Sequence Pre-processing: Focusing Analysis of Log Event Data

Alper Sarikaya, Emanuel Zgraggen, Rob DeLine, Steven Drucker, and Danyel Fisher

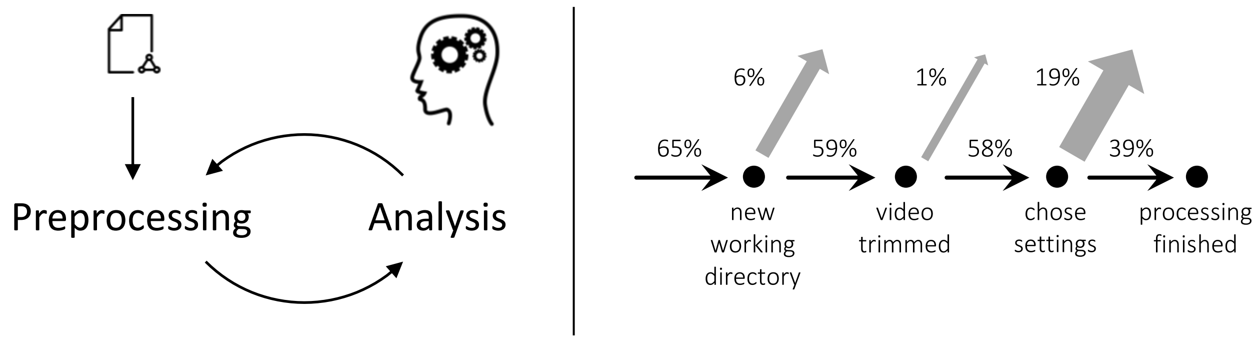

Fig. 1. On the left, an abstraction of our proposed pre-processing layer, and how it fits into the log data analysis workflow. Pre-processing helps to prepare log event data for downstream analysis by minimizing unnecessary noise during analysis. On the right, pre-processing allows for selecting and cleaning those user sessions in the data that fail to successfully save a Hyperlapse video (Sec. 5.1.1). This allows downstream analysis to see where people tend to exit the linear workflow.

Abstract—Many computational systems are generating log event data as a way to help developers understand the usage of applications in the wild. While many commercial analysis tools exist, they tend to treat log event data as a “bag of events” instead of collections of observed sequences, where each sequence represents an individual session. While recent work can support the visual analysis of event sequence data, log files tend to contain complexity in scale and noise that can foul downstream analyses. In this work, we identify common recurring problems of noise that arise from the analysis of this data, and assert that methods for pre-processing can be a valuable tool to both focus data for downstream analysis and provide provenance support for visual analytics tools. These pre-processing methods can be performed interactively and in conjunction with analysis tools to iteratively refine rules to streamline visual analysis. Through several case studies, we identify the common sources of noise in log files and demonstrate how our proposed pre-processing methods can help to minimize excess data reaching downstream analysis tools.

Index Terms—Event sequence analysis, log analysis, data cleaning

1 Introduction

Log event data are becoming increasingly prevalent in analysis scenarios. These logs can collect application usage by clients (via telemetry), communications by distributed systems (through many logging streams), medical treatments and outcomes (through electronic health records), and even tracking by wearable devices (personal analytics). While log event data exist in a wide variety of domains, it can be difficult to analyze them to understand aggregate client or system behavior. For example, in what order do users navigate through an e-commerce website before they complete a purchase? Many current commercial analytic systems for log analysis (see Sec. 4) do not treat logged events as parts of a larger sequence, instead simplifying them into bags of events. This, in turn, makes it difficult to aggregate behavior over many similar sets of linked events. A lack of a common lexicon for this type of analysis can hinder iterative, extensible improvement of analysis in this space. Through this work, we provide a discussion of the analysis tasks in this area, and suggest methods for directing interactive analysis of log data.

There are a set of tasks that an analyst may want to perform with log data. These tasks generally deal with user-centric or workflow-centric analyses, and eventually evolve to comparisons between sets of users or workflows. Common to all of these analyses, log data has a particular set of operations that can focus the downstream analysis of the data. Through this work, we show that different pre-processing methods prime different types of downstream analyses, guiding different avenues of exploration of this event data.

Log file analysis is a subset of the domain of exploratory event sequence analysis, which in turn is supported by a variety of visual analytic tools such as DecisionFlow [4], EventFlow [9], and (s|qu)eries [16]. Such tools have demonstrated their utility in surfacing insights from data exploration, such as errors in execution or highlighting potential causality of conditions of interest. However, many of these tools lack provenance support for focusing downstream analysis and sharing insights with colleagues. A typology of pre-processing methods can help to streamline analysis of log event data, and provide a common lexicon for data manipulation. We emphasize that such methods to pre-process data for focusing analysis can be a valuable mechanism for persisting analyst state across multiple analysis sessions, either to share with a colleague or to retain context when restarting the analysis to minimize clerical work.

In this work, we use several case studies to create a common typology of log data, and use this definition to identify tasks and issues in analyzing this data. We bootstrap this exploration through discussions with stakeholders and their desires to extract insights from the data. We identify recurring troubles in both visual and manual analysis of log data, and suggest standardized methods for focusing downstream analyses. We show how pre-processing methods can make analysis quicker, more efficient, and potentially more expressive through several case studies, and suggest potential applications of provenance into existing tools.

- *Alper Sarikaya is with the University of Wisconsin-Madison. Email: sarikaya@cs.wisc.edu.*
- *Emanuel Zgraggen is with Brown University. Email: ez@cs.brown.edu.*
- *Rob DeLine, Steven Drucker, and Danyel Fisher are with Microsoft Research. Email: {Rob.DeLine,sdrucker,danyelf}@microsoft.com.*

Proceedings of the IEEE VIS 2016 Workshop on Temporal & Sequential Event Analysis.
Available online at: http://eventevent.github.io

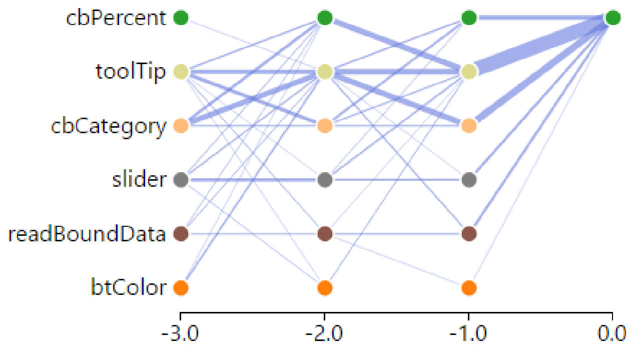

Fig. 2. An example of the noise that frequently occurs in log event data. A flow diagram shows the history of three events leading up to a “toggle Percent checkbox” (cbPercent) event, relative to the occurrence of the target event. With increasing numbers of event types, this sort of visualization rapidly becomes unusable due to noise.

2 Tasks in Log Event Data

This work concentrates on the domain of log event data, which are structured as a sequence of timed events. Log event data tend to share a basic structure: each entry in the log contains (at a minimum) a timestamp, the event name, and (optionally) other associated attributes for the particular event or session. A key component of log data is the session identifier that allows downstream analysis to reconstruct the sequence of events within each session. This reconstruction is a critical component of effective log file analysis as it permits more expressive queries of the data to be constructed with regard to their temporal order. Typically, a logged event is situated in a single point in time, though an event pair (e.g., “start”, “stop”) can define a time interval.

We introduce a model dataset to help define the tasks common in log file analysis. Microsoft Research has released several applications that bring more powerful data visualization tools to Microsoft Excel through the Office Add-In store, organized under the “Visualization Toolkit” umbrella, called *Aguvue* internally [11]. These tools include a (2D) histogram, treemap, and streamgraph visualizations. Anonymous telemetry is collected that notes the sequence of actions a user takes when using the application within Excel, such as recording which buttons were clicked, colors selected, and the size of the dataset visualized. The development team is interested in how people are using these applications—are users generally satisfied with the results of the created visualizations? If there are errors, are there common paths that generally result in an error condition? Is there an overrepresented sequence of actions that lead to an abandonment of the application?

These sort of queries of the data highlight the power of treating log file data as event sequences. Through related research and our case studies, we identify several buckets of tasks performed with log data:

- What sequences of events lead to (or follow) an event of interest?
- Which attributes of what events are potential indicators for a particular event occurring?
- What are the distribution of attributes from an event of interest?
- How does the temporal nature affect whether an event of interest is reached (or not)?
- What is the measure of “drop-out” that occurs in the sequence of events leading up (or following) an event of interest?

Certainly, there are similar analysis scenarios in sequence domains. In these tools and commercial products (see Section 4 for a discussion), there is a fair amount of work required to prime the data for analysis. While all sequence data tends to have noise and cleaning of this data before analysis has become a standard task in sequence analysis, log file data generally tends to have very standardized, regular noise that can be categorized. The genesis of this noise comes from the relatively cheap cost of logging events in a software setting—it is cheap in development time and storage cost to log events thought not to be useful initially, but could assist in debugging and analysis in the near future, a common trend in the era of big data analysis [13]. For this reason, stakeholders tend to log as much as possible as they cannot understand what is missing before issues arise [1].

The tasks for log file analysis all involve discovering and identifying motifs from noisy streams of sequence data. Noise in the data, such as extraneous events or attributes, has the potential to mask insights that would in turn hinder visual analysis. In the following subsection, we identify several motifs of noise in log event data that can foul downstream visual analyses.

2.1 Noise in Log Event Data

An example of noise in the *Aguvue* dataset could be a “slider” event, which is fired whenever a user moves a slider in the user interface. While many of these events appear in the data, generally just the final slider value is important to consider—the sequence of “slider” events can be collapsed into a single event, taking on the attribute value of the last appearing “slider” event.

From this model dataset, log files encountered in our use cases (Section 5.1), and discussions with stakeholders (*cf.* Barik, *et al.* [1]), we synthesize a generalization of common noise patterns in log data that can create difficulties in downstream analysis tools.

Repeated event — An event may be repeated many times, and the number of times it appears may not be important. Downstream analysis will be cluttered with these events that may not be important to the analysis story.

Useless event — An event may be occurring that the analyst knows to have no bearing on the current analysis task, or is interleaved from orthogonal operations. By not filtering these events, downstream analyses could mark this event as a false positive for causality. An example of this noise can be seen in Figure 2, where spurious *toolTip* events make analysis of the event sequence needlessly cluttered.

Ambiguous event — Depending on the history of events and their attributes leading to a particular event, a specific instantiation of an event may have alternate representations. Downstream analyses risk mistreating ambiguous events as noisy, hiding structure in the data.

Irrelevant sessions — An analyst may concentrate on the sessions that touched a specific feature, had a specific error message, or completed the expected sequence of actions. Given this constraint, all downstream analysis must qualify its queries to operate only over the sessions of interest.

Negation — As the dual of the previous problem, an analyst may want to understand why a successful condition was *not* met. From previous work of query construction for sequence data by Monroe, *et al.* [10], it is often difficult for an analyst to visually represent these negated queries in visual analytics systems.

Subsequences — For some analysis scenarios, it may be insufficient to treat session identifiers as the sole method for organizing events in ordered sequences. For example, an analysis may concentrate on the events that occur within a particular workflow, which may be repeated within an individual session.

Independent of the feature set of the downstream visual analytics tool used to perform sequential analysis of log data, we see a need to handle these patterns of noise to simplify and focus visual analysis.

3 Focusing Analysis Through Pre-Processing

From these common types of noise in log data, we define several pre-processing methods to simplify the data, and note how these methods can help analysis in these more focused contexts. Although we do not claim that this list enumerates all tools in the pre-processing toolbox for log event data, we have found that the following methods can be composed to encode powerful statements that are difficult to express in existing backend systems such as SQL. The decision for when and how to use these tools is governed by the analyst’s domain knowledge about the system and insights gained from previous explorations. By composing these methods appropriately, particular types of analysis

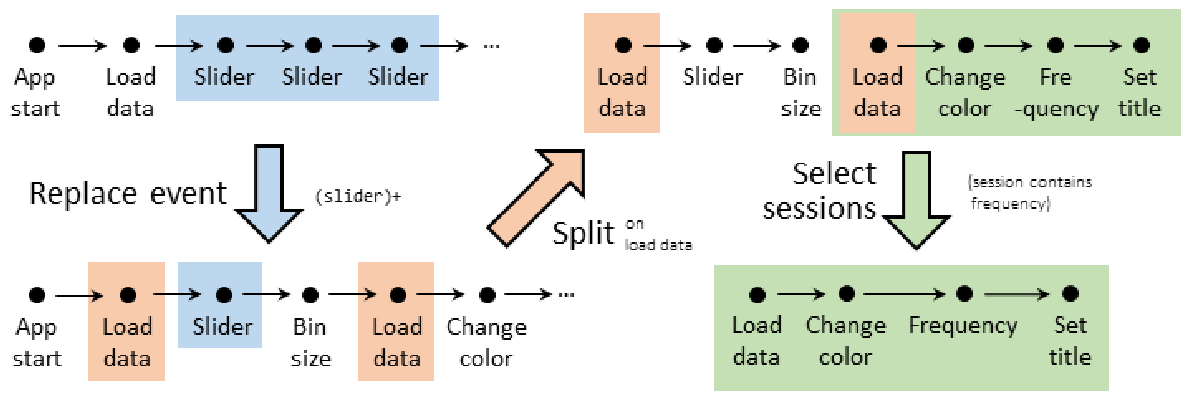

Fig. 3. An example composition of pre-processing rules for an exemplar session of log events, using data from the Histogram Excel Add-in (Sec. 5.1.2). A set of repeated slider events is *replaced* by a single slider event, then the session is *split* when new data is loaded. Finally, only sessions containing a frequency event are *selected* for downstream analysis.

tasks can be supported. Though we describe these methods in the abstract here, we have created a prototype system to implement these methods using functional reactive programming (Sec. 5).

Removing events — The useless event problem is mitigated through removing events. We have found that this is a common method for sifting through log event data—either the analyst knows that the event is irrelevant to the current thread of analysis, or discovers through exploration that a particular event is overrepresented in all analysis. To evaluate the efficacy of this method, the total number of occurrences removed or the percentage of sessions modified can be reported.

Replace with a surrogate event — By replacing a matched event or a sequence of events with a surrogate, this method attacks the repeated and ambiguous event problems. This action allows the analyst to collapse a well-known sequence of events to a single event; for example, if a sequence of seven different events identifies a successful application start, they could be replaced by a "successful start" event.

This operation can help to consolidate events based on semantic meaning, or even generalize over-specific events; for example, a set of undo events that have different names (e.g., "undoDraw", "undoSketch") could be represented by surrogate events with a common name ("undo") with an added attribute denoting the specific type of action. Similar to removing events, feedback to the analyst would denote how many events or sequences were matched by the rule, as well as the percentage of sessions.

Select sessions for analysis — Filtering the data to concentrate on a particular scenario is a common task in log data analysis, and it supports resolving the issue of irrelevant sessions. Sessions can be selected based on a wide variety of criteria, including inclusion (or exclusion) of a particular event, sequence of events, attribute value, or adherence to a temporal constraint.

A key benefit of this method is that it implicitly defines the set of "everything else" (the negation issue), making meaningful comparisons possible. Supporting the visual comparison between multiple sets of sequences has been the focus of some visual analytics systems, such as the work by Malik, *et al.* for cohort comparison [8]. The feedback to the analyst for this method can note how many surrogate events were added through this rule, as well as in what proportion of sessions.

Re-sessionize sequences — Exploding or splitting a session into smaller sessions can provide a more accurate representation for analysis scenarios that focus on a particular sequence of events. As an example, if the current thread of analysis focuses on a workflow that may occur many times within a session, it may be advantageous to explode the workflow into multiple sub-sessions. This expansion can be accomplished by matching a particular set of starting and ending events (e.g., user entered and left a special state), a long enough temporal pause in the user session (e.g., user was idle), or splitting a session on a particular event (e.g., user closed the current project). Feedback from this operation would identify the number of sub-sessions generated, along with the number of sessions matching the rule.

We note that these methods can be composited and applied in any order, see Figure 3 for an example instantiation. The interface that supports these rules should allow users to back out of a rule if they are unsatisfied with its effects, ideally presenting the order in which the rules were applied in a breadcrumb-like format. By supporting these methods, we anticipate that visual analytics tools that support log event data can use the iterative refinement of pre-processing rules within an analysis to foster serendipity in the discovery of insights. If the set of rules themselves are recorded and annotated, they can provide a powerful provenance mechanism for reuse between analysis sessions and enable collaboration in a visual analytics environment.

4 Related Work

As noted in the introduction, log event data is closely tied to many other types of event sequence data. Many visual analytics tools have been designed and evaluated on electronic health records (such as DecisionFlow [4] and EventFlow [9]), and log file analysis [16]. As part of the analysis, many of these tools prompt the viewer to explicitly specify or query relevant properties of the data. Cleaning or de-noising of the data generally falls outside of the scope of these tools, although the EventFlow tool supports many of these operations [9]. Though these cleaning steps are noted as part of the analytical workflow, the utility and reuse of this clerical work has yet to be utilized as analytical provenance, such as to share and collaborate in an analytical environment. Heer, *et al.*'s work on graphical histories [5] shows the visual analytics potential for event stream simplification to create more semantically-resonant views of a user's design process.

There has been long-evolving work in the database literature to support the rapid querying of event sequence data. Many methods treat queries over these sequences as automata (*cf.* Gehani, *et al.* [3]), where query languages extend the traditional language of SQL to deal with more temporal-specific analysis contexts. Systems such as SASE [15] and Trill [2] support methods such as the pre-processing methods discussed within this work to re-emit de-noised data for analysis. Though such tools support targeted analysis, their support for exploration is minimal and lack many human-centered advantages of the interactive interface of a visual analytics tool.

There are several commercial offerings that support the analysis of log event data, including Google Analytics, Splunk, Microsoft StreamInsights, and MixPanel. A majority of the visualizations in these analytics platforms are simple distributions of events, and require the analyst to curate events or short sequences of interest. In particular, MixPanel's "Funnel" focuses on the workflow of users and the rate of drop-out, but requires the manual curation of such funnels.

There have also been several studies that have looked at how big data analysis has been performed in the enterprise. Kandogan and co-authors [6] find that integration issues and a lack of a unified analytics language can hamper collaboration in analysis and notes that visualizations are rarely used as part of analysis. Barik, *et al.* [1] find that analytical tools for exploring log and telemetry data in the enterprise are lacking, and technological methods of collaborating on analyses are meager, identifying a need for expansion into this space.

Through the work presented herein, we provide characterization for log event data in order to support more targeted exploration. Log event data tends to share many qualities with "big data" analysis, and stands to immediately benefit from standardized methods for focusing the analysis and exploration [13].

5 Discussion

Using these methods of pre-processing to focus log event data for log event analysis, we can minimize the clerical work required to clean data for analysis and provide downstream visual analysis with more semantically-resonant events. This focused data enables the use of event-driven functional reactive programming [14], supported by many visual analytics platforms.

5.1 Case Studies

These case studies primarily look at telemetry logs of various applications, where anonymized user actions are recorded and logged for the purpose of post-hoc analysis by the development team. We instrumented the pre-processing techniques ad-hoc for each case study using the Trill temporal query engine [2].

5.1.1 Hyperlapse Pro Telemetry

Hyperlapse Pro is a publically-available Windows application developed at Microsoft Research to allow amateur videographers to create motion-steadied time-lapse videos [7]. Opt-in anonymous telemetry of use of the application is collected by the research team, who are curious about how users are using the Hyperlapse application in practice. Are people generally satisfied with the results? What sort of options do people select from processing? What leads users to save a generated Hyperlapse video?

Although it is difficult to ascertain if users are satisfied with the results of the application, a save event occurring after a video processed event implies that a user is satisfied enough to decide to save the processed video. We employed the third pre-processing technique of selecting sessions based on the criteria that a "Save project" event eventually follows a "Processing video" event. Based on this rule, we obtain two sets of sessions: those that saved a processed video, and those that did not. We chose to look at the event distribution within the sessions between these two sets, and found that a set of eight events tended to be overrepresented in the sessions that saved the video. Sensing a linkage between these events, we extended the comparison to trigram sets of sequences, and found these overrepresented events tended to reference one another in sequence, implying a wizard-like interface. In fact, the application is architected like a wizard, as shown in Figure 1. We discovered this structure purely by analyzing the logs.

Looking at the event sequences within this focused group, we observe that many user sessions dropped out between picking settings and completing the Hyperlapse processing. To understand why, we can amend our pre-process rule to separate out those sessions that completed processing and those that did not, and use downstream analysis tools to see if any factors separate the two groups. In this data, however, we could not find any distinct sequence of actions or attributes that differentiated these two groups.

5.1.2 Data Visualization Excel Add-Ins

As noted in Section 2, one of our log file sources are from the *Aguvue* dataset. Many potential analysis questions are of a sequential nature: if a user runs into an error condition, are they able to eventually make a visualization? What differentiates users who successfully bootstrap themselves out of error conditions from those that quit?

Similar to the pre-processing diagram in Figure 3, many repetitive slider events were found in downstream analysis. Adding a replace rule collapsed all slider events into a single, representative slider event. A tooltip event that was fired whenever the user hovered over an encoded shape was also removed from the analysis. To tackle the question about error bootstrapping, we selected those sessions that contained an error event—from that selection, we selected those sessions that subsequently generated a successful visualization.

6 Conclusion

This work begins to create the foundation for the next iteration of visual analytics tools for log event analysis. Through the composition of pre-processing rules, downstream analysis can benefit from focused data, and collections of rules can be used as provenance to bolster similar analyses in different contexts or to share with collaborators.

Though we have described the interaction and feedback for methods for pre-processing log event data, we have yet to realize a generalized prototype that works on many different types of log files. In the near future, we hope to realize the generalized application of these techniques through a user interface, with comparisons visualized by the system to guide the analyst to create the appropriate pre-processing rules for their analysis task. We also hope to validate our approach by comparing against existing approaches, particularly in the utility of pre-processing rules as provenance and the iteration between pre-processing the data and exploring the resulting data.

Acknowledgments

Acknowledgments omitted for review.

References

[1] T. Barik, et al. The bones of the system: A case study of logging and telemetry. In *Proc. Conf. Software Engr. (ICSE)*, 92–101. ACM, 2016.

[2] B. Chandramouli, et al. Trill: A high-performance incremental query processor for diverse analytics. In *Proc. VLDB Endow.*, 8(4), 401–412, 2014.

[3] N. H. Gehani, H. V. Jagadish, O. Shmueli. Composite event specification in active databases: Model & implementation. In *Proc. VLDB*, 327–337. VLDB Endowment, 1992.

[4] D. Gotz and H. Stavropoulos. DecisionFlow: Visual analytics for high-dimensional temporal event sequence data. *IEEE Trans. Vis. Comput. Graphics*, 20(12), 1783–1792, 2014.

[5] J. Heer, et al. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *IEEE Trans. Vis. Comput. Graphics*, 14(6), 1189–1196, 2008.

[6] E. Kandogan, et al. From data to insight: Work practices of analysts in the enterprise. *IEEE Comput. Graph. Appl. Mag.*., 34(5), 42–50, 2014.

[7] J. Kopf, M. F. Cohen, and R. Szeliski. First-person hyper-lapse videos. *ACM Transactions on Graphics*, 33(4), 78:1–7:10, July 2014.

[8] S. Malik, et al. Cohort comparison of event sequences with balanced integration of visual analytics and statistics. In *Proc. Conf. Intelligent User Interfaces (IUI)*, 38–49. ACM, 2015.

[9] M. Monroe, et al. Temporal event sequence simplification. *IEEE Trans. Vis. Comput. Graphics*, 19(12), 2227–2236, 2013.

[10] M. Monroe, et al. The challenges of specifying intervals and absences in temporal queries: A graphical language approach. In *Proc. Conf. on Human Factors in Computing Systems (CHI)*, 2349–2358. ACM, 2013.

[11] A. Perer, et al., "The Event Event: IEEE VIS 2016 Workshop on Temporal & Sequential Event Analysis," http://eventevent.github.io. 2016.

[12] A. Rind, et al. Interactive information visualization to explore and query electronic health records. *Foundations and Trends in Human Computer Interaction*, 5(3), 207–298, 2011.

[13] B. Shneiderman and C. Plaisant. Sharpening analytic focus to cope with big data volume and variety. *IEEE Comput. Graph. Appl. Mag.*, 35(3), 10–14, 2015.

[14] Z. Wan, W. Taha, and P. Hudak. Event-driven FRP. In *Proc. PADL*, 155–172. Springer Berlin Heidelberg, 2002.

[15] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *Proc. SIGMOD*, 10(1), 407–418. ACM, 2006.

[16] E. Zgraggen, et al. (s|qu)eries: Visual regular expressions for querying and exploring event sequences. In *Proc. Conf. on Human Factors in Computing Systems (CHI)*, 2683–2692. ACM, 2015.