Report

My main job is to reproduce the paper, **Sequence Synopsis**, which is from VAST 2017, mainly about event sequence data visualization.

Github: https://github.com/bebinca/ss_vast17

Background

This part is mainly about the background of this paper and some important concept.

Event Sequence Data

An **event** can be understood as who did what at what time. So arranging the events in the order of time constitutes an **event sequence**. For example, as this picture shows, in the error log of a vehicle, the actions or failure sorted in the time order is an event sequence.

missing message
short circuit
GPS lost
circuit malfunction
temperature low
battery low
Defect

Website click streams, **user interaction logs in software applications**, electronic health records (EHRs) in medical care and vehicle error logs in automotive industry can all be modeled as event sequences.

It is crucial to reason about and derive insights from such data for effective decision making in these domains. For example, by analyzing users' interaction log with software applications, usability issues and user behavior patterns can be identified to inform better designs of the interface.

Previous Methods

With the growing importance of event sequence analysis, a variety of visualization techniques have been proposed in the past years.

For example, the **Sequential pattern mining (SPM) algorithms** will generate a long list of potentially redundant patterns, but analysts have to rely on certain metrics to prune or rank them, which may result in partial coverage of the data and leave out some insights [1, 2]. Visualization techniques based on **sequence clustering** can also give an overview of data, but the algorithms are difficult to interpret [3, 4].

Therefore, it still remains a challenging task to create intuitive, simple, yet comprehensive overviews for real-world event sequence data.

Idea

In this paper, the author propose a new approach of event sequence visualization. The **goal** is to construct an overview of the data with a good balance between the simplicity of the visual representation and its information content.

Their main idea Can be summarized in two aspects, **algorithm** and **visual analytics system**. Use algorithm to find the similar parts of the events sequences, which is called sequential patterns, can describe the overview data information. And then build a visual analytics system to supports level-of-detail exploration.

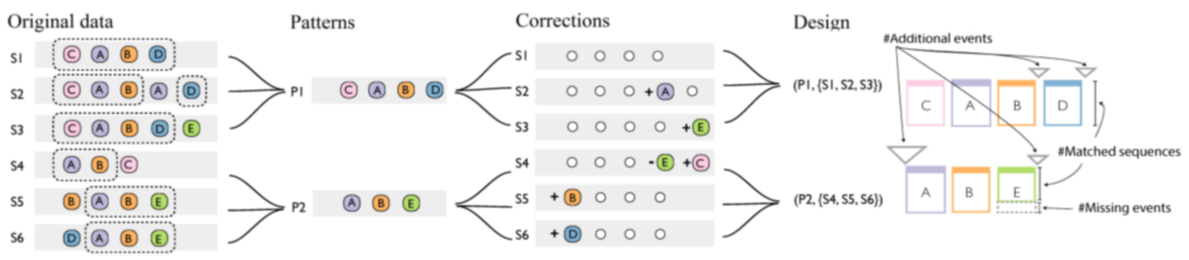
The algorithm will be mainly talked about in model part, and the visual analytics system will be introduced after that.

Model

Method Overview

The model is based on a two-part representation of the data which consists of a set of sequential patterns and a set of corrections. The original sequences are mapped to the patterns, and the corrections part specifies the edits needed to transform the patterns to the individual sequences.

For example, you can see from this picture, the original data S1 S2 S3 all contain the pattern CABD. And this pattern can be recovered to S2 by inserting A.



MDL Principle

The author design an algorithm to find the two part representation, following the **Minimum Description Length (MDL) Principle**, which states that the best model for a dataset results in the minimized description length**.**

As SPM algorithm in the previous work part, what's the difference between MDL and SPM? The characteristic of SPM is that as long as the frequency of occurrence of a certain substring is higher than a certain threshold, it is recognized as a pattern. So the pattern repetition rate may be very high, and there is a problem of pattern explosion.

MDL is to find a pattern set, this set can describe the sequence set well as a whole, so it can often summarize the sequence set well with a small number of patterns, avoiding the drawbacks of SPM.

Denotations and Formal Problem Definition

Now we start to introduce the denotations and formally define the description length of the two-part representation.

An event sequence is an ordered list of events $S = [e_1, e_2, \dots, e_n]$ where $e_i \in \Omega$, an event alphabet. Given a set of event sequences $\mathscr{S} = S_1, S_2, \dots, S_m$, the goal is to identify a set of patterns $\mathscr{P} = P|P = [e_1, e_2, \dots, e_l]$ and a mapping $f : \mathscr{S} \rightarrow \mathscr{P}$ from the event sequences to the patterns that can minimize the total description length. The author makes several definition:

- A pattern can be described by simply listing the events in it.
- An edit can be fully specified by the position and the event involved, and its description length can be roughly treated as a constant.

Therefore we get this equation:

$$L(\mathscr{P}, f) = \Sigma_{P \in \mathscr{P}} len(P) + \alpha \Sigma_{S \in \mathscr{S}} ||edits(S, f(S))|| + \lambda ||\mathscr{P}||$$

In this equation, $len(P)$ is the number of events in the pattern, which stands for the description length of pattern P, and $edits(S, f(S))$ is a set of edits that can transform pattern $f(S)$ to S, which stands for the description length of corrections.

The author further introduces the parameter α to control the importance of minimizing information loss over reducing visual clutter in the overview. The third term with the parameter λ is added to directly control the total number of patterns. Increasing λ will reduce the number of patterns in the optimized result. Therefore the scalability of the overview can be improved by setting λ properly.

Then, as every pattern maps to a cluster of sequences, the author denotes a cluster as a tuple $c = (P, G)$ where G is the set of sequences mapped to pattern P. Therefore our goal is to find a set of tuples for all the clusters as $\mathscr{C} = (P_1, G_1), \ldots, (P_k, G_k)$ that minimize $L(\mathscr{C})$:

$$L(\mathscr{C}) = \Sigma_{P,G \in \mathscr{C}} \Sigma_{S \in G} len(P) + \alpha \Sigma_{P,G \in \mathscr{C}} \Sigma_{S \in G} ||edits(S, P)|| + \lambda ||\mathscr{C}||$$

Algorithm

The goal is to find the set of the pattern and the corresponding group of sequences. The main algorithm goes like this, we first initialize every sequence as a pattern, and then find all the candidate pair of patterns that forms a shorter merging. for all the candidates, We start the merging from the one that changes the most and continue the process. Finally we get the result set.

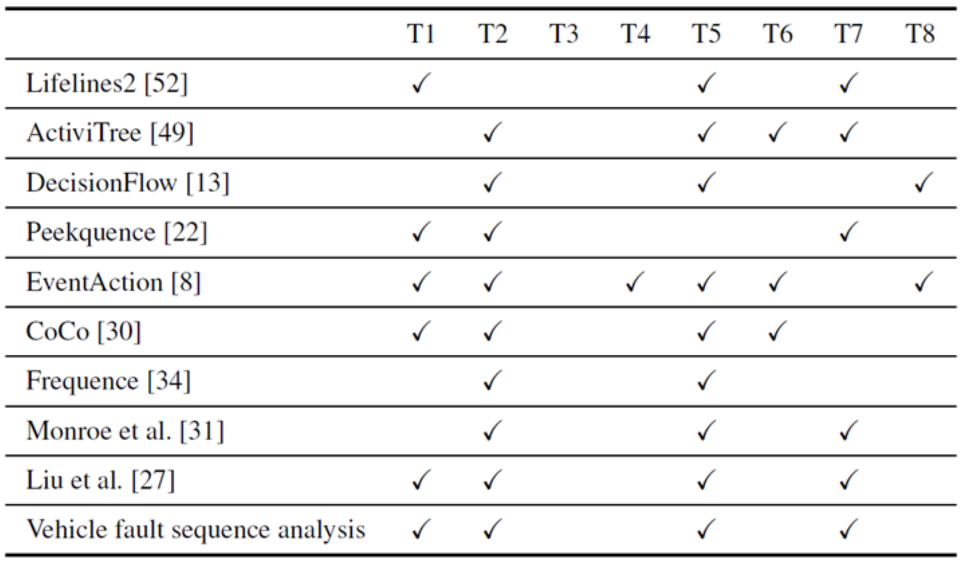
```
Input: sequences $\mathcal{S} = \{S_1, S_2, ..., S_n\}$
Output: pattern and cluster tuples
        $\mathcal{C} = \{(P_1, G_1), (P_2, G_2), ..., (P_k, G_k)\}$
/* Initialization phase                                  */
$\mathcal{C} = \{(P, G) | P = S, G = \{S\}$ for all $S \in \mathcal{S}\}$;
PriorityQueue $Q = \emptyset$;
for all pairs $c_i, c_j \in \mathcal{C}$ and $i \neq j$ do
    $\Delta L, c^* = Merge(c_i, c_j)$;
    if $\Delta L > 0$ then
        insert $(\Delta L, c^*, c_i, c_j)$ into $Q$;
    end
end
/* Iterative merging phase                                 */
while $Q \neq \emptyset$ do
    retrieve $(\Delta L, c^*, c_i, c_j)$ from $Q$ with the largest $\Delta L$;
    $c_{new} = c^*$;
    remove $c_i, c_j$ from $\mathcal{C}$, add $c_{new}$ to $\mathcal{C}$;
    remove all pairs containing $c_i$ or $c_j$ from $Q$;
    for $c \in \mathcal{C} - c_{new}$ do
        $\Delta L, c^* = Merge(c, c_{new})$;
        if $\Delta L > 0$ then
            insert $(\Delta L, c^*, c, c_{new})$ into $Q$;
        end
    end
end
return $\mathcal{C}$
```

However, the algorithm above is iterative and has a large complexity. So the author added Locality Sensitive Hashing to Speedup. With this approach, pairs of sequences/patterns which share very few common events can be skipped when searching for candidate pairs. LSH goes like this:

- Set a predefined threshold th.
- If two multisets have a similarity larger than th, they will have the same hash value with a high probability.
- Quickly identify pairs of sequences/patterns with similar sets of events regardless of their exact order, and prioritize their clusters.
- To ensure no possible merging is missed, gradually decrease th in the later runs.

System

In a previous paper in 2016, the author summarized a set of high-level analytic tasks for event sequence data, from T1 to T8 [5]. then in this paper, they survey the existing visual analytic systems for event sequence data and list the tasks they support here.



	T1	T2	T3	T4	T5	T6	T7	T8
Lifelines2 [52]	✓				✓		✓	
ActiviTree [49]		✓			✓	✓	✓	
DecisionFlow [13]		✓			✓			✓
Peekquence [22]	✓	✓					✓	
EventAction [8]	✓	✓		✓	✓	✓		✓
CoCo [30]	✓	✓			✓	✓		
Frequence [34]		✓			✓			
Monroe et al. [31]		✓			✓		✓	
Liu et al. [27]	✓	✓			✓		✓	
Vehicle fault sequence analysis	✓	✓			✓		✓	

They conclude the four tasks T1, T2, T5, T7 to be the most common and centered design around these tasks:

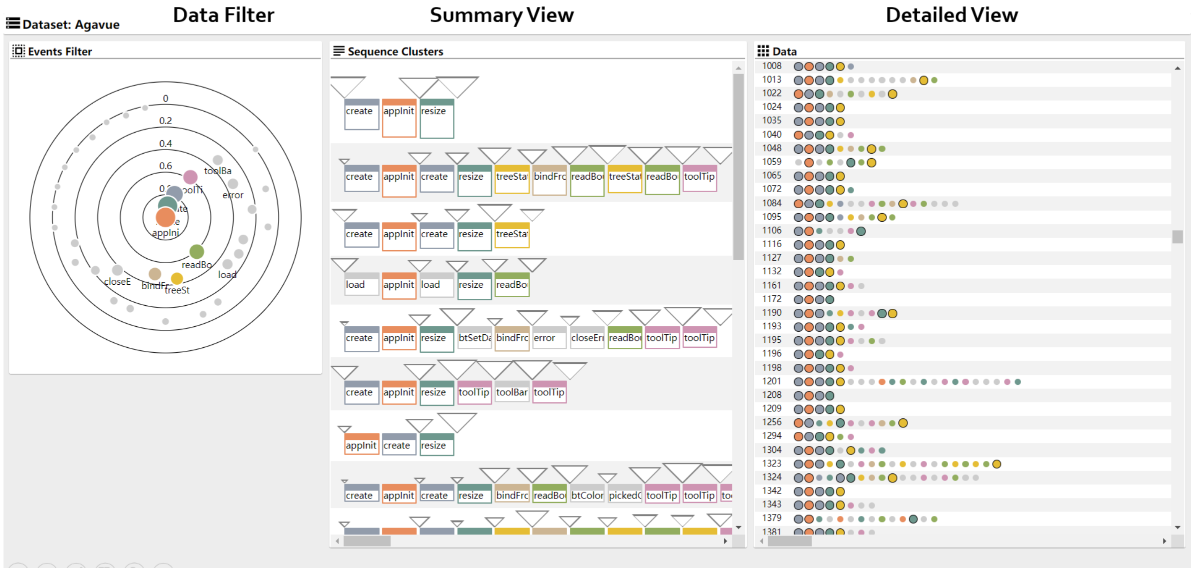
T1. Review in detail a few records.

T2. Compile descriptive information about the dataset or a subgroup of records and events (esp. through aggregated views).

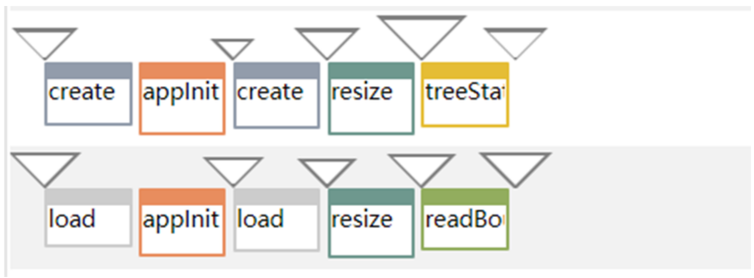
T5. Identify a set of records of interest.

T7. Study antecedents or sequelae of an event of interest.

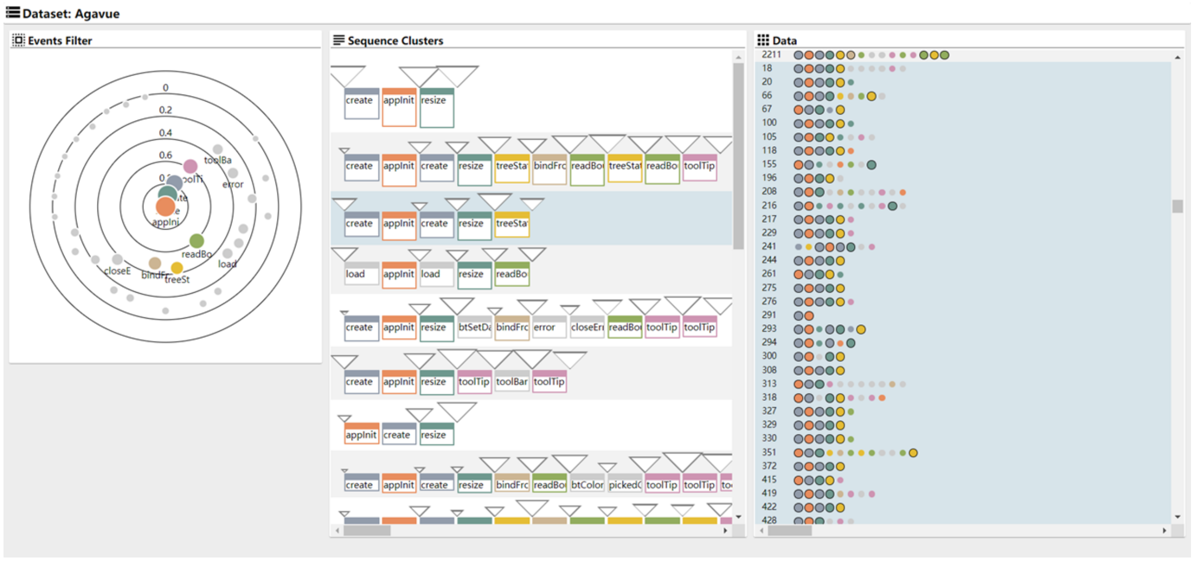
Then in this part I will introduce **my visual analytics system** and how the four tasks are fulfilled. As you can see there are three main views, data filter and summary view and detailed view.



The summary view shows the sequential patterns, which fulfill T2, show the descriptive information about the dataset or subgroup. In this view, each rectangle is an event and each row is a sequential pattern. The height of rectangle represents the quantity. The triangles stands for corrections and the size of triangles represents the number of insertions.

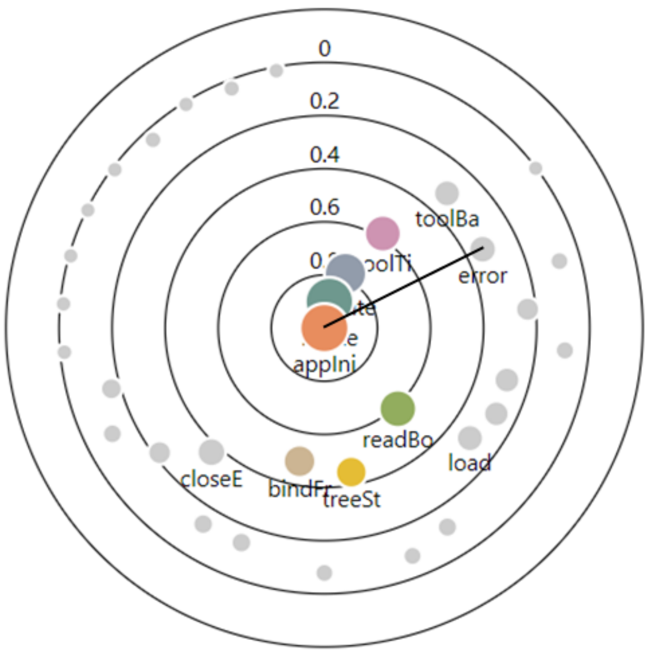


By interactions like clicking the sequential pattern, users can see the corresponding sequences in detail view, which fulfill T1, review in detail a few records.

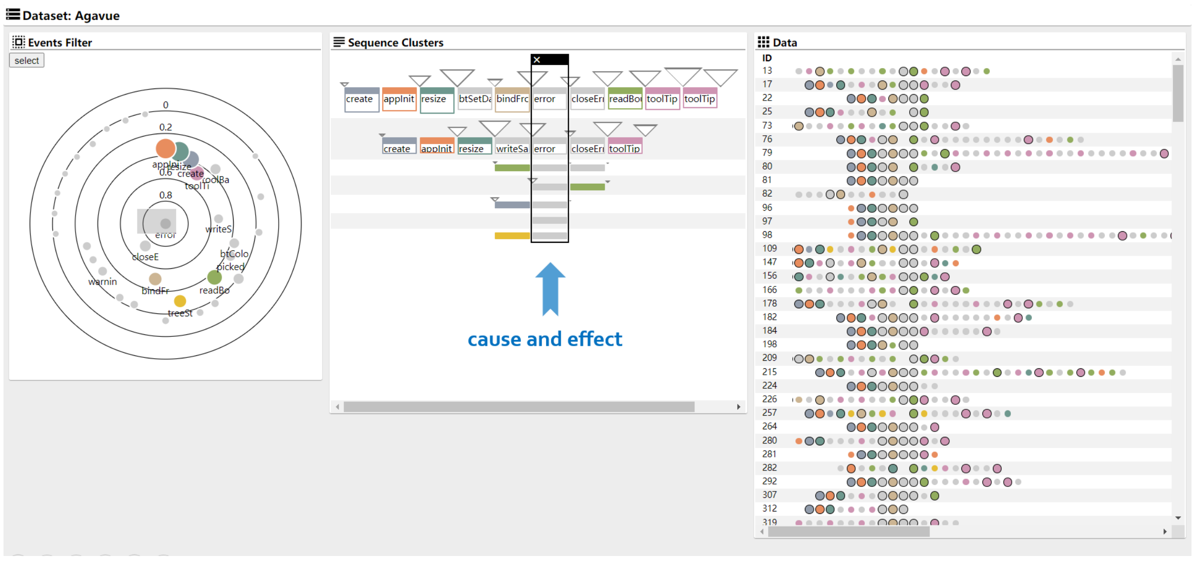


Users can also filter events by their co-occurrences in data filter and see the corresponding detail or patterns overview, which fulfill T5, Identify a set of records of interest. This also fulfill T1 and T2 mentioned above by showing overview in summary view and showing detail records in detailed view.

Here the event filter shows the co-occurrences of all the events, with the focus at the center. The co-occurrence is encoded as the radio distance to the focus. User can change the focus or select a subset of events by lasso tool.



Moreover, by double clicking the event rectangle, users can align the sequences in both summary and detail view, which can fulfill T7, study the cause and effect of a specific event.

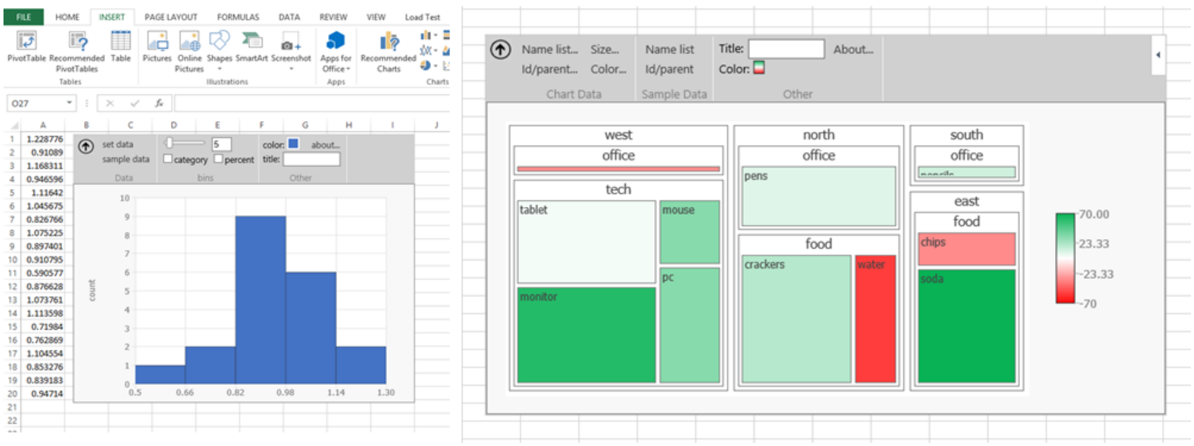


Implementation

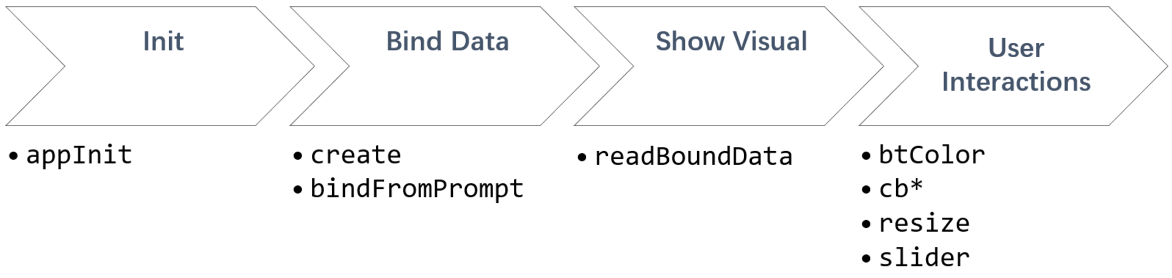
Dataset

This usage scenario of the dataset is application log analysis. Desktop, web or mobile applications can collect large amount of usage log data recording user interactions and many other events in the system.

We use a public dataset Agavue [6], which is from a **data visualization application in Excel**. It contains about 2000 user sessions, 34 kinds of events.



Roughly, a session looks like this: A user selects data, creates a data binding between data and visualization. The system visualizes their data, then the user modifies or interacts with the visualization.



Data Preprocessing

This part I use Python to preprocess the data. Each log record contains sequence id, event type, time and detail information. After observation, we will find some data that the user continuously did many the same kind of operations. As we need to calculate sequence patterns, all of these resize events should be considered as one single event. So our data preprocessing work is to merge adjacent events of the same type.

Algorithm and Visual Analytics System

I use Python to finish the algorithm and use React and D3 library to implement the visual analytics system.

For algorithm, I just follow the algorithm principle mentioned above. With this algorithm, we get **41 patterns** from 2211 event sequences. On average, each pattern has **5.7** events, and each corresponds to **54** event sequences.

For the visual analytics system, I have already introduced my interface in detail in the previous part.

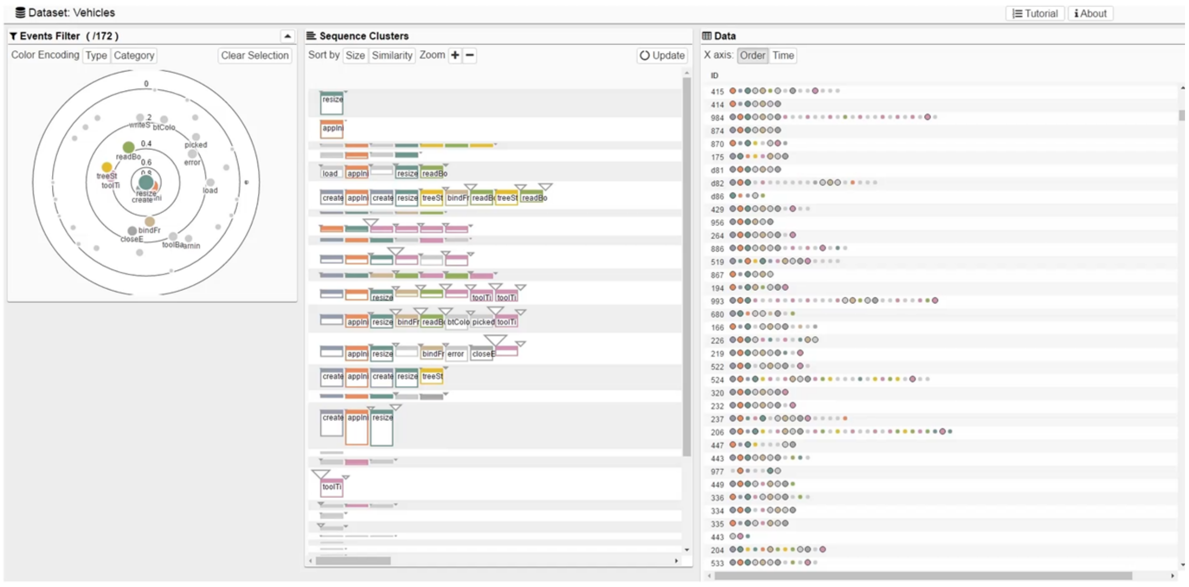
Evaluation

As is mentioned above, the system can fulfill the four tasks. This part I will mainly introduce the comparison with original paper and usage scenario.

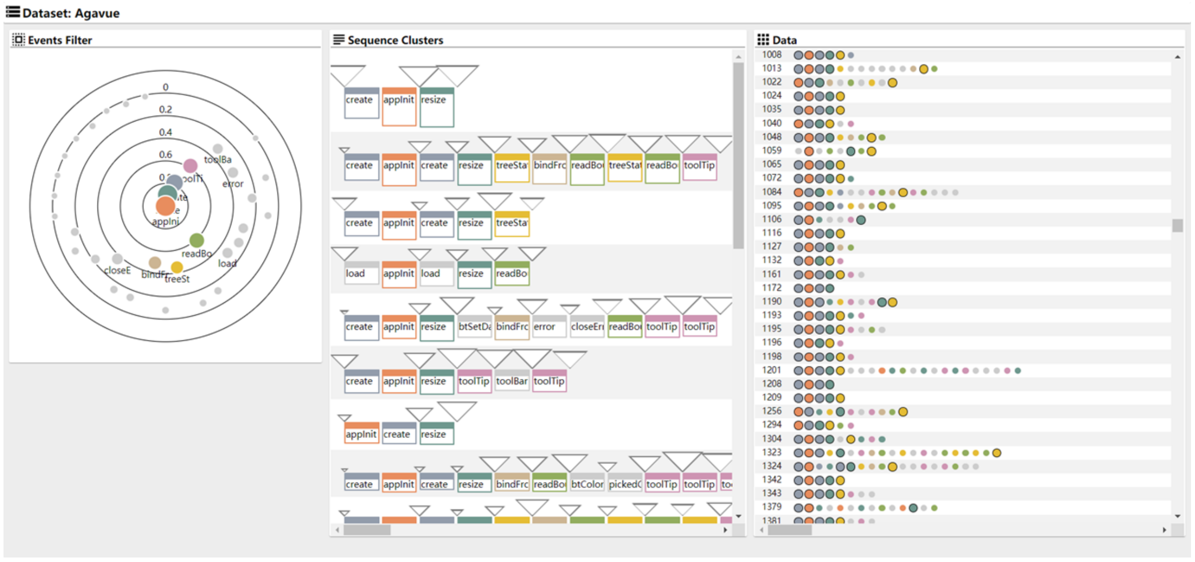
Comparison

Here I will put a **comparison** with the original paper.

Original:



My Result:



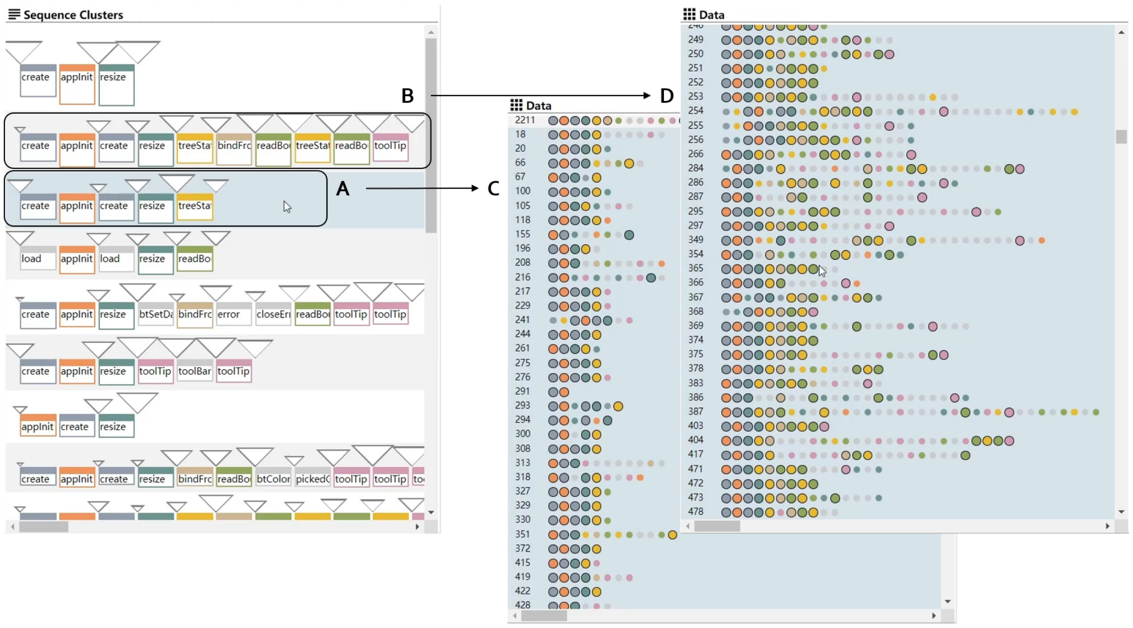
Usage Scenario

This part is divided into overview and cause & effect. I have an additional demo video as a supplement.

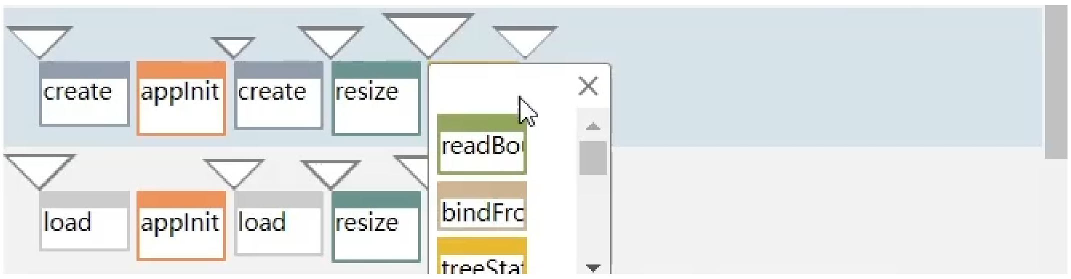
- **Overview**

  We start with an overview of the data as the following picture. Not surprisingly, most patterns(like A, B) contain a typical sequence of operations including initializing the app (appInit, create), resizing the window, binding data (bindFromPrompt, readBoundData).

  We can click on the patterns to review the sequences in each group. For example, pattern A represents a group of sequences with better consistency, indicated by the smaller sizes of the triangles. While this pattern B represents a group of sequences that are consistent in the first few events however have more significant deviations afterwards. This observation can be easily verified by looking at the detailed views (C, D).

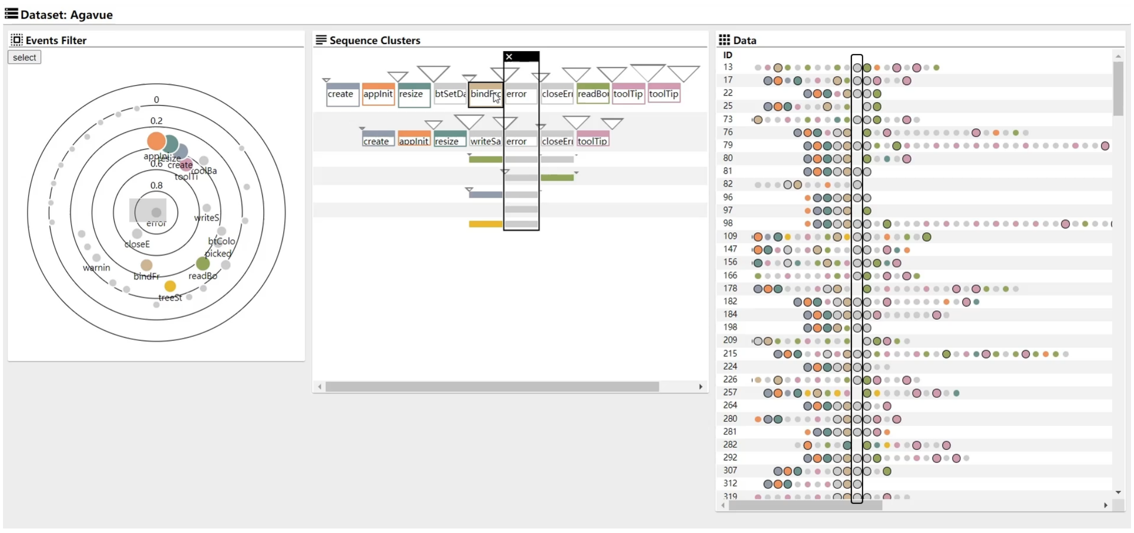


We can also double-click on the triangle to see the insertion detail.



- **Cause and effect relation analysis**

  From filter view, we can easily see the co-occurrence of events with a focus event at the center. As we want to know the relationship between error events and other events, we change the focus to error. You can see that the closest event types are closing error window, binding data and writing data. With lasso tool, we can select the patterns that contain these events. We can also just focus on error event. To learn the cause and effect of error event in detail, we can align the patterns at error event. We can see that most errors occur after users trying to bind data to the visualization, or writing sample data to the sheet.

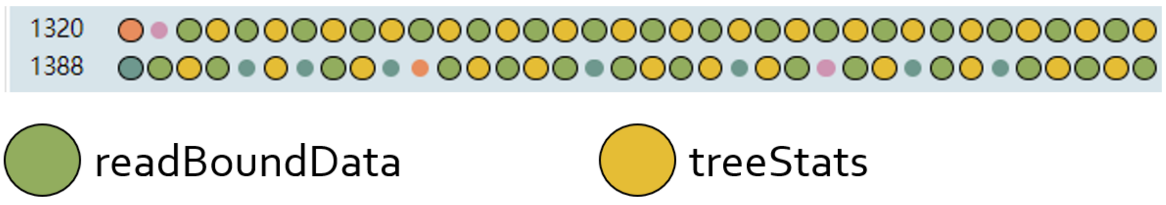


One possible explanation is that the users may not be familiar with the data format requirements, which indicates that better interface for data binding and writing can be designed to further improve user experience.

Discussion

This part I will mainly talk about the limitation.

For the preprocessing part, although we have merged the adjacent events of the same type, as you can see here, these two sequences have many repeated green and yellow events, which will also influence the result. To solve this, we can merge the adjacent repeated part as well, not only for the single event.



For algorithm, the algorithm after speedup is still much slower than that of the original paper. As I use some python library here, I think the reason is that the speed of the library is not very fast, so in the future we can use c++ to replace python and re-implement this algorithm again.

Finally for the visual analytics system, as is mentioned in using scenario, double-clicking the triangle will show the insertion detail events. However, in this way we lost the sequential information between these events. Ideally we are supposed to show short patterns here, so we can calculate the short pattern when user double-clicks the triangle to achieve this effect.

Conclusion

I spent 10 days working on this project. Although I have some experience using React and Vue before, this is the first time I didn't use any libraries such as Antd and Echarts, so I gained a lot of coding knowledge in this process.

Since the paper only describes the overall algorithm implementation, many details need to be further refined by myself. In this thinking process, I often found that many concepts and details were actually not clear for me, although I have read the paper in detail before. So reproducing the paper helped me further clarify the concept.

What's more, for me, It is a pity that, due to the lack of time, I only achieved the original result of the paper, and did not make any further contributions. I think I can improve it in the future, like pattern query and event importance.

Finally, I am very grateful to Senior Wu Jiang, who gave me a lot of help in this process.

References

[1] P. J. Polack, S.-T. Chen, M. Kahng, M. Sharmin, and D. H. Chau. Timestitch: Interactive multi-focus cohort discovery and comparison. In Visual Analytics Science and Technology (VAST), 2015 IEEE Conference on, pp. 209–210. IEEE, 2015.

[2] B. C. Kwon, J. Verma, and A. Perer. Peekquence: Visual analytics for event sequence data. In ACM SIGKDD 2016 Workshop on Interactive Data Exploration and Analytics, 2016.

[3] G. Wang, X. Zhang, S. Tang, H. Zheng, and B. Y. Zhao. Unsupervised clickstream clustering for user behavior analysis. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, pp. 225–236. ACM, 2016.

[4] J. Wei, Z. Shen, N. Sundaresan, and K.-L. Ma. Visual cluster exploration of web clickstream data. In Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on, pp. 3–12. IEEE, 2012.

[5] C. Plaisant and B. Shneiderman. The diversity of data and tasks in event analytics. In Proceedings of the IEEE VIS 2016 Workshop on Temporal & Sequential Event Analysis, 2016.

[6] D. Fisher. Agavue event data sample: Full dataset. version of october 20,2016. microsoft research. retrieved from http://eventevent.github.io.