit.

A runtime infrastructure is required to enable the later use, capable of user session tracking and relevant pattern selection. The HTTP protocol is stateless and does not support establishing a long-term connection between the Web serve and the client's browser. To address this problem, dynamic page rewriting with hidden fields can be used. When the user first submits a request, the server returns the requested page rewritten to include a hidden field with a session-specific Id. Each subsequent request of the user to the server supplies this Id to the server, enabling it to maintain the user's navigation history. This session-tracking method does not require any information on the client side and can therefore be always employed, independent of any user-defined browser settings. Since the server knows the user's Id, it can examine its recent navigation history to identify whether it includes the prefix of any of the collected patterns. If so, then the suffixes of the relevant patterns may be offered as recommendations for subsequent navigation. Then, page-rewriting technique can easily support the dynamic adaptation of the pages requested by the users with the recommendations on new potential places to visit.

5. Summary and Conclusions

This paper presented the problem of interaction pattern mining and two applications of it. This problem is in fact an instance of using sequential pattern mining to perform offline dynamic analysis of sequential data collected during the runtime of a software or information system. These sequences are called interaction traces, and the types of pattern sought in them are called interaction patterns. An Interaction pattern represents a usage scenario of a system service on one hand and a frequently performed user task on the other hand.

We presented a formal description of the problem and two applications of it. The first is recovering the functional requirements of a legacy software system from recorded traces of the system-user dialog through the legacy UI. The recovered specifications are then used for reengineering and migration activities. We also have applied interaction reengineering to discover frequent user navigation patterns from server logs of focused web sites. Then, these patterns can be used for lightweight Web site runtime reengineering by introducing on-the-fly URL recommendations.

Interaction pattern mining is powerful technique for dynamic behavior analysis and can extended to different types of software runtime repositories to discover patterns of user and/or system activities.

References

[1] R. Agrawal and R. Srikant, Mining Sequential Patterns. In Proc. of the 11th Int. Conf. on Data Engineering (ICDE), pg. 3-14, 1995.

[2] R. Agrawal and R. Srikant, Mining Sequential Patterns: Generalizations and Performance Improvements. In Proc. of the 5th Int. Conf. on Extending Database Technology (EDBT), 1996.

[3] J. Baixeries, G. Casas and J. Balcazar, Frequent Sets, Sequences, and Taxonomies: New, Efficient Algorithmic Proposals. Report Num. LSI-00-78-R, El departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Spain, 2000.

[4] B. Brejova, C. DiMarco, T. Vinar, S. R. Hidalgo, G. Holguin, and C. Patten, Finding Patterns in Biological Sequences. Unpublished project report for CS798G, University of Waterloo, Fall 2000.

[5] M. El-Ramly, Reverse Engineering Legacy User Interfaces Using Interaction Traces. Ph.D. Thesis, University of Alberta, Canada, 2003.

[6] M. El-Ramly and E. Stroulia, Analysis of Web-Usage Behavior for Focused Web Sites: A Case Study. J. of Software Maintenance and Evolution: Research and Practice, vol.16, no. 1-2, pg. 129-150, 2004.

[7] M. El-Ramly, E. Stroulia and P. Sorenson, Interaction-Pattern Mining: Extracting Usage Scenarios from Run-time Behavior Traces. In Proc. of the 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD 2002), 2002.

[8] I. Jonassen, Methods for Finding Motifs in Sets of Related Biosequences. Dr. Scient Thesis, Dept. of Informatics, Univ. of Bergen, 1996.

[9] R. Kapoor, Device-Retargetable User Interface Reengineering Using XML. Tech. Report TR01-11, Dept. of Computing Science, Univ. of Alberta, 2001.

[10] A. Floratos, Pattern Discovery in Biology: Theory and Applications. Ph.D. Thesis, Dept. of Computer Science, New York Univ., 1999.

[11] H. Mannila, H. Toivonen and A. Verkamo, Discovery of Frequent Episodes in Event Sequences. Data Mining and Knowledge Discovery, vol.1, no. 3, pg. 259-289, 1997.

[12] E. Stroulia, M. El-Ramly, P. Iglinski and P. Sorenson, User Interface Reverse Engineering in Support of Interface Migration to the Web. Automated Software Engineering, vol.3, no. 10, pg. 271-301, 2003.

[13] Workshop on Dynamic Analysis (WODA), 2003.