LogVis: Graph-Assisted Visual Analysis of Event Logs from Industrial Equipment*

Tugba Kulahcioglu[1], Dmitriy Fradkin[1]✉, Ayse Parlak[1], and Alexander Belkov[2]

[1] Siemens Corporation, Princeton NJ, USA
[2] Siemens AG, Nuernberg, DE
first.last@siemens.com

Abstract. Visual reasoning on a graph is often a challenging task mainly due to the vast number of nodes and edges displayed. It becomes particularly challenging on log graph data, where thousands of events may be logged within minutes. In this study, we focus on three common log analysis tasks, namely *Event Overview*, *Root-Cause Analysis* and *Pattern Analysis*, and propose visualization approaches to overcome challenges particularly associated with these tasks. The proposed approaches are demonstrated on sample use-cases on industrial equipment logs.

Keywords: Knowledge Graph · Log Analysis · User Interfaces

1 Introduction

Optimization of operation and maintenance of industrial machines/equipment can lead to significant benefits for the operators and service organizations. Traditionally, a domain expert manually reviews the equipment logs for troubleshooting problems and understanding patterns leading to faults. The logs are typically a combination of structured (e.g. time, component) and unstructured (e.g. log text) data. A single machine can easily produce thousands of log entries in minutes, resulting in a semi-structured large dataset and making manual log analysis challenging and laborious.

In this paper, we propose graph-based visualization approaches that facilitates otherwise cumbersome log analysis tasks. The first task we discuss is *Event Overview*, during which domain experts review the behavior of the equipment and acknowledge *errors* or *critical events*. The following task is to carry out *Root-Cause Analysis* to understand the sequence of events that led to the *critical event* that is being analyzed. Finally, the domain experts look for further evidence in the dataset to verify that the discovered sequence of events typically leads to the critical event. This is achieved through *Pattern Analysis*. We propose the following visualization approaches to help facilitate the aforementioned tasks:

- A *Template-based Group View* for log messages that reduces the number of nodes visualized from (tens of) thousands to tens/hundreds while keeping a very large portion of the semantic information needed to successfully carry out the *event overview* task

- A *Timeline View*, to aid in the *Root-cause analysis*, where the nodes are aligned based on their temporal relationships, and with further encodings we ensure a quick understanding of the whole temporal picture of the visualized log messages.
- A *Visual Pattern Search* method which facilitates *Pattern Analysis* by leveraging the power of visualization to quickly observe similarities and differences between message sequences.

The paper is structured as follows. Section 2 discusses related work. Section 3 describes knowledge graph construction from log data. Section 4 describes our approach and Section 5 describes the impact of the tool, and the lessons learned. We conclude in Section 6.

2 Related Work

Knowledge graphs have been adopted to cope with the challenges of log datasets in [5],[6]. These methods, however, mostly focus on extracting some domain-specific concepts from data, such as detection of an IP address in a log message text using predefined regular expressions.

Visualization is a frequently used approach to summarize and organize large log data. Similar to our study, [4] leverages *event templates* to extract valuable information from log messages, and applies data mining methods to make inferences. They provide an event summarization approach which uses inter-arrival histograms in order to capture the temporal relationships between the events. Same study also visualizes log data using parallel-coordinates and scatter-plots. A recent study [1] proposes a log processing model that generates a natural language report using storytelling techniques for cyber threat intelligence purposes.

3 Log-Graph Construction

A graph G consists of a set of nodes N and relationships R. We construct a graph from the log data by representing each log message with a node $n \in N$. Additional entities that log messages are associated with (e.g. customer, machine id, message category) are also represented as nodes in the graph. The nodes are linked by edges corresponding the relationships $r \in R$, ex. each log message is linked to machine it occurs on, which in turn is linked to a customer. Temporal relationships between the messages (*prev* representing previous message and *next* representing next message) are also represented with a relationship $r \in R$. Each log message node is also associated with a *severity* attribute (e.g. *info*, *warning* and *error*), which are used to color-code nodes (*green*, *yellow*, and *red*, respectively) in the graph visualization.

4 Graph-Assisted Visual Log Analysis

In this section, we present our approach to graph-assisted visual log analysis, addressing the typical tasks carried out on equipment log data. We do not focus on implementation

details in order to keep the discussion general and applicable to multiple domains and systems. Our specific implementation uses Neo4[3] backend, and the UI is created using JavaScript, in particular by adapting libraries neovis[4] and vis-network[5].

4.1 Event Overview

We describe the task of reviewing and summarizing log events, associated challenges and our approach to overcoming these challenges. We exemplify the impact of our approach on a real-world use-case.

Task Description. Domain experts often need to review event log data that can span long time periods. Several higher level tasks could require such analysis, for example reporting equipment behavior for a specific time period or analyzing critical events for maintenance purposes.

Challenges. Typically, such analysis involve thousands of logs. As the visualization community is well aware, displaying very large graphs can create viewability and usability issues [3]. In the case of log analysis, although the users could be supported with additional statistical analysis, the understanding of different relationships between the nodes remains an issue. A solution for this problem in large graph visualization is to cluster the nodes (which we will refer to as *grouping*) based on a selected property that carries meaningful information for the task to be achieved.

Our Approach: Template-based Group View. We propose applying node grouping using *Event Templates*. A *template* represents the *fixed part* of a log message, which is shared among all log messages that are produced by the same specific lines in the code.

Consider two messages: "Error occurred reading file Input12.txt from server A" and "Error occurred reading file Input14.txt from server B". They share the fixed parts "Error occurred reading file" and "from server", while the file and server names appear to be case-specific. A template for these messages would look like this:

Error occurred reading file <filename> from server <servername>

We use DRAIN method [2] to automatically extract template set T from our data. For each log message node $n \in N$, we add relation (n, t) to R where t represents the template of the log message n, and $t \in T$. For a summary view that would facilitate event overview, for each $t \in T$, we merge all nodes n where $(n, t) \in R$ into a supernode. Figure 1 shows group view for the graph on the left. Sizes of the supernodes are determined based on the count of individual nodes merged to the supernode. Similarly, edge thickness reflects the number of connections of the individual nodes from the supernode. Tooltips over nodes and edges show the counts. We use dashed lines for the connections of supernodes to reflect the *virtual* nature of these nodes/connections.

[3] https://neo4j.com/

[4] https://github.com/neo4j-contrib/neovis.js/

[5] https://github.com/visjs/vis-network

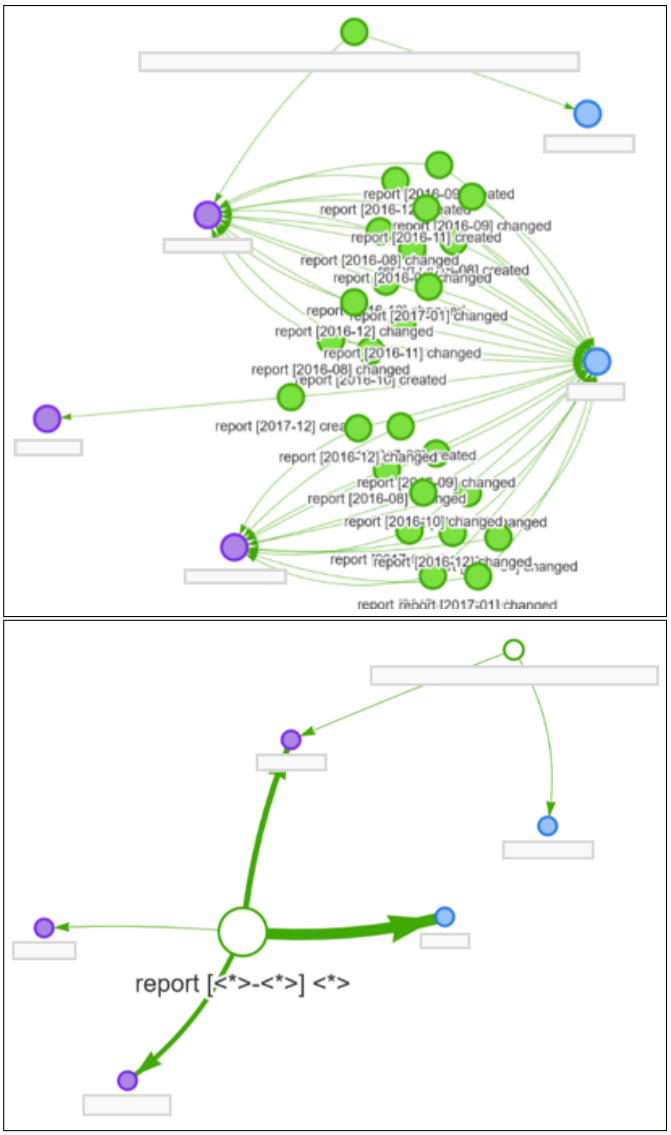

Fig. 1. A sample graph (on top), and its *Group View* (on bottom).

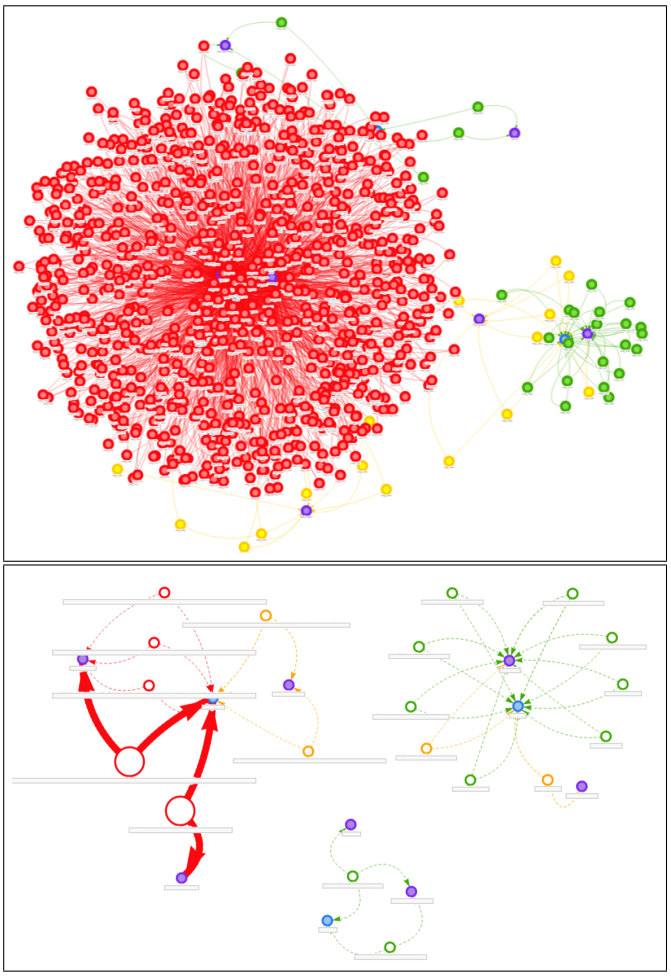

Fig. 2. Case study for the *Group View*, demonstrating the summarization provided by template-based supernodes in the bottom figure.

A larger scale example of the proposed *group view* is provided in Figure 2 on the right. Hundreds of red nodes representing error messages are merged into only five supernodes, two of which seem to cover majority of the messages. This summarization was possible because all these messages shared five unique templates.

4.2 Root-Cause Analysis

The structure of this section is similar to that of Section 4.1. We analyze challenges of Root-Cause Analysis task, present our temporal analysis approach to overcome these challenges, and demonstrate it on a real-world use-case.

Task Description. One of the primary purposes of equipment log analysis is to be able to locate and analyze equipment issues. When a problem on the equipment occurs, domain experts look through the logs to pinpoint it and then analyze preceding time period logs to find the root-cause of the issue and to develop a potential fix.

Challenges. Just like the previous task, this one also suffers from extensive amounts of logs. Grouping in this case might not be the best solution since specifics of the individual messages could carry important information about the error and its causes. Experts want to be able to filter out the logs that are unrelated, and need to see temporal relations between nodes. As a result of this analysis, experts come up with potential root causes, which may need to be verified by support from similar cases from previous such issues of the same or similar equipment.

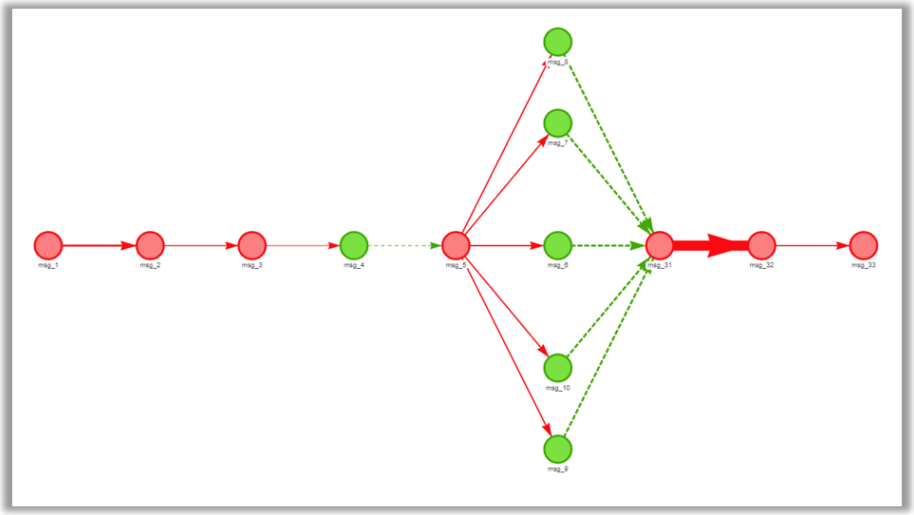

Fig. 3. A close-up of *Timeline View*.

Our Approach: Timeline Analysis. To overcome the challenges of root-cause analysis, we propose a timeline analysis method that allows the experts to focus on a subset of logs and their temporal relationships. As the first step, we developed a *Timeline View* that supports temporal analysis of the log-graph via the following:

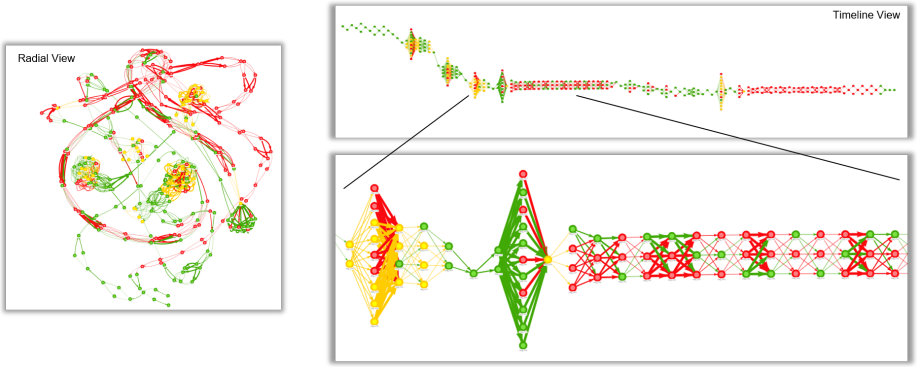

Fig. 4. Case study for Timeline Analysis: the left plot shows default radial layout which is rather confusing. Our proposed Timeline View is on the right.

- *Layout and Node Alignment.* The nodes are aligned from left-to-right based on the specified relationship (e.g. ”next message” relationship). As an example, in Figure 3, msg 32 is connected to msg 33 with a ”next message” relationship, hence they are aligned horizontally from left-to-right. It shows that msg 32 happens before msg 33 in the timeline. The nodes that have the same timestamp are aligned vertically, indicating their shared temporal order (e.g. the green nodes in Figure 3).
- *Virtual Edges.* Since the experts filter out unrelated nodes, most of the time the visualization would include several disconnected subgraphs. if the nodes are not necessarily directly connect with the specified relationship (in our case ”next message” relationship). To enable the user to see the temporal relationship between these disconnected graphs, we join them using virtual relationships that are created on the fly. We use ”dashed” lines for the virtual relationships to distinguish them from regular ones. In Figure 3, the green nodes that are parallel are connected to msg 31 with virtual relationships, showing that the currently displayed portion of the data doesn't include the messages in between, and that the events associated with messages 6 to 10 occurs before message 31.
- *Edge Thickness.* The thickness of the edges between the nodes (in our case ”log messages”), indicate the time-difference between neighbor log messages. As the distance between the log messages increase, the edge gets thinner, indicating a weaker relationship. In Figure 3, the edge between msg 31 and msg 32 is thicker than other edges, which shows that the distance between these messages are less than the others, hence the potential of these messages to be a part of a pattern is stronger.

– We note that Timeline View is separate from Group View. In fact it would be difficult to combine them since node aggregation would interfere with Time-based arrangement of nodes. Thus, Timeline view does not have supernodes. The use of edge thickness here is different from how it is used in Group View where edge thickness reflects the frequency of connections of the supernode. It is possible to include other types of nodes (ex. Category or Actor) in the Timeline view, but that leads to additional non-temporal edges.

Figure 4 shows, on a larger example, how the timeline view facilitates the root-cause analysis. On the left part of the figure, the graph is visualized using the default radial view which is a frequently used graph visualization technique, also employed by our *Group View* presented in the previous section. On the top-right is a our *Timeline View*, which aligns the nodes based on temporal relationships. The bottom-right figure shows a zoomed version of the part of the graph[6]. The *Timeline View* makes it clear which messages precede a specific error message, hinting at potential causes of the error.

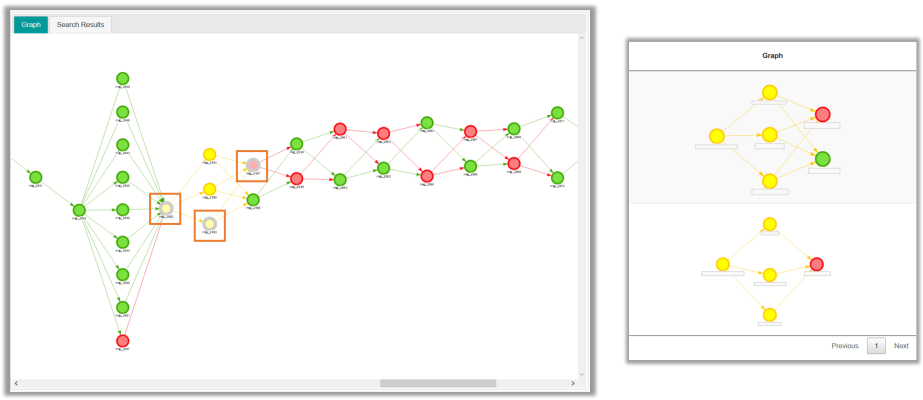

Fig. 5. Visual pattern search: User selects nodes (on the left). Relations between them are included automatically. The search results are shown on the right.

4.3 Pattern Analysis.

We analyze the task of *Pattern Analysis* on log data, and present our approach to overcome the challenges typically associated with it.

Task Description. When a problem occurs, domain experts examine the logs surrounding the time of the problem to try identifying its root cause. The *Pattern Analysis* task aims to find out whether there is a diagnostic or predictive pattern for the problem that repeats across time and other devices. Based on the outcome of this analysis, proactive actions can be taken to prevent similar errors in the future.

[6] All our visualizations support zooming in and out.

Challenges. Domain experts can use their knowledge and the logs at specific failure to come up with candidate patterns[7]. However, they need to search data across machines and time to be sure that the pattern is useful. This requires ability to specify the desired pattern and to easily visualize and review search results.

Our Approach: Visual Pattern Search. To support the users with the *Pattern Analysis*, we developed a visual pattern search capability where users can select consequent messages from the displayed graph, and search the database for other occurrences of the same pattern. The search can be carried out using the original log message texts, or message templates. Figure 5 shows a sample visual pattern search (on the left) and its results (on the right).

Pattern search results are displayed using a table. Each row presents an occurrence of the input pattern, together with related information, including start and end timestamps of the retrieved message sequence (hidden in the Figure for customer anonymization). Each search result is visualized using the "Timeline View' introduced in Section 4.2. This allows the users to quickly understand the specifics of each result, and to compare multiple results quickly, uncovering potential patterns in the data. In the sample results from Figure 5, three consequent input messages form the input, as seen in the left panel. Based on the results, the second message seem to occur with a specific set of other warning messages (yellow nodes) in both cases that are found in the data. Figure 6 presents search results for two selected consequent messages, and the results seem to contain several distinct groups, unlike the single pattern observed in the example from Figure 5.

5 Impact and Lessons Learned

The approaches presented in this work were implemented by the authors and evaluated by domain experts, specifically service technicians and data analysts from the business unit. The implementation consisted of several phases, each following an Agile process. Informal discussions with domain experts were used to identify desirable functionality and to collect feedback. Their feedback and feature requests enabled us to compile the following impact and lessons learned:

– Domain experts found the use of automatically mined templates immediately useful for their overview analysis. However, one of their requests, which we incorporated, was to generalize some of the templates while specializing a few others. Generalizing a template essentially is creating a variable from a static part of a template, which ends up combining several templates into a single one. Specializing a template is the reverse operation, in which a dynamic part of a template is changed to static, resulting potentially in multiple templates in place of a single one. These modifications allow the domain experts to see more meaningful log groups in the overview analysis. Out of a few hundred templates initially created, these adjustments included less than 10 templates. This suggest that while automated template

[7] By a pattern we mean a set of log messages and relations between them. A sequence matches a pattern if it is a superset of pattern's messages and relations.

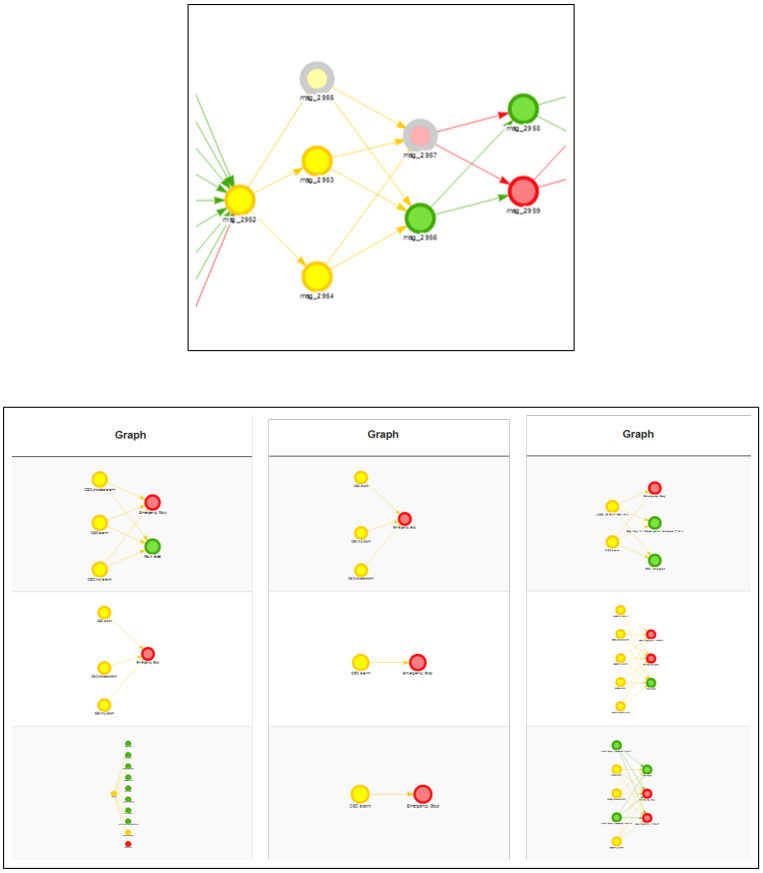

Fig. 6. Visual pattern search: User selects nodes (on top). Relations between them are included automatically. The search results are shown on the bottom.

extraction can provide significant benefits, it is useful to have domain experts to review the set of templates and provide any adjustments they find helpful.
- In the *Timeline View*, the *edge thickness* and *virtual edges* features were developed in response to domain experts' need to visually grasp the time difference and the ordering information between messages.
- Visual pattern search was found to be very useful to discover frequent patterns in the data, and to detect outlier behavior. There are cases where the input pattern matches hundreds or even more message sequences. To handle such cases, domain experts suggested having an additional column in the results table showing the number of nodes in the found sequence, to be able to cluster similar sequences together by sorting the table based on that column. Another suggestion was for functionality that would allow use of the search results to filter the dataset, giving the experts a chance to further analyze a result (e.g. expanding the timeframe to see more messages before and after the search result).
- Graph-assisted visualization made possible fast benchmarking or comparison of devices, based on distributions of different types of messages made possible by the *Group View*.
- Overall, use of the tool led to significant reduction in time needed by service technicians for root cause analysis from days to hours, while leading to more robust outcomes due to checking of discovered patterns with Pattern Search over long time periods and multiple machines and customers. The functionality of the tool also provided users not only with insights into root cause analysis but also into customer operations such as problems with the environment or incorrect use/configuration of the equipment.

6 Conclusion

Visual log graph analysis suffers from the vast amount of nodes and the lack of temporal perspective in the visualizations. In this paper, we proposed visualization approaches specific to log data that can help handle the aforementioned issues. Our template-based group view allows quick analysis of large log graphs, whereas timeline view and pattern search facilitate discovery and validation of temporal patterns. We also presented feedback from domain experts that should prove helpful in design of similar tools.

References

1. Afzaliseresht, N., Miao, Y., Michalska, S., Liu, Q., Wang, H.: From logs to stories: Human-centred data mining for cyber threat intelligence. IEEE Access **8**, 19089–19099 (2020)
2. He, P., Zhu, J., Zheng, Z., Lyu, M.R.: Drain: An online log parsing approach with fixed depth tree. In: 2017 IEEE International Conference on Web Services (ICWS). pp. 33–40. IEEE (2017)
3. Herman, I., Melançon, G., Marshall, M.S.: Graph visualization and navigation in information visualization: A survey. IEEE Transactions on visualization and computer graphics **6**(1), 24–43 (2000)

4. Li, T., Jiang, Y., Zeng, C., Xia, B., Liu, Z., Zhou, W., Zhu, X., Wang, W., Zhang, L., Wu, J., et al.: Flap: An end-to-end event log analysis platform for system management. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1547–1556 (2017)
5. do Nascimento, C.H., Ferraz, F.S., Assad, R.E., e Silva, D.L., da Rocha, V.H.: Ontolog: Using web semantic and ontology for security log analysis. In: The Sixth International Conference on Software Engineering Advances (2011)
6. Nimbalkar, P., Mulwad, V., Puranik, N., Joshi, A., Finin, T.: Semantic interpretation of structured log files. In: 2016 IEEE 17th International Conference on Information Reuse and Integration (IRI). pp. 549–555. IEEE (2016)