Eurographics Conference on Visualization (EuroVis) 2019
M. Gleicher, H. Leitte, and I. Viola
(Guest Editors)

Volume 38 (2019), Number 3

Segmentifier: Interactive Refinement of Clickstream Data

K. Dextras-Romagnino† and T. Munzner

University of British Columbia, Canada

Abstract

Clickstream data has the potential to provide insights into e-commerce consumer behavior, but previous techniques fall short of handling the scale and complexity of real-world datasets because they require relatively clean and small input. We present Segmentifier, a novel visual analytics interface that supports an iterative process of refining collections of action sequences into meaningful segments. We present task and data abstractions for clickstream data analysis, leading to a high-level model built around an iterative view-refine-record loop with outcomes of conclude with an answer, export segment for further analysis in downstream tools, or abandon the question for a more fruitful analysis path. Segmentifier supports fast and fluid refinement of segments through tightly coupled visual encoding and interaction with a rich set of views that show evocative derived attributes for segments, sequences, and actions in addition to underlying raw sequences. These views support fast and fluid refinement of segments through filtering and partitioning attribute ranges. Interactive visual queries on custom action sequences are aggregated according to a three-level hierarchy. Segmentifier features a detailed glyph-based visual history of the automatically recorded analysis process showing the provenance of each segment as an analysis path of attribute constraints. We demonstrate the effectiveness of our approach through a usage scenario with real-world data and a case study documenting the insights gained by a corporate e-commerce analyst.

1. Introduction

Companies engaged in e-commerce are recording every action consumers take on websites, resulting in rich clickstream datasets with the potential to provide insight into consumer behavior that could be used to improve user experience and increase corporate revenue. Extracting meaningful signals from these huge and noisy datasets is a difficult analysis problem. Moreover, the people tasked with understanding clickstream data often have very limited time for and minimal training in data analysis.

In practice, clickstream analysts frequently turn to third party platforms such as Google Analytics [GA] that focus on providing aggregated high-level metrics to give an overall picture of website performance, but do not provide support for fully drilling down into the details of consumer behavior.

In the research literature, many techniques have been proposed for finding patterns from logged sequences of data, ranging from identifying similarly-behaving groups of users through clustering to extracting common behavior through pattern mining to supporting interactive visual queries. While these techniques often yield excellent results with clean and relatively small datasets, our initial experiments yielded unsatisfactory results when applied to datasets faced by our collaborators engaged in real-world corporate clickstream data analysis. We realized the need for a tool that helps analysts partition these messy datasets into cleaner and more manageable **segments**, namely subsets of sequences, that contain a reasonable ratio of task-relevant sequences to extraneous ones. In some cases the expected outcome of using the tool would be to export segments for downstream analysis with these kinds of tools. We also envisioned the case where the creation and refinement process could reveal enough information about the segment to directly resolve analyst questions.

The idea that real-world datasets are extremely noisy and need to be iteratively cleaned and refined before some desired analysis technique can be conducted is actively researched under the name *data wrangling* [KPHH11]. Existing models of the data analysis process also touch on this idea. For example Grolemund and Wickham propose import and tidy phases followed by a transform-visualize-model loop [GW14]. A more specific discussion of clickstream data analysis from Sarikaya et al. points out the need for simplifying clickstream data for effective downstream analysis, noting the difficulties that arise from the clutter of repeated events, ambiguous events, useless events, and irrelevant sequences that do not pertain to a current analysis task [SZD*16]. They call for an exploratory tool that supports an iterative process between preprocessing data and viewing the results with a graphical history to record steps.

We propose Segmentifier as an answer to this call. One contribution of our work is a thorough characterization of task and data abstractions for clickstream data analysis, culminating in a high-

† now at Kabam Games, Inc.

level analysis model specific to clickstream data analysis. It is built around an iterative view-refine-record loop, where the outcome can be to conclude an analysis path with an answer, to export segments for downstream data analysis, or to abandon a question to pursue a potentially more fruitful analysis path. The Segmentifier interface is a concrete instantiation of this model. It has a rich set of tightly coupled views that show evocative derived attributes for different kinds of data: segments, sequences, and actions. It supports filtering and partitioning through visual queries for both quantitative attributes and custom sequences of events, which are aggregated according to a three-level hierarchy. It features a detailed glyph-based visual history of the automatically recorded refinement process showing the provenance of each segment as constraints on attributes, shown as a visual history tree. We show evidence for the utility of Segmentifier through a detailed usage scenario and a case study showcasing insights gained quickly by a clickstream data analyst at an e-commerce company.

2. Clickstream Data and Tasks

We describe our research process and describe the results of our requirements analysis first in domain-specific terms and then as low-level task abstractions. We then present a high-level model of the Segmentifier analysis process, followed by our data abstraction.

2.1. Process

We followed the design study methodology of Sedlmair et al. [SMM12]. We began with the *winnow*, *cast*, and *discover* stages over a period of 5 months. We conducted exploratory and followup interviews with 12 employees who conducted clickstream data analysis belonging to 3 different internal teams of an e-commerce middleware company. Our task elicitation yielded three potential tasks: analyzing user behavior based only on website clickstreams, the impact of external messages sent to consumers on their website behavior, and characterizing load times for the website. We chose the former as a manageable scope after considering stakeholder availability, data availability, and the other relevant pitfalls [SMM12]. The iterative *design* and *implement* stages lasted 11 months, with a final *reflect* stage of 3 months.

2.2. E-commerce Clickstream Analysis Goals

We focus specifically on e-commerce clickstream data recorded from websites whose main functionality is to generate revenue by selling products to consumers. Clickstream data analysis is intended to provide insight into consumer activities logged as sequences of user actions on websites. Typical metrics of success are to increase *traffic* (number of users on a site), reduce *abandonment* (number of users leaving the site), increase consumer *engagement* (time users spend on the site and chances that a user returns to the site), and increase *conversion rate* (odds of purchases).

An event sequence is any multivariate sequence of timed events; clickstream data is a special case of this data type where the individual events are categorical **actions** performed by a user interacting with a website, such as a page visit. Each action is logged with a single timestamp, so actions do not overlap. A **sequence** is a list of time-ordered actions, indicating the activity of a specific consumer over some specific time interval. Clickstream datasets, which track every user action, can contain millions of sequences in real-world logs, a scale much larger than many other types of event sequence data [LWD*17, SZD*16]. Moreover, they are inherently extremely noisy with high variability between sequences, so very few are identical.

Our requirements elicitation process yielded four main goals for data analysts seeking to understand consumer activity on websites: G1) identify trends in common between consumers who successfully make purchases and optimize the site with respect to their activities; G2) identify problems or painful paths and spend resources to fix or improve them; G3) identify groups of consumers to target with personalized email; G4) identify baseline values for standard benchmark metrics (such as number of users per day).

Examples of lower-level questions derived from these high-level goals are: *What pages do users most commonly exit on? How many users purchase? How many bounce (exit after viewing one page)? How many users make it through the purchasing funnel? Where do they drop out? What different types of buying behaviors are there?*

The analysts have considerable implicit knowledge about what consumer activities are *expected* (users add to cart before purchasing) vs. *unexpected* (sudden halt in purchasing), and *favorable* (lead to a purchase) vs. *unfavorable* (lead to abandonment). All combinations are of interest; for example, users dropping out of the purchasing funnel is both expected and unfavorable.

Our interviews with analysts revealed that they have only a brief time to conduct analysis and an even briefer time window for reporting their findings to clients through very short presentations. Also, many people in this job role have no or minimal skills in programming and statistics. A design requirement for Segmentifier is to cater to these limits of skills and available time.

2.3. Low-Level Task Abstraction

All the domain-specific questions in Section 2.2 can be abstracted as questions about subsets of sequences, such as: how large is the subset in absolute terms or relative to some other subset; does one subset have different properties than another; can a set of sequences be divided into meaningful subsets according to some property?

We use the term **segment** to mean any subset of sequences. We define **attributes** as properties of segments, sequences, or actions that are either directly provided in the logged data (actions and their times) or can be derived from it (number of actions in a sequence, time range across all actions in all sequences within a segment).

We formalize the idea of consumer activity by defining a consumer **behavior** as a set of constraints on these attributes. There is a one-to-one mapping between a behavior and the segment that contains all of the sequences matching those attribute constraints. For example, the behavior of *start before 8am and purchase* corresponds to constraints that a purchase action must exist in each sequence and the timestamp attribute of the first action in the sequence should be before 8am.

The space of possible attribute constraint sets is truly enormous.

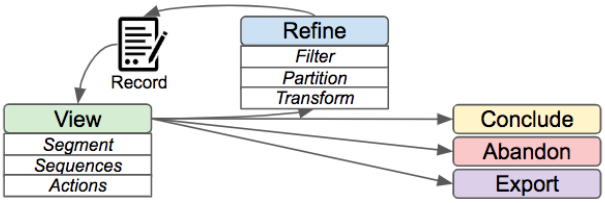

Figure 1: *The high-level Segmentifier analysis model features an iterative view-refine-record loop to create segments. Analysis paths can conclude with an answer, end by abandoning the segment, or result in exporting the segment for further analysis.*

The challenge is to translate intuition about behavior into a set of attribute constraints corresponding to a segment that simultaneously a) is semantically interpretable to the analyst, b) matches sequences of actual consumer activity found in the logs, and c) limits the variance of sequences in the segment enough to meaningfully distinguish it from other segments. For example, the behavior *visit a page on a Monday* is a very loose constraint with so many unique matching sequences that the segment is unlikely to be distinctive enough to lead to insight.

We were able to consolidate all of the domain-specific questions into four fundamental abstract tasks (with concrete examples):

- **T1: Identify** *Find some set of sequences that constitutes interesting behavior.* (consumers in loyalty program browse longer)
- **T2: Drilldown** *Distinguish more specific behaviors to further partition a segment previously defined by looser constraints.* (check if purchasers fall into natural groups by time of day)
- **T3: Frequency** *Determine how many sequences are in the segment defined by behavior X.* (check ratio of bouncers to non-bouncers)
- **T4: Ordering** within sequence: *Match if action subsequence X occurs before (or after) action subsequence Y in a sequence.* (verify that all users add to cart before purchasing)

To support all four of these low-level tasks, we propose a high-level model for segment-based analysis of clickstream data.

2.4. High-Level Segmentifier Analysis Model

The high-level Segmentifier analysis model is shown in Figure 1. It is structured around five choices for a data analyst: **view**, **refine**, **export**, **abandon**, and **conclude**. The model also includes a sixth **record** item that is automatically carried out after each refine choice. In the Segmentifier model, many segments are generated and analyzed through an iterative exploration process. Considered as a whole, this interactive analysis model provides design requirements for a visual analytics system that supports all four of our abstract tasks; there is no direct mapping from a specific low-level task to a specific choice point in this high-level model.

The exploration process begins with a view of an initial segment that contains all sequences in the dataset. Segments can be refined by the analyst to create more appropriate segments for a task by either filtering, partitioning, or transforming their sequences. Every refinement step leads to one or more new segments, which can in turn be viewed and further refined. Each step of this analysis process is automatically recorded to help analysts keep track of the many questions and hypotheses that are generated during an exploratory analysis session. These recorded steps provide enough provenance information that any segment can be revisited later to understand the full context of the analysis path leading to it. One benefit of this architecture is that it mitigates the *cold-start problem* that bedevils many previous systems, where an analyst is confronted with a completely blank view because a query must be run before seeing any results.

The model supports flowing between the abstract tasks opportunistically based on what the visual data analysis reveals. Exploring one question often spurs follow-up or entirely new questions. The entire view-refine-record analysis loop could stop when the analyst simply runs out of time, or when the flow of questions about a dataset runs dry. The analysis path for a particular question ends with a conclude-abandon-export choice. The analyst could conclude with a segment by finding a satisfactory answer within Segmentifier itself, abandon a segment as a dead end to give up on that question because there is no obvious pathway for further refinement, or export a satisfactory segment for further analysis in a more specialized tool. The Segmentifier analysis model includes export for downstream analysis as a first-class citizen to emphasize its design goal of bridging to existing techniques, such as clustering and pattern mining, that require relatively clean input data to work effectively. For example, to achieve useful results, pattern mining to discover common sequences of events requires as input a small number of sequences with low variability.

The Segmentifier analysis model explicitly requires views showing three aspects of clickstream data simultaneously: segment-oriented views to show the distributions of attributes for the entire segment, sequence-oriented views showing the details of individual sequences contained within the segment, and action-oriented views that focus on the order that actions occur within the segment's sequences and their frequency according to action types. The combination of these three views supports both directions of discovery about behaviors: from constraints to segments, and from segments to constraints. Sometimes analysts have an existing understanding of what constraints to specify (purchase action occurs) and create a segment to check if its attribute distributions meet their expectations. Sometimes analysts notice an interesting distribution of attributes within a segment and then check what choices of constraints led to that distinctive pattern. For example, splitting into segments by the day of the week might result in noticing that one of them has far fewer purchases, and tracing back would show it is the Wednesday one. It would be very difficult for analysts to translate their intuitions about consumer behavior to settings for attribute constraints without this kind of visual support. Inversely, the ability to dig into the data itself reveals consumer behavior that analysts did not previously understand.

The goal of the Segmentifier model is scalability. It provides a framework for the interactive visual refinement of extremely large clickstream datasets with an enormous level of variance between sequences into more useful lower-variance segments. These can either be successfully handed off to previously proposed approaches

for further processing, or may directly show analysts what they need to know within Segmentifier itself. It is designed to allow analysts with considerable domain knowledge but limited time and statistical expertise to quickly form and test hypotheses connecting logged consumer behavior to constraints on attributes. They find actionable and interpretable insights from understanding how specific choices of filtering and partitioning change the attribute distributions within segments, sequences, and actions. The ideas of supporting interactive filtering and partitioning, of showing the statistics of attribute distributions visually, and of combining viewing and refining are of course well established in previous work. The main technical novelty in Segmentifier lies in the integration of all aspects of our high-level model: the interlocking details of how to define informative segment, sequence, and action attributes and create tightly coupled views that allow fluid and understandable interactive refinement.

2.5. Data Abstraction

The difficult visual analytics design problem is how to connect the internal and implicit mental model that clickstream data analysts have about interesting consumer behaviors to viewable characteristics of segments, sequences, and actions. Our goal is to compute evocative derived attributes of the base data that reveal patterns, distributions, and outliers that align with their informal intuitions about interesting behavior, so that a series of constraints on these attributes would serve to define a satisfactory segment.

An **action** is a logged interaction of a consumer on a website, a **sequence** is a time-ordered list of actions from a particular consumer over some specific time interval, and a **segment** is any set of sequences. The base **action attributes** in the logs are a quantitative **timestamp**, a categorical *action type* (discussed further in Sections 4.2.2 and S1), and a categorical **clientid** identifier indicating the specific consumer. We derive **sequence attributes** from the action attributes within them, yielding the quantitative time range attributes of **sequence start**, **sequence end**, and **sequence duration**. We also derive **sequence action counts**, quantitative attributes storing the total number of actions, and per-type counts for each of the action types. In turn, we use all of this information to compute distributions over all constituent sequences as **segment attributes**, using an intermediate table with one row per sequence. We also compute action-based quantities for each segment by combining all constituent sequences together and counting the union of all actions contained within them, using a second intermediate table.

A **session** is a default time frame often used in e-commerce log analysis, where a single consumer's actions are grouped together until a gap of at least 30 minutes occurs. The original clickstream data is partitioned by these arbitrary session boundaries; we also concatenate all of these to derive per-client sequences that follow the behavior of a single consumer across longer periods of time.

We also require a dynamic data structure to support the *refine* and *record* choices of the Segmentifier model. The **analysis paths tree** has a node for each segment, and each node stores the simple quantitative attribute of **segment size**, the count of sequences it contains. The initial segment at the root of the tree contains all sequences, and the downstream segments become smaller as they are refined. Each segment node has associated with it the recorded information of exactly what *operator* was used to refine it, as discussed in Section 4.3; this derived data constitutes a full record of the analysis process. We use this tree to derive the quantitative attributes of **relative counts** of sequences at each node in the tree: the percentages within a segment compared to its direct predecessor in the tree, and compared to the root segment of all sequences.

3. Related Work

We frame the related work with respect to the Segmentifier analysis model, to walk through the nuances of our claim that no previous system encompasses all of its capabilities.

Post-Export: Specific Techniques A significant amount of research in this field proposes specific techniques for relatively clean and small sets of event sequences; we consider these to be useful for downstream analysis only after the *export* stage of our framework. Major categories of such techniques include clustering [WZT*16, WSSM12, GXZ*18, ZBS16], pattern mining [PW14, LKD*17, LWD*17], and cohort comparison [ZLD*15, MSD*16]. In all of these cases, the techniques do not handle the scale and complexity of real-world clickstream data. They are validated with input data such as manually refined datasets [WSSM12, ZLD*15], very small datasets [PW14, LWD*17], or explicitly indicate a requirement for sampling [LKD*17]. Segmentifier is designed to maximize the effectiveness of these techniques by allowing analysts to first refine the data to meet their requirements.

View Sequences: Event Sequence Visual Overviews A great deal of the previous work focuses on different ways to *view* raw sequences but does not provide mechanisms to *view* derived segment, sequence, or event attributes. Most earlier work simply displayed individual sequences, either along a horizontal timeline [AMTB05, KBK11] or using footstep graphs [hTKK04]. Later work introduced ways to show many sequences at the same time by either grouping them [BBD08] or using different visual encoding idioms such as networks [WHS*02, BB01], flow diagrams [PG13, WG12, Mui11],or icicle plots [WGGP*11]. Wang et al. [WPQ*08] extended some of this work by introducing interaction techniques such as aligning sequences. However, without any ability to view derived attributes or to refine the data, discovering insights is extremely difficult.

Refine: Visual Query Systems Visual query systems such as COQUITO [KPS16], (slqu)eries [ZDFD15], DecisionFlow [GS14], PatternFinder [FKSS06], and SparqlFilterFlow [HLBE14] emphasize the need to *refine* segments by allowing analysts to filter data. Although they *record* refinement history using analogous constructs to our analysis paths tree, they provide either a limited or nonexistent *view* of the derived segment attributes and they do not provide a *view* of the raw sequences at all. Analysts thus have trouble knowing how to refine and often take a guess–and–check approach. Moreover, these systems focus on filtering, whereas Segmentifier also provides ways to *refine* through partitioning.

Record: Graphical Histories We embrace the arguments in previous work that graphical histories to help analysts remember analysis paths are essential for any exploratory analysis [HMSA08]. In

the language of the Ragan et al. [RESC16] framework, Segmentifier provides *data provenance* for the purpose of *recall*, *replication*, and *action recovery* through our *Analysis Paths View*. Our *Analysis Paths View* is also analogous to the version tree in VisTrails [SFC07] that records the history of changes in workflows. Nguyen et al. [NXW*16] incorporate provenance for clickstream data but use *interaction provenance*, recording every user interaction, as opposed to Segmentifier which uses *data provenance*, recording how the data has been manipulated and changed.

View and Refine: Filtering Sequences To Segments Tools that emphasize the need to both iteratively *view* and *refine* data come closest to the functionality of Segmentifier. In addition to providing a view of both segment attributes and the sequences, they incorporate the additional functionality of filtering by derived ordinal or categorical segment attributes [SFCQ15, LPSH01] or filtering by derived event attributes [VJC09, SWSM12]. Wongsuphasawat et al. [WL14] incorporates both filtering options but in two different tools focusing on very specific tasks of comparing datasets and funnel analysis. PatViz [KBGE11] incorporates visual queries and multiple ways of viewing the data but is very specific to patent data.

The tools most close to Segmentifier incorporate versions of both filtering options and focus on exploring and simplifying event sequences. Session Viewer [LRTM07] has the ability to filter by derived segment and sequence attributes but only highlights sequences that follow custom event patterns. EventFlow [MLL*13] incorporates both filtering options and while it includes a view of the raw sequences, it provides very minimal derived attributes. It includes some derived sequences attributes but no derived event attributes as supported by our action hierarchy. Neither has the ability to *record* the analytic process. EventPad [CvW18] is the most similar work to our own. It fully incorporates both filtering options, but does suffer from the cold-start problem of starting with a blank slate. It allows analysts to manually choose to record segments they create, but does not automatically *record* all refinement steps. Its focus on regular expressions would be a poor match for clickstream analysts who do not have the time or training to grapple with these complex queries; Segmentifier is designed around a more restricted query scope suitable for non-programmers. These three tools have main views that focus on viewing raw sequences and provide only limited support for viewing derived segment, sequence, and event attributes. In contrast, the design of Segmentifier promotes these derived attributes to first-class citizens, a crucial factor in accommodating the scale of clickstream data.

4. Segmentifier Interface

The Segmentifier interface is a concrete instantiation of our high-level analysis model, providing meaningful views at all times as the cornerstone of fruitful exploration. It consists of three major views, as shown in Fig. 2. The *Segment Inspector* view on the right side is devoted to the *View* choice in the model, with side-by-side availability at all times of segment-level range information at the top, action-focused information in the middle, and base sequence details at the bottom. The *Operation Manager* view on the left tackles the *Refine* choice. The middle *Analysis Paths* view is gradually built up through the *Record* steps, and shows all of the segments; the selected one is shown in detail in the *Segment Inspector*. A key design feature of Segmentifier is the extensive linked interaction within and between all of these tightly coupled views.

4.1. Analysis Paths View

The middle *Analysis Paths* view, visible in Fig. 2B, allows the analyst to easily keep track of how all segments were created, showing the information saved at each *Record* step of the model. The visual encoding is a graphical history tree using two kinds of nodes: segment nodes are grey rectangles sized by their absolute sequence counts, and evocative glyphs concisely show *operations* used to create them in the *Operation Builder* on the left. The glyph visual encoding is documented in Supplemental Figures S3 and S4.

The root of the tree represents the initial segment loaded into the tool. The tree structure captures the provenance of segment refinement with a visual record of the already-conducted analysis paths, providing an easy-to-understand overview of all analysis questions pursued within an interactive session. It also supports creating a new analysis branch from any previously constructed segment. This view is tightly coupled to the other two views: selecting a segment (which turns blue) triggers a complete update of *Segment Inspector* view on the right to reflect that segment's data. It also updates the *Operation Inspector* on the left to show full details about the attribute constraints used to create the segment, featuring an expanded version of the glyph with detailed annotations.

The most fundamental design choice in Segmentifier is that each segment refinement operation corresponds to one attribute constraint, so that a path through the recorded analysis tree from root to leaf provides a crisp way to understand the provenance of a segment as a sequential series of attribute constraints.

4.2. Segment Inspector View

The *Segment Inspector* view addresses the *View* model step, showing all attributes of the *selected segment* to support decisions on whether to refine, export, or conclude with a sufficient answer.

4.2.1. Segment Ranges

The top *Ranges* section, shown in Fig. 2C, consists of linked histograms showing the distributions for all of the ordered per-sequence segment attributes. The top row contains time-related attributes: duration (with the option of switching between minutes, hours, or days), start hour, day of week, and start date. The bottom row shows the distribution of counts for individual actions: it always shows the total number of actions in a sequence, and on demand shows additional charts with counts for each specific type of action. Brushing any of the charts automatically cross-filters all other range charts to show correlations between attributes. The scale of each horizontal axis is data driven to ensure that the major trend is visible despite long-tail distributions, with an aggregate *leftover* bar on the far right labelled with $>$ representing all values beyond the point that bars are shorter than one pixel.

4.2.2. Actions

The middle *Actions* section, shown in Fig. 2C, has three charts. On the left is the *Contains Chart* that shows the distribution of action

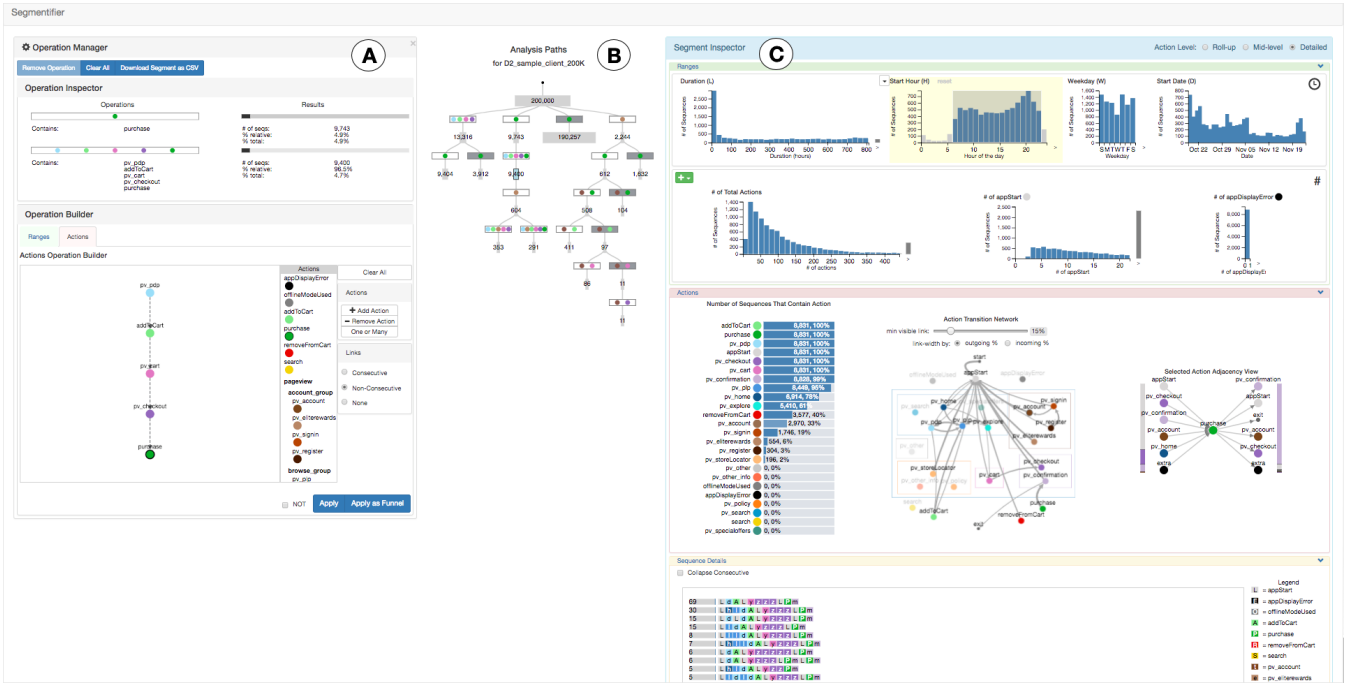

Figure 2: *The Segmentifier interface. The Operation Manager View (A) is responsible for the inspection and creation of operations used to refine segments. The Analysis Paths View (B) is a graphical history that records all segments (gray rectangles) and operations (rectangular glyphs) created during analysis; the selected segment is highlighted in blue with a black outline. The Segment Inspector View (C) shows the attributes and raw sequences associated with the selected segment.*

types across the selected segment's sequences as a bar chart with labels for both absolute counts and relative percentages, with actions color coded to match the other two charts.

The *Action Transition Network* is in the middle and *Selected Action Adjacency View* is on the right. These charts primarily support the Ordering task (T4). They are based on a derived network of **action transition** trigrams for the combination of an action, the action before it, and the action after it; these are computed from the union of actions across all sequences within the segment.

The middle *Action Transition Network* shows the full node-link graph of transitions between all action types. Selecting a single node updates the *Selected Action Adjacency View* butterfly chart on the right: the chosen action is in the middle with left-aligned incoming nodes and right-aligned outgoing ones, both flanked by stacked bar charts showing the proportions of their occurrence. The exact values appear in a tooltip on hover.

Our requirements analysis showed that different levels of detail of action types were required during different parts of the analysis process. We derive an **action hierarchy** with three levels of detail to classify actions into groups, as shown in Supplemental Figure S1. The analyst chooses which level of the action hierarchy is currently active: *Detailed*, *Mid-level*, or *Roll-up*; the action types shown in all action-related charts are updated accordingly. This action hierarchy is used to transform sequences of raw actions with high variability into derived sequences of aggregate actions with much lower variability, allowing analysts to consider more coherent large-scale structure. Supplemental Section S1 provides more details about this derived data abstraction and its visual encoding as a manually created layout and manually chosen color palette that is used consistently for actions across all views, with comparison of the results at each of the three levels in Supplemental Figure S1. While the exact details of these page groups and names may be somewhat specific to this particular e-commerce company, the generalizable idea behind this approach is the value of a derived multi-scale hierarchy of aggregate action types for analyzing consumer behavior, inspired in part by previous work incorporated in Frequence [PW14], COQUITO [KPS16], DecisionFlow [GS14], Session Viewer [LRTM07], TrailExplorer2 [SWSM12], Scribe Radar [WL14] and other visual analytic systems [WZT*16, LWD*17].

4.2.3. Sequence Details

The *Sequence Details* section is at the bottom of the *Segment Inspector*, as shown in Fig. 2C. Each row shows a sequence as series of boxes for actions, with action types updated according to the action hierarchy choice. The action boxes are labelled by a unique character, manually chosen to be evocative and memorable for all actions in the hierarchy (as were the layout and colors). Sequences are sorted by frequency, with one row for each group of identical sequences with a numerically labelled bar chart on the left showing counts. The *Collapse Consecutive* checkbox aggregates consecutive identical actions into a single action box, leading to a different grouping of sequences, as shown in Supplemental Figure S2.

4.3. Operation Manager View

The *Operation Manager* view, shown in Fig. 2A, is designed to support the fast and fluid refining of segments in the *Refine* model step. Analysts iteratively apply refinement **operations** to **filter** a segment into a smaller one via attribute constraints or **partition** it into multiple others by dividing according to attribute values. This view contains two sections, the top *Operation Inspector* and the bottom *Operation Builder*. The visual encoding and interaction are linked within this view and between the others.

The *Operation Builder* section at the bottom has two tabs, one for ranges and one for actions, following the *Segment Inspector* structure. The *Ranges Operation Builder* is linked with *Segment Inspector Ranges* section: when any of those charts are selected, both tabs are automatically populated with a bar showing the full range of values for the relevant attribute, for fast filtering or partitioning using the tab widgets. Operations are visually encoded in detail in these *Builder* tabs, with similar but more concise operation glyphs used to represent each operation in the *Operation Inspector* (and also in the *Analysis Paths* view). The rectangular glyphs are designed to be evocative of the *Builder* displays, to concisely show the type of operation and the attribute used through an icon on the left, with an indication of values selected in the main region. The details of their design and several examples are provided in Supplemental Figures S3 and S4.

The *Actions Operation Builder* supports investigation of how many sequences contain certain patterns of actions. This view, shown in detail in Supplemental Figure S5, is essentially a glyph-based regular expression query, with a limited set of choices carefully chosen to be comprehensible to analysts with limited technical backgrounds. The user selects actions from the *Action List*, specifies whether the links between them are consecutive (shown as solid lines) or non-consecutive (shown with dashed lines), chooses whether to invert the pattern, and then applies the pattern normally or as a purchasing funnel. This builder is also tightly linked with other views: selecting a sequence in the *Sequence Details* section on the lower right of the screen automatically instantiates its pattern for potential editing in the builder, as does clicking on an already created *actions* operation node in the *Operation Inspector* above.

The top *Operation Inspector* section provides detailed information about the *segment* and *operation* nodes currently selected in the *Analysis Paths* view. Each row represents one of the series of operations that was applied to create the segment. The operation details, on the left, include the corresponding *operation glyph* and text describing the constraints it specifies. The results, on the right, provide a visual and textual representation of the segment size in absolute terms and compared to the previous and initial segments.

5. Implementation and Pre-processing

Segmentifier was built in Javascript with D3.js [BOH11], using crossfilter.js [Bos13] to manage the flow of data between views. We cache crossfilter results to speed up operations that proceed down an analysis path from the root to a leaf segment, so interactive response time varies depends on the recent history of user selections in addition to the overall number of sessions and their length. Sequences grouped by *clientid* are substantially longer than those grouped by *sessionid*. On a 1.7 GHz Intel Core i7 computer with 8GB of memory, loading time is roughly linear in the number of sequences (15 seconds for 200K and 1 minute for 1M sequences). The interactive response time for updating after segment selection ranges from 2-10 seconds for smaller per-session datasets of 200K sequences but can extend in some cases to 30-300 seconds for larger per-client 200K sequence datasets. The response times for the 1M sequence per-session dataset range from 12-25 seconds; the supplemental video using it has been edited for brevity.

Pre-processing of raw clickstream datasets into Segmentifier format begins with an SQL query to extract a table of one sequence per row with start time, end time, clientid, and sessionid. Each action is mapped to a unique character corresponding to the lowest level of the action hierarchy. This processing stage was conducted upon the e-commerce company computing infrastructure and took under ten minutes in all cases. Subsequent processing occurs in Python with a script that creates two new aggregated strings for the other two levels of the action hierarchy and computes all of the derived time range attributes, which again takes under ten minutes in all cases. For each new website analyzed, we expect domain experts to map site-specific page templates to the 17 action types at the detailed level of the action hierarchy. We settled on this manual grouping by domain experts following Liu et al. [LWD*17], who concluded that this approach was more effective than the multiple automatic techniques they considered.

6. Results

We show the effectiveness of Segmentifier through a detailed usage scenario and a case study with a domain expert, showcasing two real-world datasets.

6.1. Usage Scenario

The first dataset comes from a live e-commerce site (CUST1), with 62 site-specific page templates. The data collected over two months constitutes 4 million per-session sequences and 10.5 million actions. We randomly sampled over the entire time period to a manageable size of 1 million sequences containing 2.6 million actions. Fig. 3 showcases the usage scenario as a series of small detail views extracted from full screenshots. Each subfigure has a header indicating the analysis flow according to the clickstream segment analysis framework; a checkmark by View denotes finding an answer. The supplemental video shows the look and feel of the interactive system on this example. This dataset is also used in Fig 2 and Supplemental Figs. S1-S4.

A. Identify and filter out unexpected behavior (T1). The initial segment of 1 million sequences is loaded in as the root of the tree in the *Analysis Paths* view, highlighted in blue as the selected segment (Fig. 3A *Segment*). The analyst begins by looking at the *Actions* section and notices from the *Contains Chart* that only 41% of sequences contain a PAGEVIEW action (Fig. 3A *View*). Noting unexpected behavior, they explore further by looking at the sequences in *Sequence Details* (Figure S2a). They select the *Collapse Consecutive* option to simplify the sequences (Figure S2b) and notice that over 58% of sequences only contain APPSTART actions (an action

Figure 3: *Usage scenario workflow broken down into the analysis of 7 segments/operations. Details of each in Section 6.1. View boxes with checkmarks means answers were found at this step.*

signifying when a site loads), showing the existence of an APPSTART tracking issue.

To filter out the unexpected behavior they select the sequence in *Sequence Details* which automatically creates the action pattern in the *Actions Operation Builder* (Fig. 3A *Refine*): a consecutive path from START to one or more APPSTART (black dot) to EXIT. The analyst selects the NOT option and clicks *Apply*. The operation is recorded in the *Analysis Paths* view (Fig. 3A *Record*) by a path from the initial segment to an *operation node* representing the action pattern followed by the resulting *segment node* whose size and label show the unaffected 415,980 sequences.

B. Identify unfavorable behavior (T1). The analyst selects the resulting segment (Fig. 3B *Segment*) and all attributes in the *Segment Inspector* are updated. After deselecting the *Collapse Consecutive* option in the *Sequence Details* view, the analyst notices that approximately 50% of sequences contain one APPSTART action and one PAGEVIEW action, referred to as a *bounce* (Fig. 3B *View*). The analyst decides to further explore the unfavorable behavior by once again selecting a sequence to access the pattern in the *Actions Operation Builder* (Fig. 3B *Refine*) and applying the filter operation. The operation is recorded in the *Analysis Paths* view (Fig. 3B *Record*).

C. Verify frequency of unfavorable behavior (T3). They select the resulting segment (Fig. 3C *Segment*). To get more details, the analyst switches to the *Detailed* level of the hierarchy (Fig. 3C *Refine*) which updates the charts in the Segment Inspector view. The *Contains Chart* (Fig. 3C *View*) shows the analyst the distribution of pages users bounced from. This information is an indicator of potential problem pages which should be explored. Content with the information gathered from this segment, the analyst concludes this analysis path and continues to explore other segments.

D. Identify expected behavior (T1). The analyst wants to analyze the purchasing funnel, an expected behavior, and determine how many users drop out at each step. They select the previous segment (Fig. 3D *Segment*) and create a pattern in the *Actions Operation Builder* (Fig. 3D *Refine*) of non-consecutive links between the five actions in the purchasing funnel. By selecting *Apply as Funnel*, an operation node and resulting segment node is created for each step of the pattern in the *Analysis Paths* view (Fig. 3D *Record*).

E.1 Verify frequency of expected behavior (T3). They select the last resulting segment representing all sequences that contain the full purchasing funnel. The *Operation Inspector* has a detailed breakdown of the series of operations applied to create the segment showing the dropout percentage at each step of the purchasing funnel. They notice that 67% make it to the checkout step but never purchase (Fig. 3E.1 *View*) and hypothesize the possibility of an error in the checkout process preventing consumers from purchasing.

E.2 Drill down on known behaviors (T2). In the *Ranges Attributes* section, the analyst creates the PV_PDP *count* chart (Fig. 3E.2 *Segment*), describing how many product pages are in each sequence. By selecting this chart, the *Ranges Operation Builder* range bar updates to show the total range, 1-44 product pages. To further explore the distribution, the analyst applies a partition operation with two partition bars at values 2 and 10 (Fig. 3E.2 *Refine*). A *partition node* is created in the *Analysis Paths* view with three resulting *segment nodes* (Fig. 3 E.2 *Record*).

F. Determine frequency of more-finely grained behaviors (T3). After selecting the *partition operation node* (Fig. 3F *Operation*), the *Operation Inspector* updates to show details of the resulting three segments (Fig. 3F *View*): 23% of purchasers view one product page, 67% view between 2-10 pages, and 10% view between 10-44. The analyst uses this split to tailor email for each customer group.

G. Identify favorable behavior (T1). They re-select the segment representing sequences matching a full purchasing funnel (Fig. 3G *Segment*) and notice in the *Sequence Details* view that this favorable behavior is suitable for export (Fig. 3G *View*). The number of sequences is small and there is low variability between them, satisfying the data requirements for pattern matching techniques, so they download the segment (Fig. 3G *Export*).

6.2. Case Study

An industry clickstream analyst was preparing to give an one-month post-launch presentation to a commercial customer. Their goal was to use Segmentifier to discover insights and suggestions for improvement to relay to the customer. We conducted two separate chauffeured analyses with this domain expert, who was also one of the interviewees during requirements analysis.

6.2.1. Analysis #1

This analysis used data from a different customer website (CUST2), with 25 site-specific page templates mapped into our action hierarchy. The full dataset collected in one month was 20M sequences containing 190M actions. We randomly sampled 200K sequences each representing one user *session*, containing 1.9 million actions. The resulting *Analysis Paths* tree shown in Fig. 4 reflects the four analysis tasks completed during the two hour session. We summarize the insights found for each analysis here; Supplemental Figs. S5-S25 in Sections S4 and S5 contain full screenshots and further discussion.

A. Analyze purchasing behavior. The analyst drilled-down on purchasing behavior (T2) by filtering out the sequences that contain a PURCHASE action. Two separate analysis paths emerged from this segment: 1) They drilled-down on the checkout flow (T2) and identified that there were sessions that contain more checkout pages than necessary to complete a purchase (T1) and determined the frequency of this behaviour (T3): 12% of sessions. 2) They drilled-down on the purchasing funnel (T2) and determined the percentage of the purchasing funnel completed in one session (T3) and identified that 30% of users actually exit the site and return later to complete their purchase (T1).

B. Comparing morning vs night behavior. They drilled-down on a hypothesis that the behavior of users would change based on the time of day (T1): 7-9 am versus 7-9 pm. They compared the frequency of the number of actions in each session and the percentage of sessions which completed the full purchasing funnel (T3) and found no significant differences.

C. Analyze add and remove from cart behavior. The analyst then drilled-down on *adding to cart* and *removing from cart* behavior (T2) and identified that 30% of users who removed from cart exited the session and most likely did not come back (T1, T3).

D. Analyze purchasing funnel. Using *Apply as Funnel*, the analyst was able to easily determine the percentage (frequency) of sessions that drop out at each step of the purchasing funnel (T3). Combining with the previously gathered information, they determined that 20% of people who get to checkout will not end up purchasing (T3).

6.2.2. Analysis #2

For this analysis we used the same base dataset (CUST2), but inspected the derived sequences of all actions performed by a single client over the entire one-month time frame instead of using the sessions with 30-minutes cutoff values. The sampled dataset consisted of 200,000 sequences, each representing one client, containing 4.3 million actions in total. Fig. 2 shows an overview of the analysis.

With the new type of sequences, the domain expert revisited some of the previously explored questions. They were interested in drilling down on the reasons for dropping out of the purchasing funnel at the checkout stage (T2) and the effect of removing from cart (T4). Using the *Action Transition Network* and the *Selected Action Adjacency View*, they discovered that 25% of clients that removed from cart at the checkout state, exited and never returned to the site to purchase (T3). They also noticed that an APPSTART action was triggered every time before the PV_CART action which was an unusual behavior (T4). Followup investigation determined a problem with the cart pages of the website.

The domain expert drilled down on the effect of a new loyalty program page, *myBeautyPage*, recently added on the website (T2). During the pre-processing step, the domain expert ensured that the *myBeautyPage* template name was the only one mapped to the PV_ELITEREWARDS action in the action hierarchy. They easily discovered that 1% of all clients signed up for the loyalty program and from those 27% made a purchase (T3). They partitioned these clients to understand at what point of the purchasing funnel they signed up for the loyalty program and identified that it occurred mostly before adding to cart (T1). Finally, they saw that clients that accessed the loyalty program page generally had longer sequences, signaling that they were more committed customers (T1).

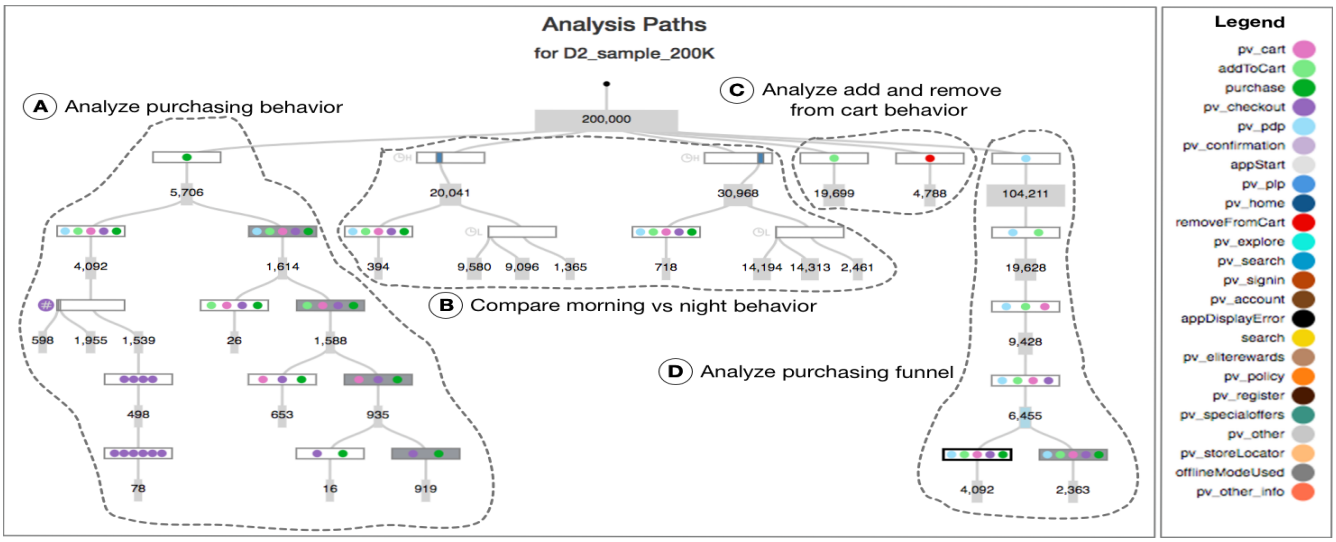

Figure 4: *Annotated Analysis Paths View representing the four analyses done in case study with domain expert.*

6.3. Domain Expert Feedback

The clickstream analyst was able to find many valuable insights to take back to their customer, and saved several screenshots for inclusion in their presentation. They thought the interface was easy to use and understand, specifically appreciating the *Sequence Details* view for generating new questions and the *Analysis Paths* view for re-examining previously created segments and helping them remember their analysis process. They also noted the usefulness of switching between different levels of the action hierarchy for different tasks and how the color scheme reinforced the hierarchy.

Their wishlist of additional capabilities included saving workflows to reuse on different data, seeing a fourth level of actions with more detail showing all site-specific *page templates*, increased capabilities of the pattern builder to filter by more sophisticated patterns, and the ability to easily compare two segments.

7. Discussion and Future Work

We carefully considered the tradeoff between power and simplicity. Many previous systems provide full support for regular expressions [CvW18, LRTM07], which are notoriously difficult for non-programmers to handle even when presented through a visual language; we deliberately kept the scope of the *Actions Operation Builder* limited. Despite the expert feedback requesting more functionality, we still suspect that additional complexity would make the system unusable for time-crunched analysts.

Our primary focus was the agile and iterative development of Segmentifier's design, with very modest engineering effort to improve loading and processing times to achieve a base level of usability. A few months of data collection can yield many millions of sequences, but our implementation strains at one million and has better responsiveness at 200K sequences. We randomly sample from the full datasets in hopes of capturing sequence variability over the time ranges of interest to the domain experts, rather than simply targeting shorter time periods. We conjecture that our fundamental design would scale to datasets larger than our current engineering accommodates, and we have some evidence for this optimism. The case studies with the domain expert were run using a sample of 200,000 sequences from the CUST2 dataset to ensure that Segmentifier interaction would be completely fluid with no hindrances in the analysis loop. We later replicated the first analysis ourselves with a larger sample of 1 million session sequences from CUST2, up to the limit of what Segmentifier can load. Although the processing time was frequently slower, the same patterns were visible. Future work to improve the capacity and speed of our approach would reveal the boundary between engineering and design issues.

8. Conclusion

Segmentifier bridges the gap from the large and noisy clickstreams in real-world e-commerce settings to downstream analysis techniques that require relatively clean data. Our task and data abstractions connect the informal mental model of clickstream analysts about interesting behavior with viewable characteristics of sequences, actions, and segments via evocative derived data attributes. Our high-level analysis model and its concrete instantiation in the Segmentifier interface allows analysts to view, refine, export, abandon, and draw conclusions from iteratively developed segments with an automatically recorded analysis path to show provenance. Understandable results can be found quickly by analysts with limited time and skills. The usage scenario and case study show the utility of Segmentifier as a step forward in handling the complexities of realistic e-commerce clickstream datasets.

Acknowledgments

We thank our collaborators at Mobify and acknowledge support from Mobify and MITACS. We appreciate feedback on this work from Anamaria Crisan, Madison Elliot, Zipeng Liu, and Michael Oppermann.

References

[AMTB05] AIGNER W., MIKSCH S., THURNHER B., BIFFL S.: PlanningLines: novel glyphs for representing temporal uncertainties and their evaluation. In *Proc. Intl. Conf. Information Visualisation (IV)* (2005), pp. 457–463. doi:10.1109/IV.2005.97. 4

[BB01] BRAINERD J., BECKER B.: Case study: e-commerce clickstream visualization. In *Proc. IEEE Symp. Information Visualization (InfoVis)* (2001), pp. 153–156. doi:10.1109/INFVIS.2001.963293. 4

[BBD08] BURCH M., BECK F., DIEHL S.: Timeline trees: Visualizing sequences of transactions in information hierarchies. In *Proc. ACM Working Conf. on Advanced Visual Interfaces (AVI)* (2008), pp. 75–82. doi:10.1145/1385569.1385584. 4

[BOH11] BOSTOCK M., OGIEVETSKY V., HEER J.: D3: Data-driven documents. *IEEE Trans. Visualization and Computer Graphics 17*, 12 (2011), 2301–2309. doi:10.1109/TVCG.2011.185. 7

[Bos13] BOSTOCK M.: crossfilter.js. https://github.com/square/crossfilter, 2013. 7

[CvW18] CAPPERS B. C. M., VAN WIJK J. J.: Exploring multivariate event sequences using rules, aggregations, and selections. *IEEE Trans. Visualization and Computer Graphics 24*, 1 (2018), 532–541. doi:10.1109/TVCG.2017.2745278. 5, 10

[FKSS06] FAILS J. A., KARLSON A., SHAHAMAT L., SHNEIDERMAN B.: A visual interface for multivariate temporal data: Finding patterns of events across multiple histories. In *Proc. IEEE Symp. Visual Analytics Science and Technology (VAST)* (2006), pp. 167–174. doi:10.1109/VAST.2006.261421. 4

[GA] Google Analytics. URL: https://analytics.google.com/. 1

[GS14] GOTZ D., STAVROPOULOS H.: DecisionFlow: Visual analytics for high-dimensional temporal event sequence data. *IEEE Trans. Visualization and Computer Graphics 20*, 12 (2014), 1783–1792. doi:10.1109/TVCG.2014.2346682. 4, 6

[GW14] GROLEMUND G., WICKHAM H.: A cognitive interpretation of data analysis. *Intl. Statistical Review 82*, 2 (2014), 184–204. 1

[GXZ*18] GUO S., XU K., ZHAO R., GOTZ D., ZHA H., CAO N.: EventThread: Visual summarization and stage analysis of event sequence data. *IEEE Trans. Visualization and Computer Graphics 24*, 1 (2018), 56–65. doi:10.1109/TVCG.2017.2745320. 4

[HLBE14] HAAG F., LOHMANN S., BOLD S., ERTL T.: Visual SPARQL querying based on extended filter/flow graphs. In *Proc. ACM Intl. Working Conf. on Advanced Visual Interfaces* (2014), AVI '14, pp. 305–312. doi:10.1145/2598153.2598185. 4

[HMSA08] HEER J., MACKINLAY J., STOLTE C., AGRAWALA M.: Graphical histories for visualization: Supporting analysis, communication, and evaluation. *IEEE Trans. Visualization and Computer Graphics 14*, 6 (2008), 1189–1196. doi:10.1109/TVCG.2008.137. 4

[hTKK04] HSIEN TING I., KIMBLE C., KUDENKO D.: Visualizing and classifying the pattern of user's browsing behavior for website design recommendation. In *Proc. Intl. ECML/PKDD Workshop on Knowledge Discovery in Data Stream* (2004), pp. 101–102. 4

[KBGE11] KOCH S., BOSCH H., GIERETH M., ERTL T.: Iterative integration of visual insights during scalable patent search and analysis. *IEEE Trans. Visualization and Computer Graphics 17*, 5 (2011), 557–569. doi:10.1109/TVCG.2010.85. 5

[KBK11] KRSTAJIC M., BERTINI E., KEIM D.: CloudLines: Compact display of event episodes in multiple time-series. *IEEE Trans. Visualization and Computer Graphics 17*, 12 (2011), 2432–2439. doi:10.1109/TVCG.2011.179. 4

[KPHH11] KANDEL S., PAEPCKE A., HELLERSTEIN J., HEER J.: Wrangler: Interactive visual specification of data transformation scripts. In *Proc. ACM Conf. on Human Factors in Computing Systems (CHI)* (2011), pp. 3363–3372. doi:10.1145/1978942.1979444. 1

[KPS16] KRAUSE J., PERER A., STAVROPOULOS H.: Supporting iterative cohort construction with visual temporal queries. *IEEE Trans. Visualization and Computer Graphics 22*, 1 (2016), 91–100. doi:10.1109/TVCG.2015.2467622. 4, 6

[LKD*17] LIU Z., KERR B., DONTCHEVA M., GROVER J., HOFFMAN M., WILSON A.: CoreFlow: Extracting and visualizing branching patterns from event sequences. *Computer Graphics Forum 36*, 3 (2017), 527–538. doi:10.1111/cgf.13208. 4

[LPSH01] LEE J., PODLASECK M., SCHONBERG E., HOCH R.: Visualization and analysis of clickstream data of online stores for understanding web merchandising. *Data Mining and Knowledge Discovery 5* (2001), 59–84. 5

[LRTM07] LAM H., RUSSELL D., TANG D., MUNZNER T.: Session Viewer: Visual exploratory analysis of web session logs. In *Proc. IEEE Symp. Visual Analytics Science and Technology (VAST)* (2007), pp. 147–154. doi:10.1109/VAST.2007.4389008. 5, 6, 10

[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 Trans. Visualization and Computer Graphics 23*, 1 (2017), 321–330. doi:10.1109/TVCG.2016.2598797. 2, 4, 6, 7

[MLL*13] MONROE M., LAN R., LEE H., PLAISANT C., SHNEIDERMAN B.: Temporal event sequence simplification. *IEEE Trans. Visualization and Computer Graphics 19*, 12 (2013), 2227–2236. doi:10.1109/TVCG.2013.200. 5

[MSD*16] MALIK S., SHNEIDERMAN B., DU F., PLAISANT C., BJARNADOTTIR M.: High-volume hypothesis testing: Systematic exploration of event sequence comparisons. *ACM Trans. Interact. Intell. Syst. 6*, 1 (2016), 9:1–9:23. doi:10.1145/2890478. 4

[Mui11] MUI P.: Introducing flow visualization: visualizing visitor flow, 2011. URL: https://analytics.googleblog.com/2011/10/introducing-flow-visualization.html. 4

[NXW*16] NGUYEN P. H., XU K., WHEAT A., WONG B. L. W., ATTFIELD S., FIELDS B.: Sensepath: Understanding the sensemaking process through analytic provenance. *IEEE Trans. Visualization and Computer Graphics 22*, 1 (2016), 41–50. doi:10.1109/TVCG.2015.2467611. 5

[PG13] PERER A., GOTZ D.: Data-driven exploration of care plans for patients. In *Extended Abstracts of ACM Conf. Human Factors in Computing Systems (CHI)* (2013), pp. 439–444. doi:10.1145/2468356.2468434. 4

[PW14] PERER A., WANG F.: Frequence: Interactive mining and visualization of temporal frequent event sequences. In *Proc. Intl. Conf. on Intelligent User Interfaces (IUI)* (2014), pp. 153–162. doi:10.1145/2557500.2557508. 4, 6

[RESC16] RAGAN E. D., ENDERT A., SANYAL J., CHEN J.: Characterizing provenance in visualization and data analysis: An organizational framework of provenance types and purposes. *IEEE Trans. Visualization and Computer Graphics 22*, 1 (2016), 31–40. doi:10.1109/TVCG.2015.2467551. 5

[SFC07] SILVA C. T., FREIRE J., CALLAHAN S. P.: Provenance for visualizations: Reproducibility and beyond. *Computing in Science Engineering 9*, 5 (2007), 82–89. doi:10.1109/MCSE.2007.106. 5

[SFCQ15] SHI C., FU S., CHEN Q., QU H.: VisMOOC: Visualizing video clickstream data from massive open online courses. In *Proc. IEEE Symp. Pacific Visualization (PacificVis)* (2015), pp. 159–166. doi:10.1109/PACIFICVIS.2015.7156373. 5

[SMM12] SEDLMAIR M., MEYER M., MUNZNER T.: Design study methodology: Reflections from the trenches and the stacks. *IEEE Trans. Visualization and Computer Graphics (Proc. Inf oVis 2012) 18*, 12 (2012), 2431–2440. doi:10.1109/TVCG.2012.213. 2

[SWSM12] SHEN Z., WEI J., SUNDARESAN N., MA K. L.: Visual analysis of massive web session data. In *Proc. IEEE Symp. Large Data Analysis and Visualization (LDAV)* (2012), pp. 65–72. doi:10.1109/LDAV.2012.6378977. 5, 6

[SZD*16] SARIKAYA A., ZGRAGGEN E., DELINE R., DRUCKER S., FISHER D.: Sequence pre-processing: Focusing analysis of log event data. In *IEEE VIS The Event Event: Temporal & Sequential Event Analysis Workshop* (2016). URL: https://eventevent.github.io/papers/EVENT_2016_paper_12.pdf. 1, 2

[VJC09] VROTSOU K., JOHANSSON J., COOPER M.: ActiviTree: Interactive visual exploration of sequences in event-based data using graph similarity. *IEEE Trans. Visualization and Computer Graphics 15*, 6 (2009), 945–952. doi:10.1109/TVCG.2009.117. 5

[WG12] WONGSUPHASAWAT K., GOTZ D.: Exploring flow, factors, and outcomes of temporal event sequences with the outflow visualization. *IEEE Trans. Visualization and Computer Graphics 18*, 12 (2012), 2659–2668. doi:10.1109/TVCG.2012.225. 4

[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 *Proc. ACM Conf. Human Factors in Computing Systems (CHI)* (2011), pp. 1747–1756. doi:10.1145/1978942.1979196. 4

[WHS*02] WATERSON S. J., HONG J. I., SOHN T., LANDAY J. A., HEER J., MATTHEWS T.: What did they do? understanding clickstreams with the WebQuilt visualization system. In *Proc. ACM Working Conf. on Advanced Visual Interfaces (AVI)* (2002), pp. 94–102. doi:10.1145/1556262.1556276. 4

[WL14] WONGSUPHASAWAT K., LIN J.: Using visualizations to monitor changes and harvest insights from a global-scale logging infrastructure at Twitter. In *Proc. IEEE Conf. Visual Analytics Science and Technology (VAST)* (2014), pp. 113–122. doi:10.1109/VAST.2014.7042487. 5, 6

[WPQ*08] WANG T. D., PLAISANT C., QUINN A. J., STANCHAK R., MURPHY S., SHNEIDERMAN B.: Aligning temporal data by sentinel events: Discovering patterns in electronic health records. In *Proc. ACM Conf. Human Factors in Computing Systems (CHI)* (2008), pp. 457–466. doi:10.1145/1357054.1357129. 4

[WSSM12] WEI J., SHEN Z., SUNDARESAN N., MA K. L.: Visual cluster exploration of web clickstream data. In *Proc. IEEE Conf. Visual Analytics Science and Technology (VAST)* (2012), pp. 3–12. doi:10.1109/VAST.2012.6400494. 4

[WZT*16] WANG G., ZHANG X., TANG S., ZHENG H., ZHAO B. Y.: Unsupervised clickstream clustering for user behavior analysis. In *Proc. ACM Conf. on Human Factors in Computing Systems (CHI)* (2016), pp. 225–236. doi:10.1145/2858036.2858107. 4, 6

[ZBS16] ZHANG X., BROWN H.-F., SHANKAR A.: Data-driven personas: Constructing archetypal users with clickstreams and user telemetry. In *Proc. ACM Conf. on Human Factors in Computing Systems (CHI)* (2016), pp. 5350–5359. doi:10.1145/2858036.2858523. 4

[ZDFD15] ZGRAGGEN E., DRUCKER S. M., FISHER D., DELINE R.: (s,Qu)Eries: Visual regular expressions for querying and exploring event sequences. In *Proc. ACM Conf. on Human Factors in Computing Systems (CHI)* (2015), pp. 2683–2692. doi:10.1145/2702123.2702262. 4

[ZLD*15] ZHAO J., LIU Z., DONTCHEVA M., HERTZMANN A., WILSON A.: MatrixWave: Visual comparison of event sequence data. In *Proc ACM Conf. Human Factors in Computing Systems (CHI)* (2015), pp. 259–268. doi:10.1145/2702123.2702419. 4