Eurographics Conference on Visualization (EuroVis) 2017
J. Heer, T. Ropinski and J. van Wijk
(Guest Editors)

Volume 36 (2017), Number 3

CoreFlow: Extracting and Visualizing Branching Patterns from Event Sequences

Zhicheng Liu[1], Bernard Kerr[2], Mira Dontcheva[1], Justin Grover[2], Matthew Hoffman[1,3], Alan Wilson[2]

[1]Adobe Research [2]Adobe Systems Inc. [3]Google Research

Abstract

Event sequence datasets with high event cardinality and long sequences are difficult to visualize and analyze. In particular, it is hard to generate a high level visual summary of paths and volume of flow. Existing approaches of mining and visualizing frequent sequential patterns look promising, but have limitations in terms of scalability, interpretability and utility. We propose CoreFlow, a technique that automatically extracts and visualizes branching patterns in event sequences. CoreFlow constructs a tree by recursively applying a three-step procedure: rank events, divide sequences into groups, and trim sequences by the chosen event. The resulting tree contains key events as nodes, and links represent aggregated flows between key events. Based on CoreFlow, we have developed an interactive system for event sequence analysis. Our approach can compute branching patterns for millions of events in a few seconds, with improved interpretability of extracted patterns compared to previous work. We also present case studies of using the system in three different domains and discuss success and failure cases of applying CoreFlow to real-world analytic problems. These case studies call forth future research on metrics and models to evaluate the quality of visual summaries of event sequences.

1. Introduction

Temporal event sequences, such as application log data and web visitor clickstreams, are valuable in helping us understand user behavior and in informing decisions. Visualization and analysis of event sequence data is a well-studied research area [LRTM07, LWD*17, MLL*13, PW14, WZT*16, ZDF15, ZLD*15] but remains an unsolved problem. Depending on the domain, a sequence dataset can contain thousands or more distinct sequences. Each of these sequences may consist of hundreds of ordered events. The number of unique events can be in the hundreds or thousands. The volume and complexity of such datasets render traditional visualization techniques inadequate for effective analysis. Even when we aggregate events into fewer categories, it is still difficult to provide a useful overview of the paths and their volume (Figure 1).

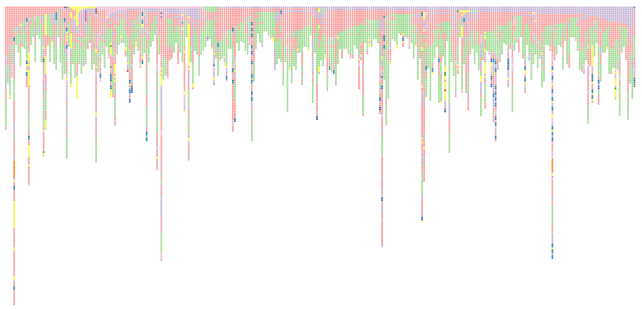


Figure 1: An icicle plot [KL83] visualizing 10143 events in 300 clickstream sequences. Each sequence goes from top to bottom, where the events are represented as small rectangles. The color of the rectangles represents event category.

Instead of trying to show all the events and sequences, visual analytics researchers employ data mining methods to extract and visualize meaningful high-level abstractions for interactive analysis [LWD*17, PW14, WZT*16, WSSM12]. Frequent pattern mining, in particular, has shown promise in extracting interpretable and useful insights from complex event sequences [KVP16, LWD*17, PW14]. A *frequent pattern* is a set of ordered or unordered events that frequently co-occur. Many pattern mining algorithms require a parameter called *minimum support*, which specifies the minimum number of input sequences containing the patterns. For example, Figure 2 shows six sample sequences with 12 unique events, and Figure 3 shows frequent sequential patterns (Here we are mining a type of frequent patterns called *closed sequential patterns*, refer to [Bah03] for detailed definition.) extracted from the input sequences with minimum support set to 30% (i.e. each pattern represents at least 2 sequences).

Visualizations based on sequential pattern mining offer a visual summary while still supporting users in drilling down to individual sequences from the patterns [KVP16, LWD*17]. From a human-centered perspective, however, sequential patterns still have several limitations in aiding the analysis of event sequences:

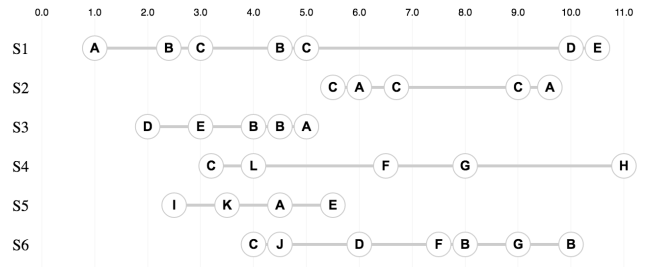


Figure 2: Six event sequences with 12 unique events (A - L). The horizontal axis represents the timestamp (in seconds) for the events.

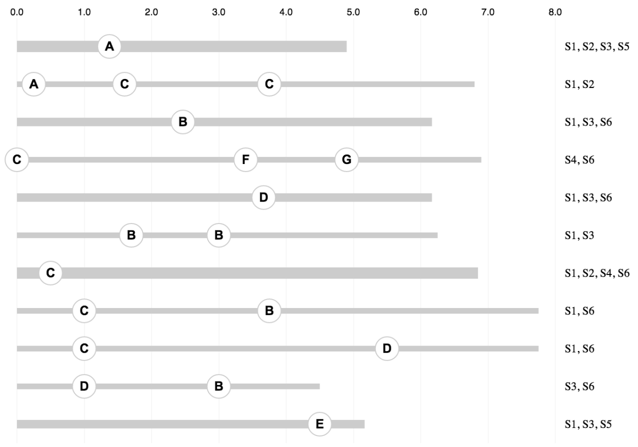


Figure 3: Eleven sequential patterns extracted from the input sequences in Figure 2 with minimum support set to 30%. The horizontal position of the events represent the average time elapsed relative to the beginning of the sequences. For example, the second pattern shows that two sequences share the ordered events “A”, “C” and “C”, and it takes on average 0.25 second to observe the first occurrence of “A”. The line width represents the number of sequences containing that pattern. For each pattern, we show the ID of the input sequences containing that pattern.

Scalability: When the dataset is large, computing patterns is expensive and unsuitable for interactive analysis [LWD*17]. With hundreds of thousands of events, the pattern mining time can take minutes or even hours.

Interpretability: While sequential patterns are effective in reducing the number of events to be displayed, the number of patterns can outgrow the number of input sequences, making it hard to review all the patterns. These patterns also tend to have significant overlap. For example, Figure 3 shows that the same sequence is often represented in multiple patterns. It is also not clear how these sequential patterns relate to each other, and how they fit together to provide an overview of the data.

Utility: Finally, frequent events do not always correspond to important or meaningful milestones, and there is little research that evaluates how useful sequential patterns are in actual data analysis.

In this paper, we present CoreFlow, a novel technique that extracts *branching patterns* from event sequences by recursively applying the *Rank-Divide-Trim* three-step procedure. In the *Rank* step, events are ranked using a frequency-based function. In the *Divide* step, sequences are separated into two groups. Finally, in the *Trim* step, the subsequences leading to the top-ranked event are removed. The resulting branching pattern is a tree showing an overview of path flow across frequent events in the sequences. We implement CoreFlow and explore several visualization and interaction techniques in the context of the CoreFlowVis system.

We evaluate our approach in terms of scalability, interpretability and utility. Compared to approaches that mine and visualize frequent sequential patterns, CoreFlowVis is more scalable for large datasets with less computational time. The generated visualization presents a unifying view of frequent events with improved interpretability. To investigate if the branching patterns are meaningful and useful, we conduct three case studies with domain experts. Results show that frequency is a reasonable metric in automatically picking important events in clickstream and media touch point analysis. We discuss the limitations of our approach in the domain of application log analysis and suggests future directions for further improving the system.

2. Related Work

Visualizing event sequences as trees is not a new idea. Previous work has applied tree visualization techniques to event sequence data. FP-Viz uses the SunBurst [SZ00] approach to represent the frequent pattern growth tree [KSS05]. LifeFlow [WGGP*11] and Event-Flow [MLL*13] apply the icicle tree visualization [KL83] to visualize temporal event sequences such as health records. These systems are effective when the number of events in each sequence is small (e.g. less than 20), and users manually filter and aggregate events through a user interface to simplify visualizations. These approaches are less scalable to longer sequences where the events have high cardinality. Recognizing these limitations, DecisionFlow [GS14] proposes an approach similar to the rank and divide procedures in CoreFlow. The main difference between our work and DecisionFlow is that Coreflow tries to automatically identify key events while DecisionFlow starts with user-defined milestones. In addition, CoreFlow focuses on representing branching patterns as trees. The interactive visualizations based on the CoreFlow technique also incorporate several novel features including drilling down to perform pattern extraction on tree branches.

In addition to frequent pattern mining, which we discussed in the introduction, research on event sequence analysis and visualization has also explored the potential of various sequence clustering approaches. In particular, hierarchical clustering is a commonly used method where the results are also visualized as trees [WZT*16, ZBS16]. The difference between CoreFlow and hierarchical clustering of sequences is evident when we examine how the trees are interpreted. In the trees generated by CoreFlow, the nodes are events and the links represent flows between the events. In hierarchical clustering, the nodes in the resulting hierarchy represent sequence clusters, not events. To emphasize this difference, we choose to use the term *branching pattern* instead of *hierarchical pattern.* A child event in a branching pattern is not a subordinate or a member of its parent event. The tree structure only indicates that the parent event appears earlier than the child event in the sequences. Hierarchical clustering is better suited for tasks such as user segmentation [ZBS16], while CoreFlow is useful for interactive exploration of sequences.

Finally, CoreFlow is also inspired by works like WordTree [WV08], which extract tree structures from text documents. CoreFlow deals with temporal event sequences, and requires novel algorithms and visualization considerations for different problem domains.

3. CoreFlow: Extracting Branching Patterns

The design of CoreFlow is motivated by a metaphor used by analysts to describe their understanding of temporal event sequences. If we think of each sequence as a traveler's journey, an effective visualization should show an overview of the important milestones (events) along the way. Despite differences in the exact paths and time taken, the travelers may share a few common milestones in the earlier parts of their journeys. At reaching certain milestones, some travelers end their journeys early, while others branch out to other milestones and destinations.

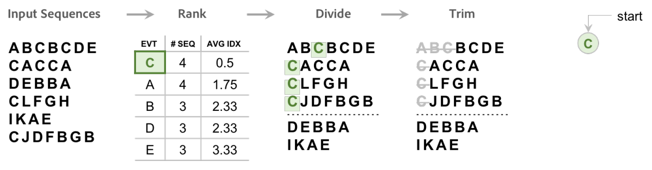
The CoreFlow technique tries to extract branching patterns based on this metaphor. To do so, CoreFlow recursively applies a strategy consisting of three steps: 1) *ranking* the events based on a ranking function and picking the top ranked event *e*, 2) *dividing* sequences into two groups using *e*: those that contain *e* and those that do not, and 3) *trimming* sequences containing *e* from the beginning to the first occurrence of *e*. At each recursive call, the top ranked event gets added to the tree as the node. The recursion stops when the end of the sequences are reached or when the number of sequences containing *e* is smaller than a pre-defined *minimum support*. The minimum support is the only parameter required.

3.1. Recursive Rank-Divide-Trim: An Example

Figure 4 illustrates all the steps involved in extracting a branching pattern from the six sample sequences shown in Figure 2. We define a minimum support of 30% (i.e. any link in the branching pattern should represent at least two input sequences) and use a depth-first strategy in this example. A breath-first approach gives the same result.

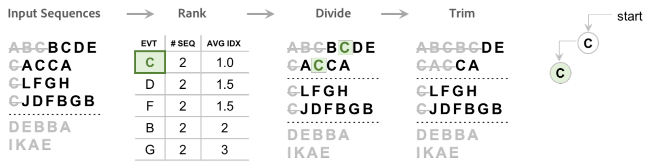
Ranking events: Starting with all the sequences, we rank the events with a predefined ranking function. For any pair of events, the event with a higher frequency (the number of sequences it appears in) gets a higher rank. If two events appear in the same number of sequences, for each event, we compute the average index of the first occurrence of the event in the sequences. The event with a lower average index is ranked higher. Figure 4(1) shows that "C" is the highest ranked event in the input sequences. In using this ranking function, we characterize the importance of an event (milestone) using the number of sequences passing through it. Since the journeys can be long and contain multiple milestones, we give preference to milestones that appear earlier in the journeys.

Dividing sequences: We add the top-ranked event "C" to the tree and use it to divide the sequences into two groups: those that contain



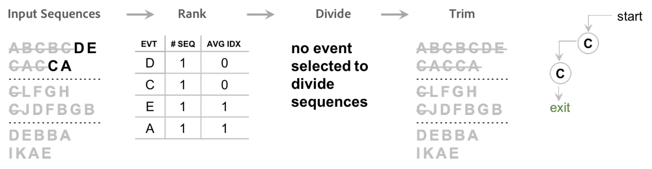
EVT	# SEQ	AVG IDX
C	4	0.5
A	4	1.75
B	3	2.33
D	3	2.33
E	3	3.33

(1) Choose "C" as the first milestone event



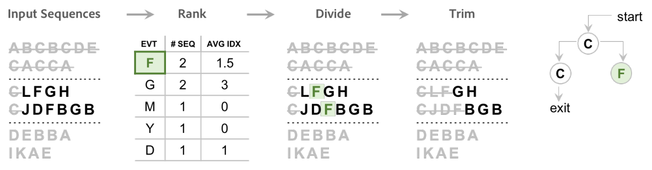
EVT	# SEQ	AVG IDX
C	2	1.0
D	2	1.5
F	2	1.5
B	2	2
G	2	3

(2) "C" is ranked highest again for the trimmed sequences



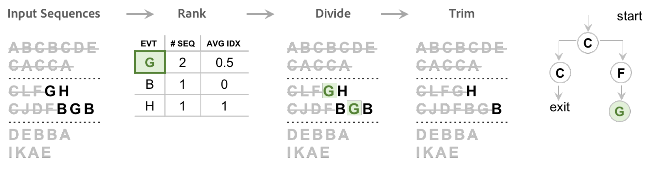
EVT	# SEQ	AVG IDX
D	1	0
C	1	0
E	1	1
A	1	1

(3) Top ranked event appears in less than 2 sequences, stop computing for current group



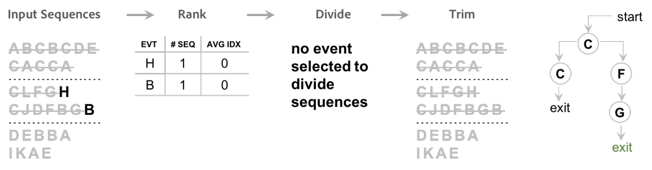
EVT	# SEQ	AVG IDX
F	2	1.5
G	2	3
M	1	0
Y	1	0
D	1	1

(4) "F" is the top ranked event in the two trimmed sequences



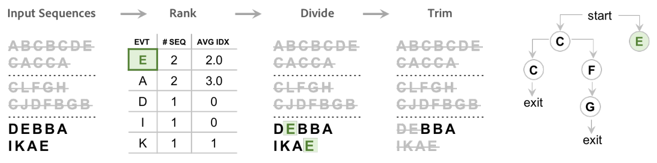
EVT	# SEQ	AVG IDX
G	2	0.5
B	1	0
H	1	1

(5) Add "G" as the milestone event after "F"



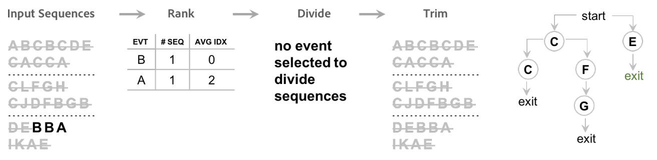
EVT	# SEQ	AVG IDX
H	1	0
B	1	0

(6) Top ranked event appears in less than 2 sequences, add "exit" node



EVT	# SEQ	AVG IDX
E	2	2.0
A	2	3.0
D	1	0
I	1	0
K	1	1

(7) Perform same routines for sequences that do not contain "C"



EVT	# SEQ	AVG IDX
B	1	0
A	1	2

(8) End the extraction process. All branches terminate at "exit"

Figure 4: Step-by-step process to extract a branching pattern. The tables show top 5 events in the (sub)sequences with event name, number of sequences containing that event, and the average index of the event.

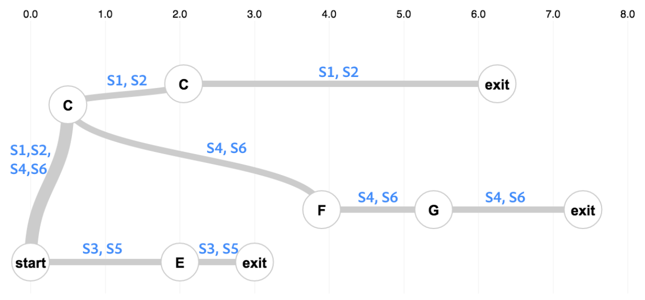


Figure 5: A visualization of the branching pattern extracted in Figure 4. The horizontal distance between adjacent nodes represent the average time taken to go from the parent node to the child node. The link width represents the number of sequences following the path, and we annotate the links with the raw sequence IDs. The branching pattern provides a succinct overview of the input sequences. Compared with visualization based on frequent pattern in Figure 3, the branching pattern provides a unified view connecting frequent events in continuous paths.

"C" and those that do not. The figure separates these two groups with a dotted line (Figure 4(1)).

Trimming sequences: For the four sequences containing "C", we trim these sequences from the beginning to the first occurrence of "C" (Figure 4(1)).

This completes the first round of the rank-divide-trim procedure. We now have two groups of sequences: one contains trimmed sequences and the other contains sequences where there is no observation of the event "C". The same procedure in 4(1) applies to each of these sequence groups. We recursively apply the 3-step procedure to construct the branching pattern, until one of the following conditions is met: we reach the end of the sequences, or the number of sequences containing the top ranked event in the division is smaller than the minimum support. In these cases, we group all the trimmed sequences into one branch, with "exit" as the terminal node. Figure 5 shows a visualization of the final branching pattern extracted. In this example, the pattern is a binary tree, where any node has at most two children. Depending on the dataset, this is not always the case. The branching pattern can be a tree of arbitrary levels and branching factor.

Algorithm 1 in the supplemental materials outlines the complete CoreFlow technique with the three-step approach in pseudo code.

3.2. Frequency as a Metric of Event Importance

In the ranking function used in the previous section, we use frequency as a primary metric for choosing events as milestones. This decision is based on longitudinal studies with domain experts analyzing data such as web clickstreams and application logs (Section 6). While there have been no agreed-upon objective metrics for ranking events in terms of importance, the analysts confirm that frequency of appearance is a reasonable and commonly used metric for choosing important events. We evaluate whether the branching patterns generated using this approach are meaningful and useful in Section 6.

Frequency-based ranking function can also have many variants. For example, there are many ways to break ties when two events appear in the same number of sequences. Instead of using average index in the example above, we can use median index; we can also compare average or median timestamp instead of comparing index. We have experimented with these approaches using different real-world datasets. In most cases, the choice does not matter because rarely do two events share the same frequency of appearance.

In the example, we use the number of enclosing sequences (SEQ) as the metric to rank events. If a sequence has multiple occurrences of the same event, we count the sequence only once. There are other frequency-based metrics such as the number of occurrences (OCCURRENCE) and the number of occurrences as the head of sequence (HEAD). The OCCURRENCE ranking function gives preference to repeating events in potentially fewer sequences; the HEAD ranking function incorporates the position of events in the metric. We experimented with these approaches using different datasets. Theoretically, the OCCURRENCE and HEAD approaches should result in more branches. However, we find that the actual properties of the branching patterns depend more on the characteristics of the dataset. The supplemental materials include statistics of the generated branching patterns in terms of number of nodes, number of unique nodes, tree height and average branching factor.

4. Visualization and Interaction Design

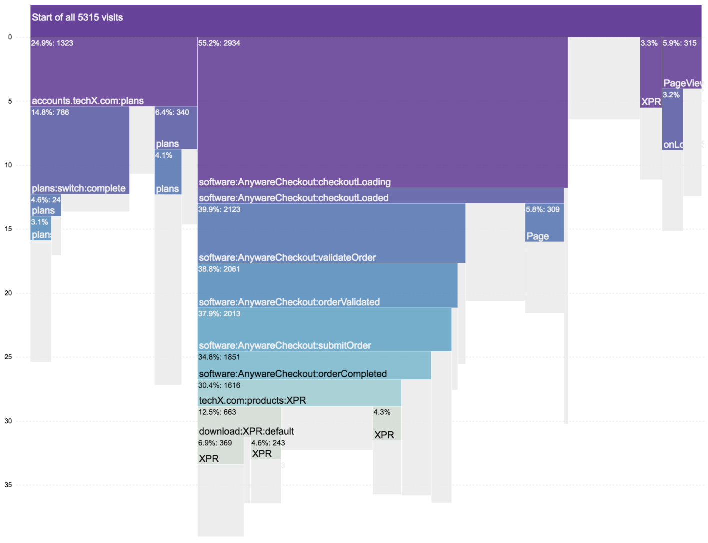
We have designed and built a system called CoreFlowVis that visualizes branching patterns extracted using the CoreFlow technique. The system provides user interfaces to select dataset, change parameters such as minimum support and canvas size, and define funnels or drill down to a link in the pattern.

4.1. Tree Visualization Design

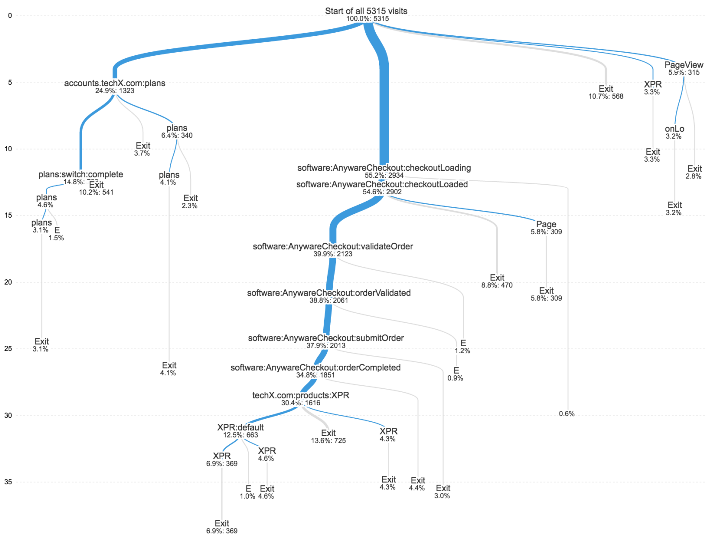
A variety of tree visualization techniques can be applied to display branching patterns. CoreFlowVis offers three different representations: an icicle plot (Figure 6(a)), a node-link visualization (Figure 6(b)) and a hybrid design combining features of the node-link visualization and the icicle plot (Figure 6(c)). In all three visualizations, the vertical axis represents the average number of events or the average time elapsed since the beginning of sequences. We apply a vertical orientation for these visualizations in order to have more room to display event names and avoid cluttering.

In a traditional vertical icicle plot, each rectangular partition represents a tree node, each row represents a level and the labels are placed at the center of each partition[†]. It is not necessary to render branches in the tree because all the nodes at the same level have the same vertical position. In the CoreFlow icicle plot, the nodes have variable vertical positions, so we need to use the partitions to represent branches. We display branch statistics at the top of the partition, and place the event labels at the bottom of each partition to indicate that the bottom is where the event is first observed (Figure

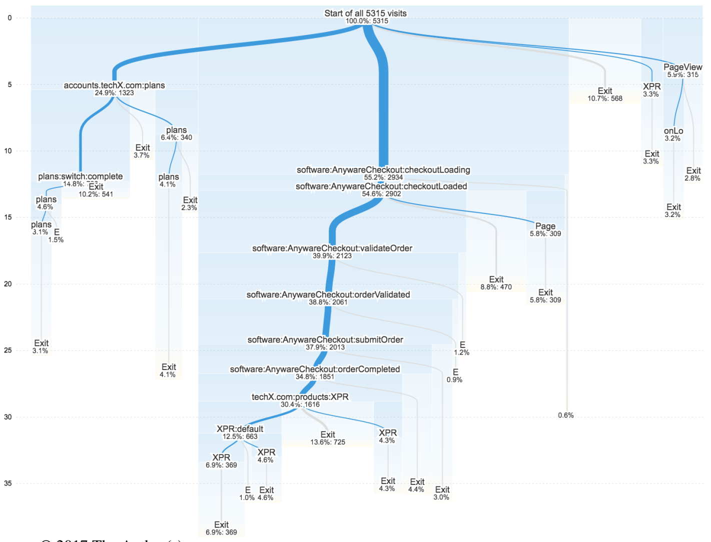
[†] For an example, visit http://bl.ocks.org/mbostock/4347473



(a) Icicle Plot

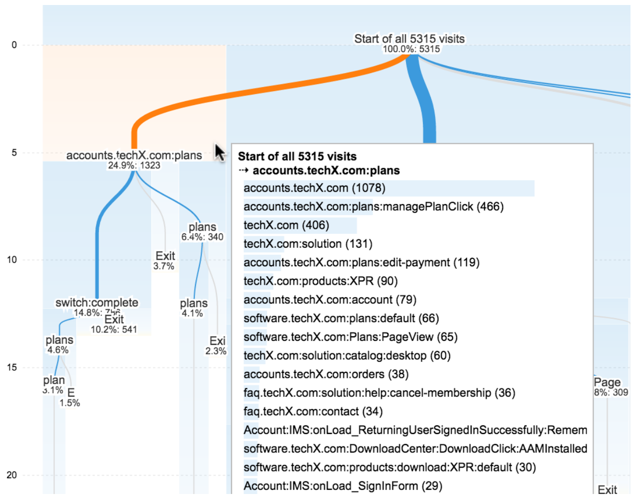


(b) Node-Link Visualization



(c) Hybrid visualization with an icicle plot as the background of a tree

Figure 6: Three visualization designs implemented based on CoreFlow



Start of all 5315 visits
⇢ accounts.techX.com:plans
accounts.techX.com (1078)
accounts.techX.com:plans:managePlanClick (466)
techX.com (406)
techX.com:solution (131)
accounts.techX.com:plans:edit-payment (119)
techX.com:products:XPR (90)
accounts.techX.com:account (79)
software.techX.com:plans:default (66)
software.techX.com:Plans:PageView (65)
techX.com:solution:catalog:desktop (60)
accounts.techX.com:orders (38)
faq.techX.com:solution:help:cancel-membership (36)
faq.techX.com:contact (34)
Account:IMS:onLoad_ReturningUserSignedInSuccessfully:Remem
software.techX.com:DownloadCenter:DownloadClick:AAMInstalled
software.techX.com:products:download:XPR:default (30)
Account:IMS:onLoad_SignInForm (29)

Figure 7: Hovering over a branch displays the most frequent events in the branch

6(a)). The width of the partitions represents the number of sequences. We use a sequential color scheme to indicate the level of node. The gray partitions represent exit branches. Our final design resembles the LifeFlow visualization [WGGP*11], with minor differences in terms of layout orientation and color scheme.

For the node-link visualization, initially we tried to use Buchheim et. al.'s algorithm [BJL02] implemented in D3 [BOH11] to compute the tree layout, but did not achieve satisfactory results. Since the node position represents the average number of events or time elapsed instead of discrete level information, the algorithm fails to minimize edge crossings. In the end, we use the icicle layout to position the nodes, and draw curved links between the nodes. This layout approach also allows us to render the hybrid visualization easily.

The analysts used the icicle plot the most because it is effective in showing how the volume of outgoing links sums up to the number of sequences at the parent event. It is also easy to visually compare the volume of the branches. However, it took them some time to get used to the icicle plot, because it was a novel representation to them. They were more familiar with the node-link visualization, which is effective in showing explicitly the links between events. However, comparison is more difficult. The hybrid visualization attempts to combine the strengths of both visualizations, but the effectiveness is unclear. For example, in Figure 7, the same link is represented as a rectangle and a curved path, highlighted in orange. It is a little confusing to see the curved path cuts across other links represented as rectangles.

4.2. Incorporating User-defined Event Importance

As noted in Section 3.2, frequent events do not always correspond to meaningful milestones. The branching pattern generated by CoreFlow may not always contain the events that the analysts are interested in. In actual practices, analysts may want to define a funnel (i.e. an ordered list of milestone events), and visualize the volume

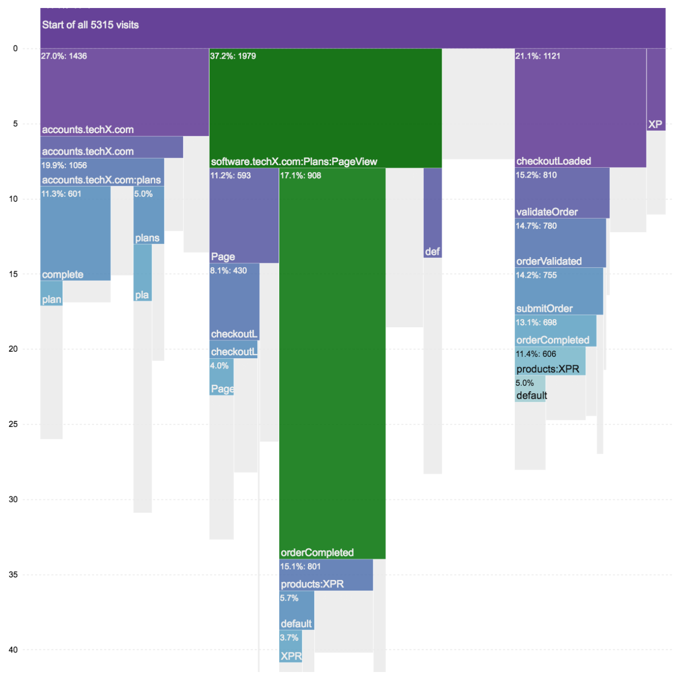


Figure 8: The analysts define a funnel consisting of two milestones: the plans page and a completed order. CoreFlowVis highlights this funnel in green, and displays it in the context of branching patterns.

of sequences flowing through the milestones. Such a query-driven approach is widely used in research works [GS14,LRTM07,ZDF15] and commercial applications (e.g. Google Analytics).

CoreFlow incorporates and augments funnel-based analysis by giving priority to user-defined milestones and ranking the events in the funnel higher than the other events. It is guaranteed that user-defined milestones will appear in the branching pattern and the visualization, even if they are not frequent. For example, in Figure 8, the analysts would like to see the traffic volume going through two important events: the plans page and the completed order page. They can define the funnel in terms of this two pages, and CoreFlow will give these two pages priority in the branching pattern generation process. It first divides sequences using the plans page as a milestone event. For the sequences containing the plans page, CoreFlow ranks the completed order page highest; for the other sequences not containing the plans page, the regular rank-divide-trim procedure applies. Figure 8 shows the branching pattern where user-defined event importance is incorporated. The defined funnel is highlighted in green, showing that 37.2% of the total visits went to the plans page, and 17.1% of the total completed an order. Since CoreFlow incorporates user queries in the pattern extraction process, we can also see what happens outside the funnel, as well as major patterns of how visitors dropped out of the funnel.

When a frequent event does not correspond to an event the analysts care about, CoreFlowVis allows analysts to exclude events by search for the event and adding it to an exclusion list. Events on this list will be ignored during the pattern computation process.

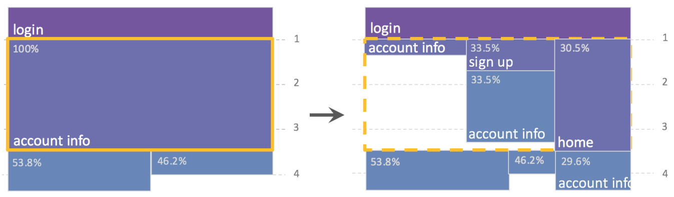


Figure 9: Drilling down on the link "login ⟶ account info". The link is highlighted with an orange outline (left). The new branching pattern for the subsequences in this link does not always fit within the original partition (right).

4.3. Details on Demand and Drilling Down

To provide information on the subsequences in the branching pattern, CoreFlow displays a tooltip when users hover over a link (Figure 7). The tooltip includes a bar chart showing the frequency of events in the subsequences. In this figure, we see that between the start of all sequences and the "accounts.techx.com:plans" event, the most frequent events are the clicks on accounts homepage and techX homepage.

The CoreFlow technique automatically generates an overview of temporal event sequences for users to start their exploratory analysis. In addition to examining the milestone events represented as tree nodes and the volume of sequences passing through the milestones, analysts can drill down to individual links of interest. Consider Figure 5, four sequences go through "C" after the start, and the analysts might be interested in this part of the journey and drill down to link "Start ⟶ C". The CoreFlow technique can be applied to any link in the branching pattern, allowing effective drilling down to the desired level of granularity. This feature is especially useful when the dataset is large and the overview only provides a high-level summary.

In the visualizations, users drill down to a link and compute branching patterns on subsequences in that link by double-clicking on the link. To display the newly computed branching pattern, we started by experimenting with the idea of nesting the pattern inside the original visualization to preserve context, as shown in Figure 9. This approach has two potential problems. First, the link might have a small area and cannot display the pattern satisfactorily. This problem can be resolved through automatic zooming to give the new branching pattern more space. The second problem is harder to address. Since the height of each link represents the average time elapsed or number of events of the subsequences, when we compute branching patterns on these subsequences in the link, some of the links in the new pattern will have a much greater average and outgrow the original link height. Our final design addresses these two problem through two-stage animated transitions: 1) zoom in so that the link of interest expands to occupy the entire canvas, and 2) display the newly computed branching pattern in the canvas. Figure 10 shows the branching pattern for the "checkoutLoaded ⟶ validateOrder" link in Figure 6a. We render all the ancestor nodes that lead to the current view. Users can navigate back to the overview via a back button on the top left corner.

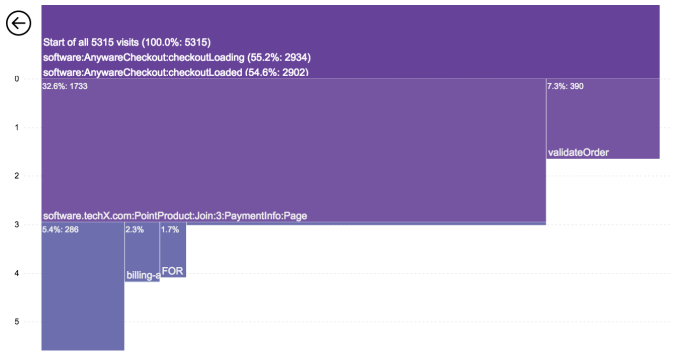


Figure 10: The branching pattern for the "checkoutLoaded ⟶ validateOrder" link in Figure 6a

	Dataset1	Dataset2	Dataset3
No. of sequences	5,315	60,553	301,360
No. of events	120,777	1,491,653	5,517,395
No. of unique events	4,209	36	107,707
avg sequence length	22.73	24.63	18.31
max sequence length	186	3,174	2,500
domain	web clickstreams	app logs	web clickstreams

Table 1: Three datasets with varying size and complexity used in our evaluation study

5. Evaluation

To evaluate our approach, we compare CoreFlowVis to the system by Liu et al. [LWD*17], which offers an interactive visual summary based on frequent sequential pattern mining (FSPM). FSPM is the closest work to this research: the goal of both works is to support interactive analysis of large event sequence data by providing an automatically generated overview. We provide benchmark results for the computation time for pattern extraction and qualitatively compare the interpretability of the extracted patterns.

Table 1 describes the three real-world datasets we use in this evaluation. These datasets vary in size and complexity. The number of sequences range from thousands to hundreds of thousands, and the number of events range from hundreds of thousands to millions. We also vary the minimum support paramater from 5% to 20% in computing the patterns.

5.1. Scalability

We implement the CoreFlow technique in Java. For the FSPM approach, we use the Java implementation of the vertical maximal sequential pattern mining algorithm [FVWGT14] in the SPMF library [FVGG*14]. Using the pattern pruning technique described in [LWD*17], we set the threshold value to 0.85. We run the tests on a quad-core 2.7 GHz MacBook Pro (OS X 10.11.5) with per-core 256K L2 caches, shared 6MB L3 cache and 16GB RAM. The Java Runtime version is 1.8.0 and we set the maximum heap size to be 8192 MB.

Figure 11 shows the benchmarking results. While the time to

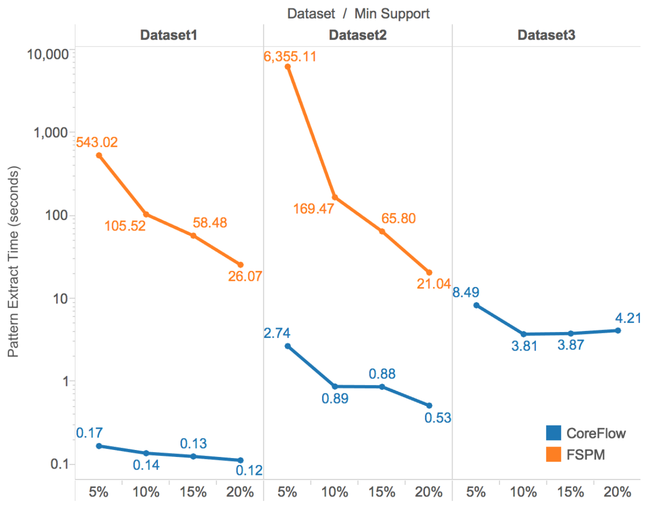


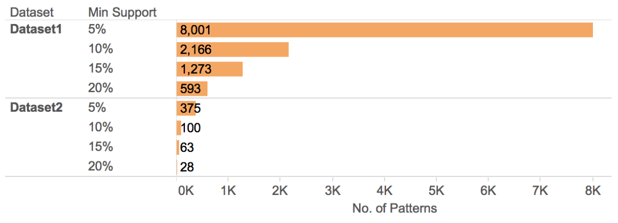
Figure 11: CoreFlow scales better to larger datasets and consumes less computation time. Due to memory limitation, we could not compute patterns for Dataset3 using the FSPM approach. To better display the labels and avoid overlap, we use a log scale for the y axis representing computation time.

compute sequential patterns with FSPM decreases with increasing minimum support, the computation time is not well suited for interactive analysis. Due to memory limitations, the FSPM implementation cannot handle Dataset3, which contains more than 300K sequences and 5.5M events. CoreFlow clearly scales better to larger datasets and consumes less computation time. CoreFlow is able to extract branching patterns from millions of events in a few seconds.

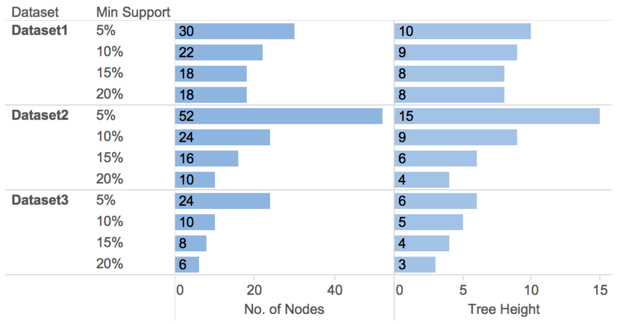
5.2. Qualitative Comparison of Pattern Interpretability

Computational time is an objective measure of the two techniques. More importantly, we would like to know if the extracted patterns are usable by human analysts. Figure 12(a) shows statistics about the sequential patterns the FSPM approach extracted. In many cases, the number of patterns is still overwhelming. Figure 12(b) shows the number of nodes in the branching patterns extracted by CoreFlow. The size of the pattern is more manageable, although with a concise overview, certain important events may be overlooked.

To better gauge the interpretability of the patterns, we focus on the patterns for Dataset 1 with minimum support set to 20%. Figure 13 shows the patterns discovered by FSPM and CoreFlow. While Dataset 1 contains one month of web clickstream data for a commercial website, Figure 13 shows the results of a query for visitors who purchased a product named *XPR* from the website. Figure 13(b) shows an overview that is consistent with this premise and provides summary statistics on these visitors' behavior. For example, a significant part of the traffic (34.8%) successfully goes through the checkout process, where visitors validate and submit their orders (highlighted in orange contour). Another 14.8% of the visitors manage their purchased plans (highlighted in green contour). On average, it takes fewer events to manage plans than to make purchases. Figure 13(a) shows the top 12 patterns (in terms of support)



(a) The number of sequential patterns generated using the FSPM approach. We could not compute patterns for Dataset3 due to memory limitation



Dataset	Min Support	No. of Nodes	Tree Height
Dataset1	5%	30	10
	10%	22	9
	15%	18	8
	20%	18	8
Dataset2	5%	52	15
	10%	24	9
	15%	16	6
	20%	10	4
Dataset3	5%	24	6
	10%	10	5
	15%	8	4
	20%	6	3

(b) Statistics of branching patterns generated by CoreFlow

Figure 12: Patterns generated by FSPM and CoreFlow

mined by the FSPM approach. The visualization shows patterns with significant overlap (e.g. the first and sixth pattern), and it is not evident how the patterns relate to each other. The visualization shows patterns related to the checkout process (annotated in red) as well as some other events that are not visible in the CoreFlow overview (e.g. “PaymentInfo” in the 9th pattern). In CoreFlow, the “PaymentInfo” becomes visible when we drill down to the “checkoutLoaded $\longrightarrow$ validateOrder” link (Figure 10). One may argue that CoreFlow does not show as much information as the FSPM approach; a counter argument is that CoreFlow is able to prioritize and organize information into meaningful hierarchies.

6. Case Studies: Meaningfulness and Usefulness

While our approach is more scalable and offers improved interpretability, it remains unclear if the branching patterns visualized in CoreFlowVis are meaningful and useful in real-world analysis situations. We decide to evaluate the utility of branching patterns using a case study approach, since there are no established metrics to evaluate pattern quality, and there are too few domain experts to do a controlled study. We worked with three analysts over a period of three years, where we conducted multiple discussion sessions with each analyst, showing them the visualization, guiding them in using the system to explore their data, soliciting their feedback and iterating on the system and interface design.

We report qualitative feedback in this section and focus on two issues. First, CoreFlow uses a frequency-based ranking function to choose milestone events. Is frequency a useful metric for event importance? To answer this question, we use the analysts’ domain knowledge as a baseline to evaluate the patterns. These analysts are domain experts who have worked on their data for years before CoreFlow was invented. They know what is important and their knowledge serves as a reasonable baseline. Second, we want to know if the analysts understand the visualizations and if they are able to discover useful insights using CoreFlowVis. The supplemental materials contain screenshots for each case study.

The results from these three case studies show that frequent events do correspond to meaningful milestones for certain datasets and domains. In particular, web clickstreams seem to be an ideal use case for CoreFlow (case 1). In such datasets, the number of unique event is large (e.g. more than 4000), each event is a page URL, which records both the action as well as state information (i.e. where a visitor is at a particular step in his journey). In contrast, it may not be appropriate to directly apply CoreFlow to application logs (case 2), where the number of unique events is relatively small (less than 50), and the events do not have state information (i.e. we have no information on the state of the document after each action).

6.1. Case 1: Analyze Visitor Paths in Web Clickstreams

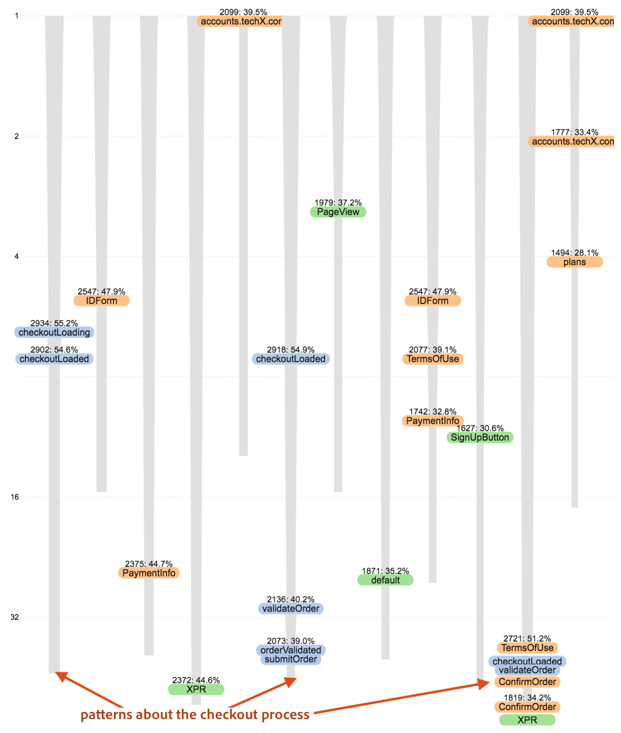
Background: Brian is an analyst for the marketing team at a large company. His job involves understanding customers’ journeys on his company’s website. He performs analysis and prepares reports for his manager and company executives. He is interested in examining major traffic flow patterns, understanding how the traffic flows after certain key pages, and reviewing the website checkout process to understand where visitors drop off. Currently, Brian uses multiple tools to do analysis. None of them support the customer journey analysis he is interested in.

Method: The CoreFlow research team has been interacting with Brian for three years, meeting monthly to discuss how to help his work. Initially, the team was learning about Brian’s work and his tasks. Over time, he helped the team iterate on data mining algorithms and system design. With CoreFlowVis, Brian has tried to analyze six datasets with varying size from different time periods.

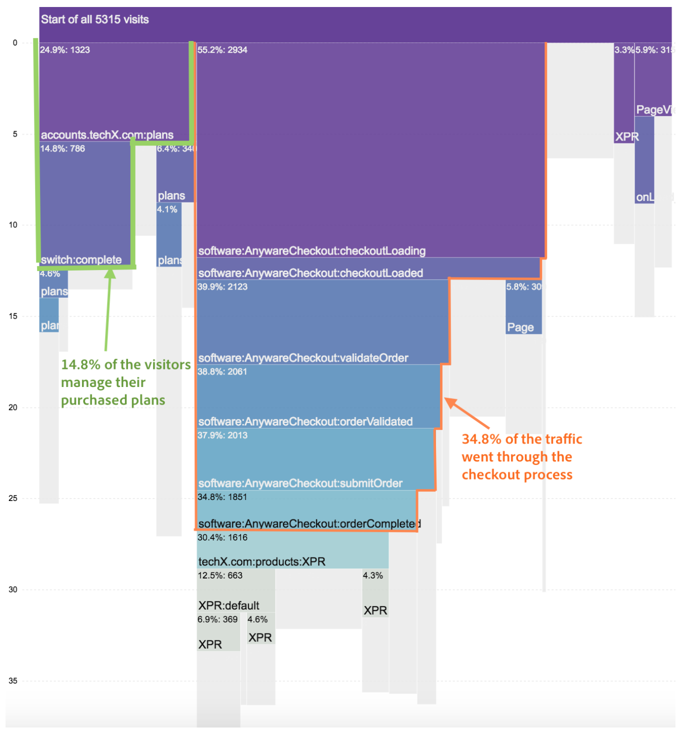
Analysis Process: When first working with CoreFlowVis, Brian thought that starting with as much data as possible was a good idea. He queried for all events on the company’s website for a period of 6 months resulting in a dataset of more than 300K sequences and 5 million events. Unfortunately, he found that there were no frequent patterns for more than 40% of the sequences, indicating that the sequences were very heterogeneous. For the remaining sequences, the help page appears many times as the milestone. Brian was not interested in help pages, so he decided to focus on sequences landing on a portion of the website describing a particular product.

Pattern Quality: With the sharpened focus on the input data, CoreFlowVis shows visualizations that aligned well with Brian’s knowledge. Brian had defined a funnel of important pages previously in other tools, and he was amazed that CoreFlow was able to automatically identify these pages as milestones. He commented “this is perfect” as the visualization also showed frequent pages visited outside the funnel and after the last page in the funnel.

Insights Discovered: The icicle plot helped Brian find a new user segment that he wasn’t tracking already. By grouping all sequences by the first event in the sequence, Brian was able to isolate existing



(a) Top 12 sequential patterns generated by the FSPM approach. Each node represents a milestone event, colored by event category. The line width represents the pattern's support.



(b) Branching pattern extracted by the CoreFlow approach. The width of the partitions in this icicle plot represents the number of sequences.

Figure 13: Comparison of patterns extracted by FSPM and CoreFlow. The vertical scale represents the average number of events.

customers who were getting to the website from a link inside of the product. It turned out that 25% of the traffic was from this group of users and understanding their behavior was critical to understanding the traffic on the website.

Brian was also able to see that 25% of visitors were switching from one payment plan to another. This finding helped him confirm what the company was seeing in the sales department.

Feedback: Brian was pleased to see that CoreFlowVis provided answers to many of his questions within a unified view, and commented on its "potential to be very powerful if we continue to fully optimize it to our needs. I am excited to see where we can take it". He mentioned that the tooltip showing top events in a link was definitely helpful. Initially, he liked the tree visualization because it was easy to follow the links. He felt that the icicle plot was hard to interpret. As he became more familiar with the icicle plot, he saw its utility and used it as his default view.

Takeaways: Contrary to the belief that we should gather "big data" for insight discovery, we find that segmenting a large dataset in a meaningful way is a prerequisite for focused exploration using CoreFlowVis.

6.2. Case 2: Understand Workflows in Application Log Data

Background: Stephanie is a product analyst working on several applications. She heard about CoreFlowVis from a colleague and asked to try the system using her datasets. Stephanie is interested in understanding how customers use her applications. For example, what are the most common workflows and application features? And, what do people do after they publish and share?

Method: We had three 1-hour meetings with Stephanie. In the first meeting, we demonstrated CoreFlowVis and discussed input data format. She then prepared a dataset, which we discussed and analyzed together in the second and third meetings.

Analysis Process: Stephanie's dataset contained approximately 1.5 million events, representing one-month of user usage data for an application for video editing. She filtered out events related to login and account management, because she was not interested in events not directly related to the actual workflow. Each log event had four attributes: timestamp, user ID, project ID and event name.

Stephanie first grouped events into sequences by project ID, resulting in 38K sequences. Each sequence represents user activities on a project. In CoreFlowVis, she set the vertical axis to represent actual timestamps instead of number of event, and the visualization showed that some users spent up to a week working on a project, potentially intermittently. The most frequent pattern in the visualization was not very interesting: users were loading projects periodically, suggesting that they worked on the video projects in multiple sessions. She could drill down into individual links to see their workflows during each of the sessions, but the process could be cumbersome.

Stephanie suggested breaking down the sequences into sessions: if the temporal gap between adjacent events was greater than 20 minutes, it was safe to assume a new session had started. Dividing project sequences into sessions generated 60K sequences in total.

Insights Discovered: Visualizations based on session sequences offered interesting information. Using the funnel query, Stephanie saw that it took about 12 minutes for users to begin publishing and sharing their videos, and that the most frequent user workflow was choosing images to embed in their videos. The repeated “choosing image” events in the pattern might seem boring at first, but when Stephanie hovered over each of the links, she saw that users were focusing on different aspects of videoing editing over time. Initially, they were changing video themes and changing layout, later on they performed more text editing and page sequence restructuring more. Stephanie thought the development team could use this insight to more intelligently guide users in the video creation process. She was also surprised to see that after the publishing began, multiple sharing actions were performed (she was expecting one sharing action).

Pattern Quality: The patterns with frequent events as milestones are not very meaningful. Stephanie had to hover over the links to find interesting information. We think CoreFlowVis did not totally succeed in this case for two reasons. First, an event like “open project" may be more frequent than “draw rectangle”, but both events are common features in the tool. Using frequency as the sole metric resulted in picking “open project” over “draw rectangle”, and this could be limiting. Second, in application log data, a meaningful milestone is often a task, not a single event. For example, a task might be creating an image from a screenshot, which consists of multiple operations (events). Extracting single events is too low-level, and CoreFlow needs to be extended to extract frequent event groups.

Feedback: Although the patterns were not very meaningful, Stephanie was still very excited and found CoreFlowVis to be more useful than the tools provided by a commercial application she was currently using. She thought CoreFlowVis had the potential to evolve into a tool that could be a regular part of her work.

Takeaways: In case study 1, we see that the number of sequences affects data heterogeneity and the usefulness of branching patterns. Case study 2 suggests that the length of sequences and how a sequence is defined determine the quality of branching patterns as well. In addition to showing the important events and how traffic branches out across these events, CoreFlowVis also provides a way to segment the sequences so that comparative analysis of patterns within the links can reveal insights on the evolution of user behavior over time.

6.3. Case 3: Compare First Marketing Touch Points

Background: Jason oversees business strategy for the marketing team. His current work focuses on understanding the effectiveness of digital marketing on different media platforms. The marketing team at a company reaches out to potential users using multiple media platforms including emails, promotional materials on the company’s website and advertisements on third-party websites. Potential users may experience a sequence of “media touch points” before they make a purchase. These “media touch points” events include opening an email, seeing a promotion when opening a trial version of the software, or seeing an advertisement on a news website, search engine or social media.

Method: Like Brian, Jason is an analyst we have met regularly for three years. We met with him once every two months and conducted in-depth interviews to learn about his work and analytic tasks.

Analysis Process: Jason’s team is working on boosting sales for one of his company’s products. He collected 14787 sequences with 40K unique events and 1.9 million events in total, and we imported them into CoreFlowVis. The initial overview showed that the marketing efforts on different media platforms eventually led 90% of users to the homepage of the product, and on average it took about 30 days to achieve this result.

Jason thought it was useful to know the average length of time to lead visitors to the homepage, and he asked if we could further group the sequences by the first events. After we added this “group by entry event” feature in CoreFlowVis, he was excited because he could compare customer journeys by the first media touch point.

Insights Discovered: Using CoreFlowVis, Jason was able to determine how the media touch point related to visiting the product webpage. For example he found that visitors whose first touch point was using a related product took the longest to get to the product website (40 days). In contrast, marketing emails led users to the product within on average 30 days.

Pattern Quality: This dataset has an interesting characteristic: there was a highly frequent event (“landing on product website”) occurring in most of the the sequences, and that event happened very late in many sequences. We were concerned if this could be a problem, because CoreFlowVis would not show events that happened before “landing on product website” by default. It turned out that the visualization showed exactly what the analyst wanted to see: how long it took people to come to the product website. In this case, using frequency as a metric to choose milestone event proved to be a reasonable strategy again.

Feedback: Jason presented these insights to his manager and team, and the feedback was very positive. Two features of CoreFlowVis were especially helpful: being able to see an overview of the complete sequences, and comparing the effects of first media touch points. The team also commented that they wanted to further perform comparative analysis on different user groups in the datasets.

Takeaways: Comparative analysis emerged as a crucial task in this case study. There are many different ways to segment data and perform comparisons. Previous research such as MatrixWave [ZLD*15] and HVHT [MSD*16] provided valuable insights on this research direction, and we intend to incorporate more powerful comparative analytics such as hypothesis testing into CoreFlowVis in future work.

7. Discussion

While we have provided evidence demonstrating the usefulness of our approach, CoreFlowVis is not a panacea for temporal event sequence analysis: it is unrealistic to assume that we can feed any dataset into CoreFlowVis and insights will pop up. CoreFlow often needs to be used in conjunction with analytic sharpening strategies [DSP*16] such as goal-driven record extracting (case study 1) and temporal folding (case study 2). However, once we have tuned analytical focus, CoreFlowVis is able to provide insightful visualizations with and without user-defined milestone events, as in the cases of clickstreams and multi-media touch points analyses.

One unexpected finding from the case studies is that users did not like to drill down to extract branching subpatterns. Based on the feedback, drilling down changes the visualization too much, and users find it tedious to navigate across multiple visualizations at different levels of granularity. They often hovered over the links to see top events in a link and used that information to decide if drilling down made sense. One analyst commented, "Drilling down is useful, but it would be nice if I can see the information without having to do drill down". Future work to improve this workflow includes rendering information about subsequences in a link without cluttering the visualization and precomputing subpatterns to recommend interesting links for further exploration.

Displaying event labels in the visualizations remains a challenge. Long event names result in overlap and clutter. In the icicle plot, we trim the event names to fit the size of the rectangular links. Consequently, some event names are not very readable. Techniques that automatically shorten event names while preserving their readability will be an interesting direction to explore further.

Finally, this research suggests the need of a larger research agenda on evaluating visual summaries of event sequences. The case studies show that the usefulness of the patterns can be highly contextual and domain-dependent. Doing case studies is one way to evaluate pattern utility, but this approach is time consuming and not very scalable. In addition, comparative evaluation can be difficult using a case study approach. When we worked with the analysts, we wanted to have a deeper understanding on how the choice of ranking functions affects pattern quality. We generated branching patterns using the frequency-based method described in this paper, as well as using alternative methods that characterize event importance with weighted averages of frequency and position. When we informally asked for feedback on these branching patterns, the analysts remarked that it was difficult to rank the patterns in terms of usefulness and meaningfulness. These experiences suggest that establishing robust metrics or models to validate pattern quality can be an important yet challenging research direction to pursue.

8. Conclusion

In this paper, we propose the idea of using *branching patterns* for temporal event sequence analysis. We present a novel technique, CoreFlow, that extracts branching patterns by recursively applying the three-step *Rank-Divide-Trim* procedure. Compared with state of the art in sequential pattern mining, CoreFlow is more scalable and able to compute patterns for millions of events in seconds. We have designed and developed a system based on CoreFlow. Feedback from analysts confirms the usefulness of our approach in certain domains, and points to the challenge of evaluating pattern quality beyond a case study approach.

References

[Bah03] BAHAR E.: Clospan: Mining: Closed sequential patterns in large datasets. 1

[BJL02] BUCHHEIM C., JÜNGER M., LEIPERT S.: Improving walker's algorithm to run in linear time. In *International Symposium on Graph Drawing* (2002), Springer, pp. 344–353. 5

[BOH11] BOSTOCK M., OGIEVETSKY V., HEER J.: D^3: data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on 17*, 12 (2011), 2301–2309. 5

[DSP*16] DU F., SHNEIDERMAN B., PLAISANT C., MALIK S., PERER A.: Coping with volume and variety in temporal event sequences: Strategies for sharpening analytic focus. *IEEE transactions on visualization and computer graphics* (2016). 11

[FVGG*14] FOURNIER-VIGER P., GOMARIZ A., GUENICHE T., SOLTANI A., WU C.-W., TSENG V. S.: SPMF: a java open-source pattern mining library. *The Journal of Machine Learning Research 15*, 1 (2014), 3389–3393. 7

[FVWGT14] FOURNIER-VIGER P., WU C.-W., GOMARIZ A., TSENG V. S.: VMSP: Efficient vertical mining of maximal sequential patterns. In *Advances in Artificial Intelligence*. Springer, 2014, pp. 83–94. 7

[GS14] GOTZ D., STAVROPOULOS H.: Decisionflow: Visual analytics for high-dimensional temporal event sequence data. *IEEE transactions on visualization and computer graphics 20*, 12 (2014), 1783–1792. 2, 6

[KL83] KRUSKAL J. B., LANDWEHR J. M.: Icicle plots: Better displays for hierarchical clustering. *The American Statistician 37*, 2 (1983), 162–168. 1, 2

[KSS05] KEIM D. A., SCHNEIDEWIND J., SIPS M.: Fp-viz: Visual frequent pattern mining. 2

[KVP16] KWON B. C., VERMA J., PERER A.: Peekquence: Visual analytics for event sequence data. In *ACM SIGKDD 2016 Workshop on Interactive Data Exploration and Analytics* (2016). 1

[LRTM07] LAM H., RUSSELL D., TANG D., MUNZNER T.: Session viewer: Visual exploratory analysis of web session logs. In *Visual Analytics Science and Technology, 2007. VAST 2007. IEEE Symposium on* (Oct 2007), pp. 147–154. 1, 6

[LWD*17] LIU Z., WANG Y., DONTCHEVA M., HOFFMAN M., WALKER S., WILSON A.: Patterns and sequences: Interactive exploration of clickstreams to understand common visitor paths. *IEEE Transactions on Visualization and Computer Graphics 23*, 01 (Janaury 2017). 1, 2, 7

[MLL*13] MONROE M., LAN R., LEE H., PLAISANT C., SHNEIDERMAN B.: Temporal event sequence simplification. *Visualization and Computer Graphics, IEEE Transactions on 19*, 12 (2013), 2227–2236. 1, 2

[MSD*16] MALIK S., SHNEIDERMAN B., DU F., PLAISANT C., BJARNADOTTIR M.: High-volume hypothesis testing: Systematic exploration of event sequence comparisons. *ACM Transactions on Interactive Intelligent Systems 6*, 1 (Mar. 2016), 9:1–9:23. 10

[PW14] PERER A., WANG F.: Frequence: interactive mining and visualization of temporal frequent event sequences. In *Proceedings of the 19th international conference on Intelligent User Interfaces* (2014), ACM, pp. 153–162. 1

[SZ00] STASKO J., ZHANG E.: Focus+ context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on* (2000), IEEE, pp. 57–65. 2

[WGGP*11] WONGSUPHASAWAT K., GUERRA GÓMEZ J. A., PLAISANT C., WANG T. D., TAIEB-MAIMON M., SHNEIDERMAN B.:

Lifeflow: visualizing an overview of event sequences. In *Proceedings of the SIGCHI conference on human factors in computing systems* (2011), ACM, pp. 1747–1756. 2, 5

[WSSM12] WEI J., SHEN Z., SUNDARESAN N., MA K.-L.: Visual cluster exploration of web clickstream data. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on* (2012), IEEE, pp. 3–12. 1

[WV08] WATTENBERG M., VIÉGAS F. B.: The word tree, an interactive visual concordance. *IEEE transactions on visualization and computer graphics 14*, 6 (2008), 1221–1228. 3

[WZT*16] WANG G., ZHANG X., TANG S., ZHENG H., ZHAO B. Y.: Unsupervised clickstream clustering for user behavior analysis. In *SIGCHI Conference on Human Factors in Computing Systems* (2016). 1, 2

[ZBS16] ZHANG X., BROWN H.-F., SHANKAR A.: Data-driven personas: Constructing archetypal users with clickstreams and user telemetry. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (2016), ACM, pp. 5350–5359. 2, 3

[ZDF15] ZGRAGGEN E., DRUCKER S. M., FISHER D.: (s|qu)eries: Visual regular expressions for querying and exploring event sequences. *Proceedings of CHI 2015* (2015). 1, 6

[ZLD*15] ZHAO J., LIU Z., DONTCHEVA M., HERTZMANN A., WILSON A.: Matrixwave: Visual comparison of event sequence data. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (2015), ACM, pp. 259–268. 1, 10