Segmentifier: Interactively Refining Clickstream Data into Actionable Segments

by

Kimberly Dextras-Romagnino

B. Sc., Concordia University, 2015

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

April 2018

Abstract

Clickstream data has the potential to provide actionable insights into e-commerce consumer behavior, but previous techniques fall short of handling the scale and complexity of real-world datasets. We present Segmentifier, a novel visual analytics interface that supports an iterative process of refining collections of action sequences into meaningful segments that are suitable for downstream analysis with techniques that require relatively small and clean input. We also present task and data abstractions for the application domain of clickstream data analysis, leading to a framework that abstracts the segment analysis process in terms of six functions: view, refine, record, export, abandon, and conclude. The Segmentifier interface is simple to use for analysts operating under tight time constraints. 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 rich set of views that show the underlying raw sequence details and the derived data of segment attributes, and a detailed glyph-based visual history of the automatically recorded analysis process showing the provenance of each segment in terms of 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 an e-commerce analyst.

Lay Summary

Companies engaged in e-commerce are recording every action consumers take on websites, resulting in rich clickstream datasets that have the potential to provide insight into consumer behavior. However, analyzing this type of data is difficult and an open research problem due the size and noisiness associated with real world clickstream datasets. To address this problem, we designed and implemented Segmentifier, a novel interface that embodies an initial analysis step that gives analysts an overview of the data and allows them to iteratively refine and separate the data into segments that either provide insights or allow more effective use of other techniques. We also introduce a framework that encapsulates the design of our system which explains this segment analysis process in terms of six functions: view, refine, record, export, abandon, and conclude. We evaluate our system through a usage scenario with real-world data and a case study with an e-commerce analyst.

Preface

This thesis is based on material contained in the following paper:

- Kimberly Dextras-Romagnino and Tamara Munzner. Segmentifier: Interactively Refining Clickstream Data into Actionable Segments. Submitted for publication.

I was the lead researcher responsible for gathering requirements from our domain experts, designing, implementing, and evaluating the *Segmentifier* interface, and writing the paper that this thesis is based on. Dr. Tamara Munzner contributed largely in the design process of the interface as well as helping to frame, write, and edit parts of the paper.

Table of Contents

List of Figures

Acknowledgments

First and foremost, I would like to thank my supervisor Tamara Munzner who I would have been lost without. Her dedication to her students and her work and her overall awesome personality is what made these past few years both incredibly rewarding and enjoyable.

I would like to thank everyone else who has contributed to the work in this thesis. Thanks to Mobify for collaborating with me and partially funding this work. More specifically thanks to Peter McLachlan, Boris Lau, Amanda Naso, Ben Cole, and Tim Schultz who were always available to chat and help me in any way they could. Thanks to my second reader Rachel Pottinger for taking the time to read my thesis and provide very helpful comments.

I would like to thank all my friends who are the reason I successfully made it through this program. Most importantly, thanks to my housemates India Stephenson, Hudson Mclellan, and Taz for being my second family. I could not have asked for better roommates and friends! Thanks to the members of the Infovis group, Michael Oppermann, Zipeng Liu, Anamaria Crisan, and Madison Elliott, for their continuous feedback and keeping the lab an exciting place to be. Thanks to Dilan Ustek, Meghana Venkatswamy, Vaden Masrani, Louie Dinh, Jacob Chen, Robbie Rolin, Yasha Pushak, Halldor Thorhallsson, Giovanni Viviani, Neil Newman, Antoine Ponsard, Clement Fung and all my other graduate school friends for making the last few years full of great times and adventures. Thanks to all my oldest friends all across Canada whose friendships I will treasure forever.

Finally, I would like to thank my parents, my brother, and the rest of my family for all their support throughout my Masters. I would not have made it without you!

Chapter 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. Analysis of this data includes answering a variety of questions ranging from discovering the page users most commonly exit on to classifying different types of buying behaviors. The people tasked with understanding clickstream data, referred to as **clickstream analysts**, generally sit on the data analysis team of an e-commerce company. They can range from computer scientists to people with no technical background. Moreover, they often have many demands on their time and minimal training in data analysis.

In practice, clickstream analysts frequently turn to third party platforms such as Google Analytics [2] or Adobe Analytics [1] that focus on making aggregated high-level metrics easy to see quickly but do not provide support for fully drilling down into the details of consumer behavior. While these platforms do provide an overall picture of website performance, many potentially actionable insights are left unfound.

A substantial amount of research has been devoted to developing techniques for analyzing clickstream event data, typically supporting very specific tasks such as extracting common behavior through pattern mining or identifying similarly-behaving groups of users through clustering. While these techniques often yield

excellent results with clean and relatively small datasets, they fail when applied to the noisy and large datasets that occur in real-world practice. We propose an initial visual data analysis stage to convert raw datasets into data *segments* that are iteratively refined until they are sufficiently clean and small to directly show answers, or to match the assumptions of previously proposed techniques that would be used for further downstream analysis. Our flexible human-in-the-loop visual analytics approach leverages the domain expertise of clickstream analysts and is suitable for the broad set of questions that they must answer in order to obtain actionable results. It bridges a gap between the characteristics of real-world clickstream data and the assumptions of previous technique-oriented work.

One contribution of our work is a thorough characterization of task and data abstractions for clickstream data analysis, culminating in a *Clickstream Segment Analysis Framework* that abstracts the iterative and exploratory analysis process in terms of six core functions: an analyst can view, refine, record, export, abandon, and conclude data segments. Our main contribution is Segmentifier, a novel visual analytics interface that supports clickstream analysts in the iterative process of refining data segments and viewing their characteristics before downstream fine-grained analysis is conducted. The simple-to-use interface includes a rich set of views that show the derived attributes characterizing the segments in addition to the raw sequence details. 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 that we derived based on our requirements analysis. It features a detailed glyph-based visual history of the automatically recorded refinement process showing the provenance of each segment in terms of its analysis path. We show evidence for the utility of Segmentifier through a detailed usage scenario and a case study showcasing the actionable insights gained quickly by a data analyst looking at clickstream data generated by his e-commerce company.

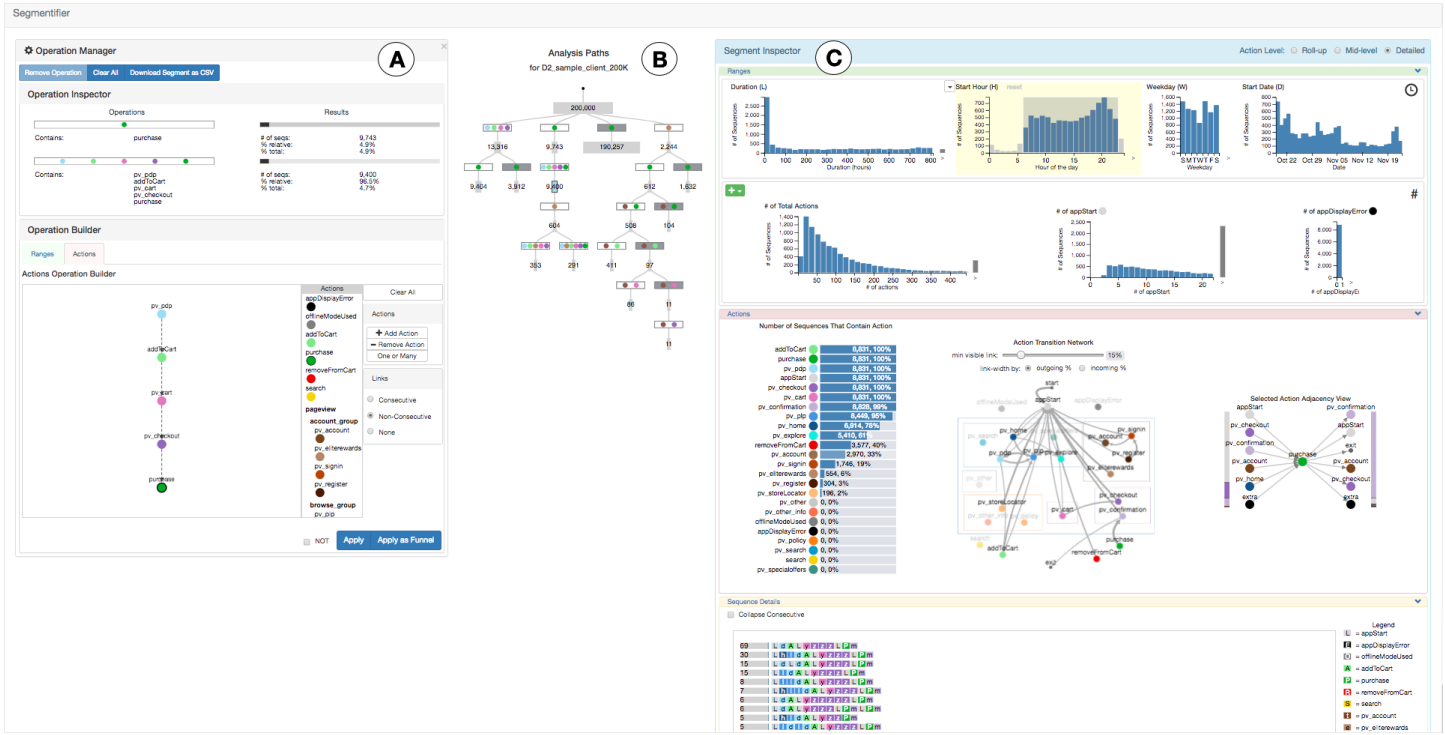

Figure 1.1: The Segmentifier interface. The Operation Manager View (A) is responsible for the inspection and creation of operations used to refine segments. The Analysis Paths View (B) is a graphical history that records all segments (gray rectangles) and operations (rectangular glyphs) created during 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.

Chapter 2

Clickstream Data and Tasks

Our requirements analysis process involved exploratory and followup interviews with 12 employees of an e-commerce middleware company over a period of 7 months. The company builds mobile websites for their clients, large e-commerce companies, while also performing internal clickstream data analysis of the websites to provide their clients with valuable insights. The interviewees included people who already conducted clickstream data analysis internally, who might conduct such analysis if provided with appropriate tools, and who met regularly with and understood the work context of their external clients' clickstream data analysts. We first describe the e-commerce clickstream analysis in domain terms, emphasizing what questions could have actionable answers in this application domain. We then present abstract versions of their tasks and a framework for clickstream segment analysis that we developed to reason about design considerations. We finally present our data abstraction featuring multiple levels of derived data.

2.1 E-commerce Clickstream Analysis Goals

We focus specifically on e-commerce clickstream data recorded from websites whose main functionality is to sell products to consumers. These websites try to generate the most revenue by finding ways 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 a user purchases).

Clickstream data analysis is intended to provide insight into consumer behavior by allowing analysts to achieve the following goals: 1) identify common successful consumer trends and optimize for them; 2) identify problems or painful paths and spend resources to fix or improve them; 3) identify groups of common behaviors in order to personalize consumer experience; 4) identify site metrics or benchmarks to keep track of the state of the website. We define analysis results to be **actionable** if they are aligned with these goals.

Examples of commonly asked 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? Can you classify different types of buying behaviors?*

Our interviews with experts revealed that industry analysts often only have a few hours to conduct analysis to discover insights in the data and then an even shorter time frame of several minutes to concisely report their findings to clients through very short progress reports or presentations. Moreover, many people in this job role have no or minimal skills in programming and statistics. The design requirement for Segmentifier is to cater to these limits of skills and available time, allowing analysts to quickly find actionable answers that are simple to interpret and convey to their clients.

2.2 Clickstream Task Abstraction

In clickstream analysis, the information available about consumer behavior is logged sequences of actions that a user performs on a website. We translate the goals of Section 2.1 into two kinds of abstract tasks, relating these sequences to behavior via subtasks with analysis questions:

Identify Tasks (I): The goal is to discover new interesting behavior.

- **I1** *Identify new **interesting** behaviors*: Is there a way to distinguish a set of sequences representing behaviors that are **expected** (example: users add to cart before purchasing), **unexpected** (example: no purchases for a month), **favorable** (example: lead to a purchase), or **unfavorable** (example: lead to

abandonment)?

- **I2** *Identify the cause or effect of a behavior*: What behaviors trigger/follow behavior X?
- **I3** *Identify more fine-grained behaviors from an initial behavior*: Can sequences that follow behavior X be described by more specific behavior Y?

Verify Tasks (V): The goal is to check whether conjectured interesting behaviors actually occur.

- **V1** *Verify existence of a behavior*: Do any sequences follow behavior X?
- **V2** *Verify amount of support for a behavior*: How many sequences follow behavior X?
- **V3** *Verify if a behavior causes another behavior*: Do sequences that follow behavior X also follow behavior Y?

2.3 Clickstream Segment Analysis Framework

Our task abstractions motivate the idea that clickstream datasets can provide answers for many different analysis questions if they can be partitioned into appropriate **segments** that encapsulate interesting behaviors. This idea follows the same premise as previous philosophies of data wrangling, that real-world datasets are extremely noisy and need to be cleaned and refined before analysis can be conducted [17]. 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 [27, 36]. Moreover, they are inherently extremely noisy with high variability between sequences, meaning that very few are identical. Sarikaya *et al*. point out the need for simplifying clickstream data for effective downstream analysis, noting the analysis 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 [36]. They call for an exploratory tool that supports an iterative process between pre-processing data and viewing the results with a graphical history to record steps. Segmentifier is our answer to this call; in

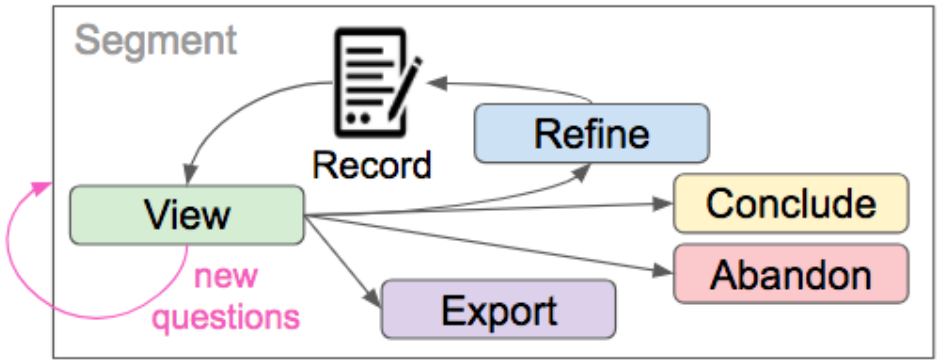

Figure 2.1: Clickstream Segment Analysis Framework iteratively generates segments that analysts can view, refine, export, abandon, or use to conclude answers for a flowing series of analysis questions.

this section we introduce a framework for clickstream segment analysis, shown in Figure 2.1, to concisely and clearly explain our design rationale.

In this abstract framework, many segments are generated and analyzed through an iterative exploration process that traverses through six functions that pertain to a current segment: **view**, **refine**, **record**, **export**, **abandon**, and **conclude**. The exploration begins with an initial segment that contains all sequences in the dataset, and typically involves flowing between many tasks opportunistically based on what the visual data analysis reveals. Exploring one question often spurs follow-up or entirely new questions; in some cases these are based on the satisfactory answer that is found, or they may arise after giving up on finding an answer for a previous question. Each refinement step is recorded. Analysis may stop when the flow of questions runs dry, or the analyst simply runs out of time.

View Viewing a segment provides information to the analyst not only about the low-level details of the raw sequences that it contains, but also a high-level characterization of attributes that characterize the segment. The result of viewing might be to notice that the segment provides answers for the current task or is a good fit and should be exported, to spark ideas on how to further refine a segment, to generate new analysis questions and thus change to a new task, or to determine whether

a segment should be abandoned.

Refine Refining allows analysts to create more appropriate segments for a task by either filtering, partitioning, or transforming the sequences of any segment. Every refinement step leads to a new segment that can be viewed, analyzed, and further refined. Decisions on how to refine are determined by the current analysis question or inspired by the discovery of interesting patterns when viewing the segment.

Record We embrace the suggestion from previous work that graphical histories to help analysts remember analysis paths are essential for any exploratory analysis [15]. Recording all refinement steps automatically helps analysts keep track of the many questions and hypotheses that are generated during an exploratory analysis session, providing enough provenance information that answers can be revisited later and understood in context.

Export A segment can be exported for further downstream analysis by applying techniques such as clustering and pattern mining that require relatively clean input data to work effectively. For example, pattern mining helps discover common sequences of events but requires as input a small number of sequences with low variability to achieve useful results. By viewing and refining segments, analysts can create segments that meet these requirements to maximize the effectiveness of these downstream analysis techniques after the export.

Conclude Successfully concluding an analysis path by finding an actionable answer for the current analysis question may occur as an immediate outcome from viewing a segment.

Abandon A segment would be abandoned after viewing if the analyst sees no obvious pathway for further refining and decides it contains no actionable results and is not suitable for export.

Each of the tasks in Section 2.2 can be answered by iteratively applying these functions until the appropriate segment is found. For *Verify* tasks, analysts can

keep refining and viewing until they create a segment that follows the known task-related behavior. The number of sequences within the segment will then indicate the support for that behavior. If the sequences within a segment match the input data requirements for some specific technique, the segment can be exported for further analysis. For *Identify* tasks, analysts discover behaviors by developing hypotheses through viewing the data of segments, and testing them by refining and creating different segments.

This framework explicitly addresses the **cold-start problem** that bedevils many previous systems, where an analyst is required to run a query before seeing any results. It is difficult to know what to query when confronted with a completely blank view; ensuring a meaningful view at all times is the cornerstone of our approach to guide fruitful exploration.

2.4 Clickstream 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 sequences and segments. We compute a series of derived data attributes intended to reveal patterns, distributions, and outliers that connect with the analysts' informal sense of what behaviors are interesting. Specifically, we try to generate evocative attributes such that a series of constraints on these attributes could serve to define a segment that contains expected, unexpected, favorable, or unfavorable behavior.

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. The information associated with each action, referred to as **action attributes**, is a quantitative single **timestamp**, a categorical *action type*, a categorical *sessionid* identifier as described below, and a categorical **clientid** identifier indicating the specific user. A **sequence** is a time-ordered list of actions that are associated with a particular id over some specific time interval. The information associated with each sequence, referred to as **sequence attributes**, is derived from the action attributes within it, yielding the quantitative time range attributes of **sequence start**, **sequence end**, and **sequence**

duration. The **sequence action counts** are also quantitative attributes storing the total number of actions and also per-type counts for each of the action types.

A **session** is a default time frame often used in industry when logging sequences of events, where actions performed by a single user are grouped together until a gap of a specific amount of time has occurred. Our industry partner sets this time to 30 minutes. This one-size-fits-all **sessionid** cutoff leads to analysis challenges because there may be multiple interesting subsequences within each of these per-session sequences, if activities that are conceptually different for the site user are grouped together. Conversely, grouping sequences according to the per-session id will often lead to multiple sequences for each site user in the clickstream datasets typically studied by the analysts due to time scales ranging from days to months. Although the original clickstream data is partitioned by these arbitrary session boundaries, we can also derive per-user sequences where all of these are concatenated together in order to follow the behavior of a single clientid across longer periods of time.

The number of different raw **action types** recorded depends on the granularity of the tracking and ranges from tens to hundreds in the clickstream datasets studied by the analysts we interviewed. The core actions tracked include the important 4 e-commerce actions of adding to cart, removing from cart, searching, and purchasing; 3 site functionality actions of errors, loading mode, and offline mode; and the frequent action of viewing a page. Websites can have hundreds of unique pages but very few analysis tasks require distinguishing between every single page on a site. Sites are often built using a single **template** for an entire group of similar pages, so that the builders are only concerned with tens of templates rather than hundreds of individual pages. However, our inspection of template structures across a variety of sites showed that they are not necessarily consistent, hampering their utility for analysis. We derive a three-level categorical attribute **action hierarchy** that classifies actions into groups that provide appropriate levels of detail for the clickstream analysis tasks that we target, based on the recurring common elements in the site-specific templates. The actions in each level of the hierarchy are shown in Figure 2.2 through the *Action Transition Network* chart discussed in Chapter 4.1. All three levels include the full set of 4 e-commerce actions and 3 site actions. The **detailed actions** level, shown in Figure 2.2(a), also has a set of 17 page groupings

that roughly corresponds to the consistent templates across sites. The **mid-level actions**, Figure 2.2(b), aggregates the detailed page groups into 6 higher-level categories, and the coarsest **roll-up actions** level, Figure 2.2c, groups all page views together into a single aggregate action. 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. 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 which has been also adopted by some of the previous work [13, 20, 24, 27, 33, 37, 40, 45].

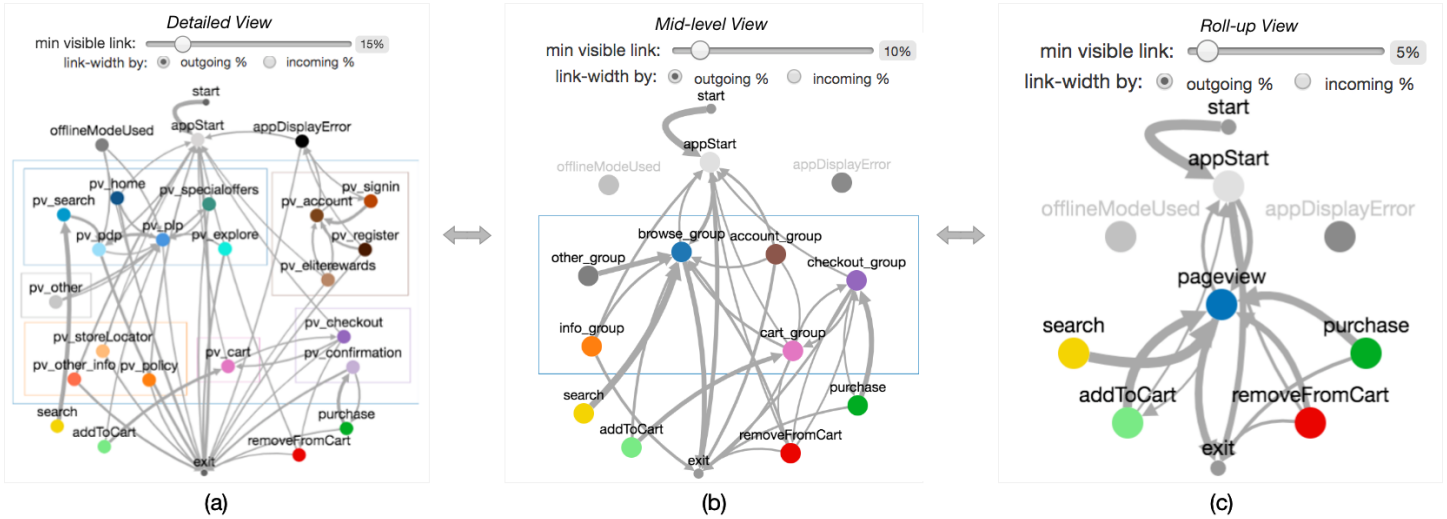

Figure 2.2: Action Transition Network showing all actions at each level of hierarchy: (a) Detailed, (b) Mid-level, (c) Roll-up.

We define a **segment** as any set of sequences. We derive many additional quantitative **segment attributes** and a derived network of *action transitions* based on the constituent sequence attributes and the union of all the actions contained within them. The simple quantitative attribute of **segment size** is simply the count of sequences that it contains. More complex computations use two intermediate tables. One table has one row per sequence, and is used to compute distributions across all countable **ranges** showing the number of sequences that fall into bins between the minimum and maximum values. The underlying sequence time ranges are used to create four temporal distributions: **duration**, **start hour**, **day of the week**, and **date**. The distributions of per-sequence **absolute action counts** are computed for all actions together, plus each action type individually. Finally, the table contains a column for each categorical type in the action hierarchy to record how many times it appears within the sequence, in order to compute the number of sequences that contain each action, both as an absolute number and a relative percentage compared to the segment size.

The other table consists of **action transition** trigrams for the combination of an action, the action before it, and the action after it. We count the number of times that this trigram occurs within the union of all actions within a segment's constituent sequences. The results are the **action transition networks** of all transitions between actions that occur in the segment. There is one network for each level of the action hierarchy, and the number of nodes at each level is constant (24 at the detailed level, 13 at the mid-level, and 7 at the roll-up level). The data-driven aspect that changes for each segment are the edges: the transition counts that serve as quantitative edge weights, which can be zero to indicate that there is no transition between two actions in any of the sequences. Transitions are unidirectional; links in both directions are computed.

The *refine* and *record* functions of the clickstream segment analysis framework that we propose in Section 2.3 and instantiate in Segmentifier leads to an **analysis paths tree** where each node is a segment. 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* used to refine it, as discussed in Section 4.2; this derived data constitutes a full record of the analysis process. We use this tree to derive the

quantitative attributes of **relative counts** for each segment: the percentages within a segment compared to its direct predecessor in the tree, and compared to the root segment of all sequences.

Chapter 3

Related Work

We frame the related work with respect to the clickstream segment analysis framework presented in Section 2.3 by grouping previous systems into categories according to the framework's functions. No previous system encompasses all aspects of our framework (*view*, *refine*, *export*, *abandon*, *conclude*, and *record*); Segmentifier is unique in providing an initial analysis process that can handle the size and noisiness of real-world clickstream data.

Post-Export 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 [8, 14, 18, 28, 40, 43, 48], pattern mining [10, 23, 26, 27, 33], and cohort comparison [29, 49]. 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 [43, 49], very small datasets [27, 33], or explicitly indicate a requirement for sampling [26]. Segmentifier is designed to maximize the effectiveness of these techniques by allowing analysts to first refine the data to meet their requirements.

View Sequences A great deal of the previous work focuses on different ways to *view* sequences but does not provide mechanisms to *view* segment-level attributes. Most earlier work simply displayed individual sequences, either along a horizon-

tal timeline [3, 19, 21, 35] or using footstep graphs [16]. Later work introduced ways to show many sequences at the same time by either grouping them [7] or using different visual encoding idioms such as networks [6, 34, 42], flow diagrams [31, 32, 44], disk trees [11], or icicle plots [22, 46]. Wang *et al*. [41] extended some of this work by introducing interaction techniques such as aligning sequences. However, without any ability to view *segment attributes* or to refine the data, discovering insights is extremely difficult.

Refine Visual query systems [12, 13, 20, 47] emphasize the need to *refine* segments by allowing analysts to filter data. Although they *record* refinement history using something analogous to our analysis paths tree, they provide either a limited or nonexistent *view* of the segment attributes and they do not provide a *view* of the sequences at all. Analysts thus have trouble knowing how to refine and often have to take a guess–and–check approach. Moreover, these systems focus on filtering, whereas Segmentifier also provides ways to *refine* through partitioning.

View and Refine Tools that emphasize the need to both iteratively *view* and *refine* data come closer to the functionality of Segmentifier. In addition to providing a view of both the segment attributes and the sequences, they incorporate the additional functionality of filtering by ordinal or categorical segment attributes [25, 38] or filtering by custom patterns of events [37, 39]. Wongsuphasawat *et al*. [45] incorporates both these filtering options but in two different tools focusing on very specific tasks of comparing datasets and funnel analysis.

The tools closest to Segmentifier incorporate versions of both filtering options and focus on exploring and simplifying event sequences. Session Viewer [24] has the ability to filter by segment attributes but only highlights sequences that follow custom event patterns. EventFlow [30] incorporates both filtering options but provides very minimal segment attributes. It does feature aggregation, but at the sequence level rather than the event level as supported by our action hierarchy. Neither has the ability to *record* the analytic process.

EventPad [9] 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 sequence attributes. In contrast, the design of Segmentifier promotes segment attributes to first-class citizens, a crucial factor in accommodating the scale of clickstream data.

Chapter 4

Segmentifier Interface

Segmentifier embodies the clickstream data analysis pipeline by supporting an iterative process of refining segments and viewing the results while recording all analysis paths. Its design capitalizes on the domain knowledge of e-commerce experts to refine raw sequences into clean, meaningful segments tied to consumer behavior that can lead to actionable insights, either directly or after being exported for different types of downstream analysis. The interface consists of three major views, as shown in Figure 1.1: the *Segment Inspector* view, the *Operation Manager* view, and the *Analysis Paths* view.

4.1 Segment Inspector View

The *Segment Inspector* view tackles the *View* part of the framework by showing all attributes of the *selected segment*, allowing analysts to identify new interesting behaviors that they can explore further or verify if the segment matches some behavior of interest. The view is divided into three sections: *Ranges*, *Actions*, and *Sequence Details*.

4.1.1 Ranges Attributes

The top *Ranges* section, shown in Figure 1.1C, consists of linked histograms showing the sequence count distributions for all of the ordered segment attributes. By having access to these distributions, instead of just the sequences as is the case in

most of the previous work, analysts can now immediately spot patterns and unusual trends of different attributes in the data. For example, 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.

4.1.2 Actions

The middle *Actions* section, shown in Figure 1.1C, focuses on the action hierarchy discussed in Section 2.4 through three different charts: *Contains Chart*, *Action Transition Network*, and *Selected Action Adjacency View*. The analyst chooses which level of the action hierarchy is currently active: *Detailed*, *Mid-level*, or *Roll-up*, which then updates the action types shown in each of these charts accordingly.

The *Contains Chart* shows the distribution of action 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 middle chart shows the derived *Action Transition Network* between the aggregated action types at the chosen level of the action hierarchy. The nodes are always at the same fixed positions at each hierarchy level; the transition count data derived from the selected segment drives which edges are drawn and at what thickness, for either the outgoing or incoming direction. The *min visible links* slider at the top filters out edges whose percentages fall below the specified threshold, allowing the analyst to trade off between detail and clutter. Figure 4.1 shows the visual encoding of the chart at each level of the hierarchy. The dimensions of the chart adapt according to the hierarchy level.

The spatial layout and color of the nodes emphasize the action hierarchy structure by providing consistency between views, in order to help analysts spot pat-

terns. Our goal was to test the utility of the novel derived action hierarchy data abstraction with a visual encoding that was well tuned to reveal its structure. Since the action hierarchy nodes do not change dynamically, we created the layouts and the color scheme manually with a few rounds of iterative refinement.

The layout provides spatial stability between the hierarchy levels, preserving the relative positions of the e-commerce actions at the bottom and site functionality actions at the top. The pageview region in the middle has detailed-level actions positioned close to others within the same mid-level group, and preserves relative positions of the mid-level groups. The design targets high information density for maximum label legibility, with minimal edge crossings. Similarly, the color palette is designed to preserve visual consistency between levels, with identical colors for the actions that appear across all levels and similar color families for all detailed-level actions within the same mid-level action group. These colors are used consistently across many elements representing actions in all three main views. The goal is maximum distinguishability between the full set of 24 colors that are visible simultaneously at the detailed level and some distinguishability between the similar colors, where the largest *mid-level* group, BROWSE_GROUP, has size 6. This requirement is particularly difficult due to the small size of the marks: circular nodes in these charts and the operation glyphs in the other views, and the small boxes in the *Sequence Details* section.

The third chart is the *Selected Action Adjacency View*, which supports further analysis of a single node selected from the full network in a butterfly view: the chosen action is in the middle, with all of the incoming nodes aligned on the left and the outgoing ones aligned on the right. The aligned nodes are flanked by stacked bar charts showing the proportions of their occurrence, using consistent coloring. The exact values appear in a tooltip on hover.

Some of previous work includes standalone charts similar to the *Selected Action Adjacency View*. We observed that our industry partners were manually creating simple versions of the *Action Transition Network*, which gives an overview of all possible action transitions. We provide automatic support for this reasoning process in a novel view that does not appear in any previous work. It solves the cold-start problem for the *Selected Action Adjacency View*, allowing analysts to decide which actions are worth exploring further.

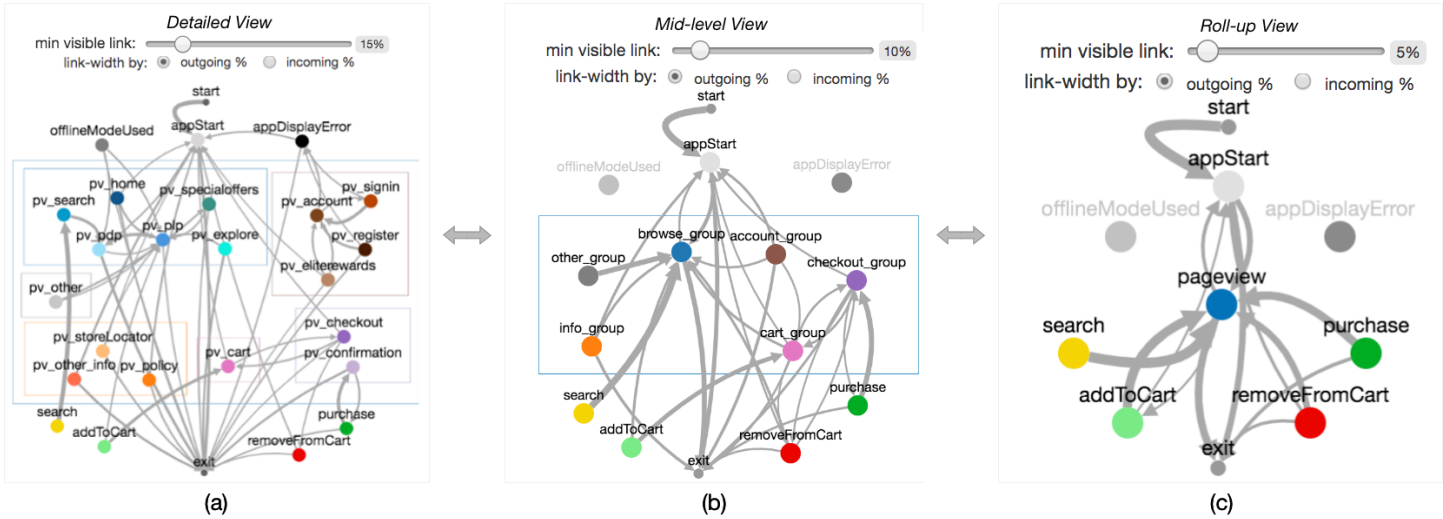

Figure 4.1: Action Transition Network showing all actions at each level of hierarchy: (a) Detailed, (b) Mid-level, (c) Roll-up. Layout and color scheme chosen to enforce hierarchical structure. The Mid-level and Roll-up charts are not to scale; they have been enlarged to show details.

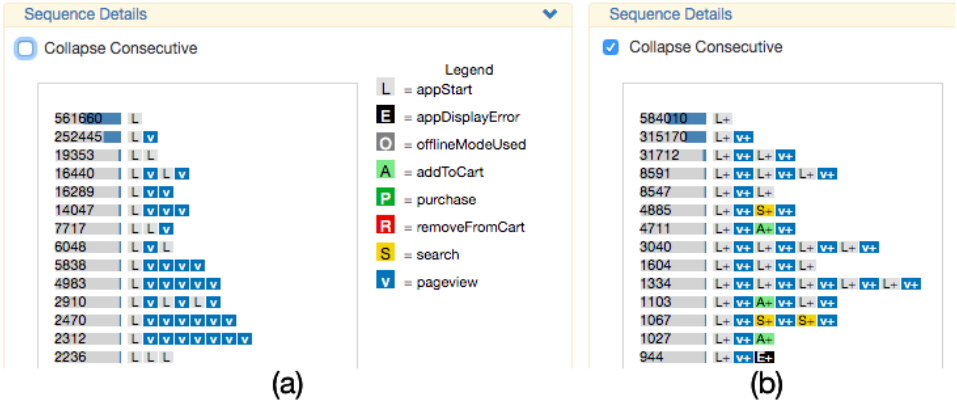

Figure 4.2: Sequence Details. All sequences in the selected segment shown as series of glyphs. Blue bar and number indicates frequency of occurrence of each sequence. (a) Sequences in selected segment. (b) Sequences after being collapsed.

4.1.3 Sequence Details

The bottom *Sequence Details* section is shown in Figure 4.2. Each row shows a sequence as series of boxes, one for each action, encoded according to the action types at the chosen hierarchy level. Their color is consistent with the nodes in the *Actions* section and they are labelled by a character that is unique for each action. This character set was manually chosen to be evocative and memorable for the static set of actions, following the same rationale as above for the layout and colors. Identical sequences are grouped together with only one row for each unique sequence, with a bar chart and numerical label on the left showing the counts. Sequences are sorted by frequency. The grouping and sorting yields a view that is much more scalable then showing the raw sequences alone while preserving the low level details required to spot patterns of user behavior. The *Collapse Consecutive* checkbox aggregates consecutive identical actions into a single action box, leading to a different grouping of sequences and visual encoding. Figure 4.2(b) shows the collapsed version of 4.2(a).

4.2 Operation Manager View

The *Operation Manager* view, shown in Figure 1.1A, tackles the *Refine* part of the framework. Analysts iteratively apply refinement **operations** to either **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*.

4.2.1 Operation Builder

The *Operation Builder* at the bottom has two tabs, one for ranges and one for actions, following the *Segment Inspector* structure.

Ranges Operation Builder The *Ranges Operation Builder* is automatically updated when any chart is selected in the *Segment Inspector Ranges* section, with a bar showing the full range of values for the relevant attribute. It can be filtered in the *Range Filter Builder* or partitioned in the *Range Partition Builder* using the widgets in the respective tab. Figure 4.3(a) and (b) show examples of an analyst using the *Range Filter Builder* to filter a subset of the sequences based on values of a different attributes. The Figure 4.3(a) example shows filtering down dates to the month of October using sliders, and in Figure 4.3(b) the analyst uses the NOT option to select all values that are outside regular working hours. Figure 4.3(c) and (d) show examples of an analyst using the *Range Partition Builder* to partition a segment into smaller segments based on the values of an attribute different attributes. Figure 4.3(c) shows the result of selecting the *purchase count* chart and adding three partition bars at values 1, 2, and 8, while Figure 4.3(d) adds only a single partition bar to compare sequences that lasted less versus more than an hour.

Actions Operation Builder The *Actions Operation Builder*, shown in Figure 4.4, supports investigation of how many sequences contain certain patterns of actions. This view 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

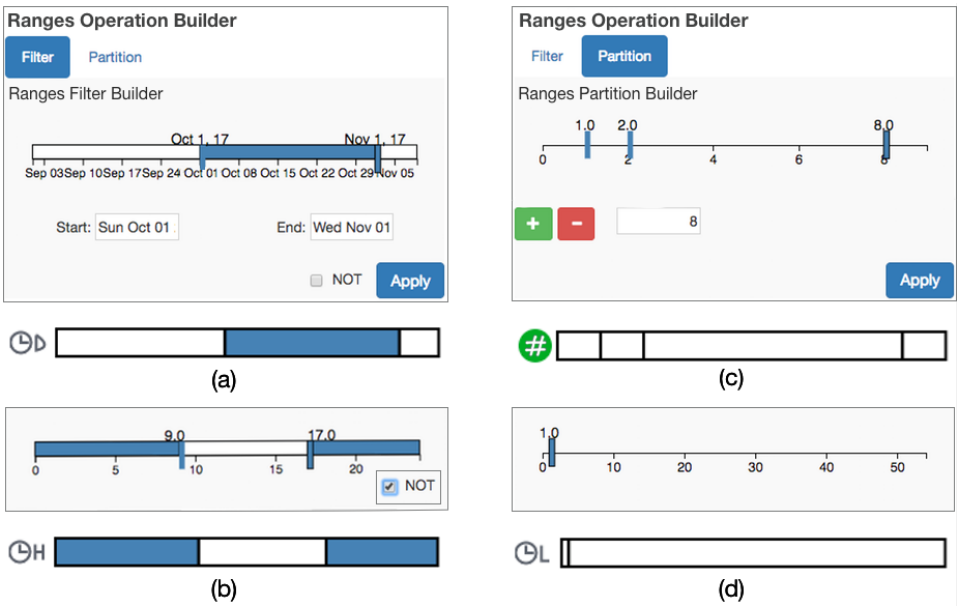

Figure 4.3: Ranges Operation Builders with corresponding operation glyphs. (a) filter operation using date chart. (b) NOT filter operation using hour chart. (c) partition operation with 3 partitions using purchase count chart. (d) partition operation with 1 partition using duration chart.

(shown with dashed lines), chooses whether or not to invert the whole pattern with the NOT button, and then applies the pattern normally or as a purchasing funnel. Selecting a sequence in the *Sequence Details* view automatically instantiates its pattern in the builder for potential editing, as does clicking on an already created *actions* operation node.

4.2.2 Operation Glyph Design

Operation glyphs are used to represent each operation in the *Operation Inspector* and *Analysis Paths* views. The rectangular glyphs are designed to concisely show the type of operation, the attribute used, and an indication of values selected.

Figure 4.3 shows four **ranges operation glyphs** with their corresponding builders. The icon on the left represents the source chart: time-related attributes have a clock symbol and a unique letter indicating the specific chart, and attributes related to action counts have a pound sign within a circle colored by the specific action. For

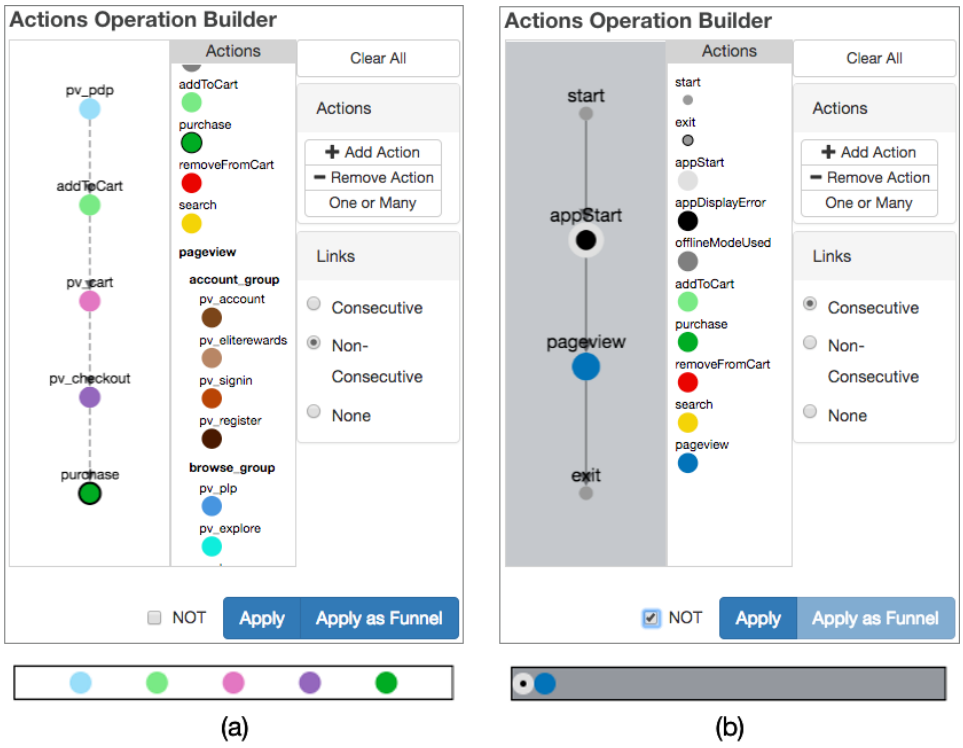

Figure 4.4: Actions Operation Builders with corresponding operation glyphs. (a) A non-consecutive action pattern filter at the Detailed action hierarchy level. (b) A consecutive action pattern NOT filter at the Roll-up level.

ranges filter operations shown in Figure 4.3(a) and 4.3(b), the blue shaded portion of the rectangle represents the user-selected filtered range compared to the full range of values for the specified attribute. For *ranges partition operations* shown in Figure 4.3(c) and 4.3(d), a line is drawn for every partition the analyst added. The position of the line represents the value of the partition compared to the full value range of the attribute.

Figure 4.4 shows two **actions operation glyphs** with their corresponding builders. Colored circles represent the actions, touching to show consecutive links and separated to show non-consecutive ones. Figure 4.4(b) shows a NOT filter represented with a dark gray background.

4.2.3 Operation Inspector

The top *Operation Inspector* section, shown in Figure 1.1A, 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 absolutely and compared to the previous and initial segments.

4.3 Analysis Paths View

The *Analysis Paths* view, visible in Figure 1.1B, tackles the *record* part of the framework. It keeps track of all conducted analysis paths through a graphical history tree of nodes for all created *operations*, represented by their glyphs, and segments represented by gray rectangles sized by their absolute sequence counts. Selecting a segment turns it blue, triggers a complete update of *Segment Inspector* view to reflect that segment's data, and updates the *Operation Inspector* to show the attribute constraints used to create the segment. The analyst can then refine it to create a new segment in the tree by building a new operation.

Chapter 5

Implementation and Pre-processing

Segmentifier was built in Javascript with D3.js [5], using crossfilter.js [4] 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 the 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.

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 hierar-

chy 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, as discussed in Section 2.4 and shown in Figure 2.2(a). We settled on this manual grouping by domain experts following Liu *et al*. [27], who concluded that this approach was more effective than the multiple automatic techniques they considered.

Chapter 6

Results

We show the effectiveness of Segmentifier in two ways, both using real-world datasets. First, we walk through a detailed usage scenario to illustrate the utility of the tool based on our own analysis of this dataset. Second, we summarize the insights found through a case study with an e-commerce industry clickstream analyst, showcasing the effectiveness of Segmentifier for a domain expert target user.

6.1 Usage Scenario

The first dataset comes from a live e-commerce site (CUST1). 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. The website was built with 62 unique site-specific templates that we mapped into our action hierarchy. This dataset is also used in Figures 4.2, 4.3, 4.4, and 6.1.

Through our own analysis using Segmentifier, we constructed a usage scenario that shows the utility of the tool and illustrates the type of insights that can be found. Figure 6.1 showcases this usage scenario step by step 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 actionable result.

A. Identify and filter out unexpected behavior (I1) The initial segment with 1 million sequences is loaded in as the root of the tree in the *Analysis Paths* view, and highlighted in blue showing that it is the selected segment (Figure 6.1A *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 (Figure 6.1 A *View*). Noting this situation as *unexpected behavior*, they explore further by looking at the sequences in *Sequence Details* (Figure 4.2a). They select the *Compress Consecutive* option to simplify the sequences (Figure 4.2b) and notice that over 58% of sequences only contain APPSTART actions (an action signifying when a site loads). The analyst determines that there must be an APPSTART tracking issue.

They filter out the *unexpected behavior* by selecting the sequence in *Sequence Details* which automatically creates the action pattern in the *Actions Operation Builder* (Figure 6.1A *Refine*): a consecutive path from START to APPSTART to EXIT with a black dot in the APPSTART to include any number of those actions consecutively. The analyst selects the NOT option and clicks *Apply*. The operation is recorded in the *Analysis Paths* view (Figure 6.1A *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 that 415,980 sequences are not affected by this behavior.

B. Identify unfavorable behavior (I1) The analyst selects the resulting segment (Figure 6.1B *Segment*) and all attributes in the *Segment Inspector* are updated. After deselecting the *Compress 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* (Figure 6.1B *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* (Figure 6.1B *Refine*) and applying the filter operation. The operation is recorded in the *Analysis Paths* view (Figure 6.1B *Record*).

C. Verify support for unfavorable behavior (V2) They select the resulting segment (Figure 6.1C *Segment*). To get more details, the analyst switches to the *De-*

tailed level of the hierarchy (Figure 6.1C *Refine*) which updates the charts in the Segment Inspector view. The *Contains Chart* (Figure 6.1C *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. Verify existence of expected behavior (V1) 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 (Figure 6.1D *Segment*) and create a pattern in the *Actions Operation Builder* (Figure 6.1D *Refine*) consisting 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 (Figure 6.1D *Record*).

E.1 Verify support of expected behavior (V2) They select the last resulting segment representing all sequences that contain the full purchasing funnel. The *Operation Inspector* shows a detailed breakdown of the series of operations applied to create the segment allowing them to determine the dropout percentage at each step of the purchasing funnel. They notice that 67% make it to the checkout step but never purchase (Figure 6.1E.1 *View*) and hypothesize that there may be an error in the checkout process preventing consumers from purchasing.

E.2 Identify more-fine grained behaviors (I3) In the *Ranges Attributes* section, the analyst creates the PV_PDP *count* chart (Figure 6.1E.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 (Figure 6.1E.2 *Refine*). A *partition node* is created in the *Analysis Paths* view with three resulting *segment nodes* (Figure 6.1 E.2 *Record*).

F. Verify support of more-fine grained behaviors (I2) After selecting the *partition operation node* (Figure 6.1F *Operation*), the *Operation Inspector* updates to show details of the resulting three segments (Figure 6.1F *View*): 23% of purchasers view one product page, 67% view between 2-10 pages, and 10% view between 10-44. The analyst decides to use this information to send targeted messages to consumers based on the type of purchasing behavior followed.

G. Verify favorable behavior (V1) They re-select the segment representing sequences that followed a full purchasing funnel (Figure 6.1G *Segment*) and notice in the *Sequence Details* view that the sequences follow a *favorable behavior* for exporting to the pattern mining technique (Figure 6.1 G *View*). The number of sequences is small and there is low variability between them satisfying the data requirements for this technique. They download the segment (Figure 6.1G *Export*) and end their analysis. Figure 1.1B shows the final analysis path tree.

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

6.2 Case Study

A clickstream analyst at the e-commerce company was preparing to give a one-month post website launch presentation to their commercial customer. Their goal was to use Segmentifier to discover actionable insights and suggestions for improvement that they could relay back to their customer. We conducted two separate chauffeured analyses with this domain expert, who was also one of the interviewees during the requirements analysis phase.

6.2.1 Analysis #1

This analysis used data from a different customer website (CUST2), with 25 site-specific 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 Figure 1.1 reflects the four analysis tasks completed during the two hour session. We summarize the insights found for each analysis here; the supporting materials contain further screenshots showing these findings.

A. Analyze purchasing behavior The analyst explored purchasing behavior by filtering out the sequences that contain a PURCHASE action. Two separate analysis paths emerged from this segment: 1) They explored the checkout flow and discovered that 12% of sessions contain more checkout pages than necessary to complete a purchase. 2) They explored the percentage of the purchasing funnel completed in one session and discovered that 30% of users actually exit the site and return later to complete their purchase.

B. Comparing morning vs night behavior The analyst investigated a hypothesis that the behavior of users would change based on the time of day so they specifically chose to investigate the difference of behavior between sessions that began at 7-9 am versus 7-9 pm. They compared the number of actions in each session and the percentage of sessions which completed the full purchasing funnel and found no significant differences.

C. Analyze add and remove from cart behavior The analyst then explored *adding to cart* and *removing from cart* behavior and discovered that 30% of users who removed from cart exited the session and most likely did not come back.

D. Analyze purchasing funnel Using *Apply as Funnel*, the analyst was able to easily determine the percentage of sessions that drop out at each step of the purchasing funnel. Using the information gathered in *Analysis A*, they were able to determine that about 20% of people who get to checkout will not end up purchasing.

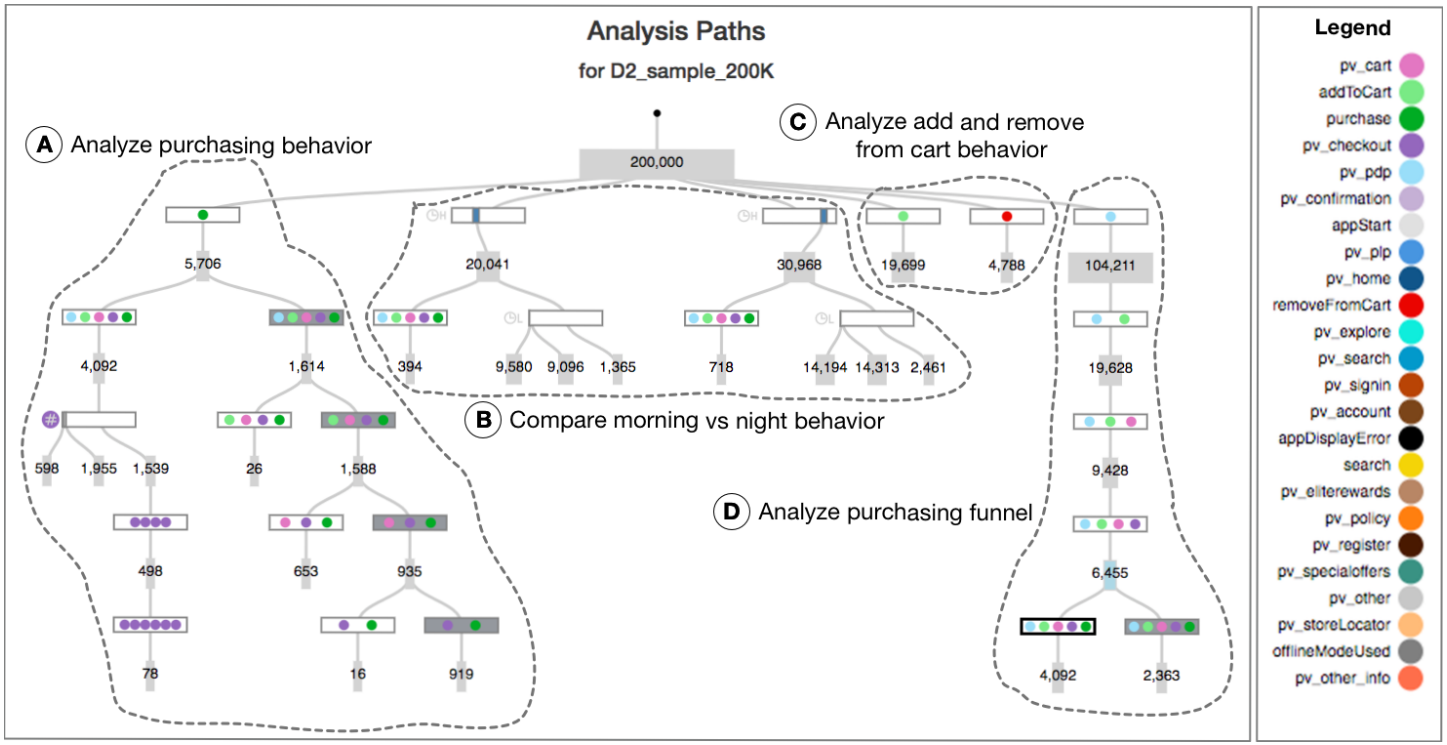

Figure 6.2: Analysis Paths View representing four analyses done in case study with domain expert.

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. Figure 1.1 shows an overview of the analysis.

With access to a new type of sequences, the domain expert decided to revisit some of the questions they explored in the previous analysis. They were interested in analyzing the reasons for dropping out of the purchasing funnel at the checkout stage and the effect of removing from cart. They discovered that 25% of clients that removed from cart at the checkout state, exited and never returned to the site to purchase. They also noticed that an APPSTART action was triggered every time before the PV_CART action which was an unusual behavior. Followup investigation determined a problem with the cart pages of the website.

The domain expert was also interested in analyzing the effect of a particular awards account, *myBeautyPage*, recently added on the website. Therefore, during the pre-processing step, the domain expert ensured that the *myBeautyPage* template name was the only template mapped to the PV_ELITEREWARDS action in the action hierarchy. The domain expert was able to easily discover that 1% of all clients signed up for the awards account and from those 27% made a purchase. They partitioned these clients to understand at what point of the purchasing funnel they signed up for the awards account and discovered it occurred mostly before adding to cart. Finally, they discovered that clients that accessed the awards account generally had longer sequences, signaling that they were more committed customers.

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.

Chapter 7

Discussion and Future Work

A key aspect of the Segmentifier design is that each possible refinement operation corresponds to one attribute constraint. Thus, 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. We saw the need for this kind of understandability after initial experiments using existing techniques such as clustering and pattern matching on the full dataset, which yielded incomprehensible results because of the high variability between sequences.

We carefully considered the tradeoff between power and simplicity for this application context. Many previous systems provide full support for regular expressions [9, 24], which are notoriously difficult for non-programmers to handle even when presented through a visual language, so we deliberately kept the scope of the Actions Operation Builder limited. Despite the request for more functionality from the domain expert, we still suspect that adding too much additional complexity on this front would make the system unusable for the untrained and extremely time-crunched analysts who we are targeting.

Segmentifier explicitly supports refinement through both filtering and partitioning. Although in theory partitioning could be achieved by the workaround of filtering multiple times, the utility of partitioning is that the analysis tree explicitly shows the split into multiple segments in a way that is a better match for a clickstream analyst's mental model and encourages subsequent analysis of the different cases. It is possible to compare segment sizes quickly using the *Analysis Paths*

view, but support for direct comparison between segments in detail is left for future work. An additional form of refinement is to transform segments by altering their sequences. In Segmentifier, transform operations occur implicitly through two choices: by switching between levels of the action hierarchy to change the action aggregation level and by selecting the *Compress Consecutive* option in the *Sequence Details* view to collapse consecutive, identical actions into one. Supporting further transformations explicitly would also be interesting future work.

Our primary focus was the agile and iterative development of Segmentifier's design, with only 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 in order 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. Improving the capacity and speed of our approach using computational infrastructure such as Hadoop and MapReduce [37] would be very interesting future work, to find the boundary between engineering and design issues.

Chapter 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 by allowing analysts to view, refine, record, export, abandon, and draw conclusions from iteratively developed segments. Understandable results can be found quickly by analysts with limited time and skills. Our task and data abstraction connects the mental model of clickstream analysts about interesting behavior to viewable characteristics of sequences and segments via a series of evocative derived data attributes. 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.

Bibliography

[1] Adobe Analytics. https://www.adobe.com/data-analytics-cloud. URL https://www.adobe.com/data-analytics-cloud. → page 1

[2] Google Analytics. https://analytics.google.com/. URL https://analytics.google.com/. → page 1

[3] W. Aigner, S. Miksch, B. Thurnher, and S. Biffl. PlanningLines: novel glyphs for representing temporal uncertainties and their evaluation. In *Proc. Intl. Conf. Information Visualisation (IV)*, pages 457–463, 2005. doi:10.1109/IV.2005.97. → page 16

[4] M. Bostock. crossfilter.js. https://github.com/square/crossfilter, 2013. → page 27

[5] M. Bostock, V. Ogievetsky, and J. Heer. D3 data-driven documents. *IEEE Trans. Visualization and Computer Graphics*, 17(12):2301–2309, 2011. ISSN 1077-2626. doi:10.1109/TVCG.2011.185. → page 27

[6] J. Brainerd and B. Becker. Case study: e-commerce clickstream visualization. In *Proc. IEEE Symp. Information Visualization (InfoVis)*, pages 153–156, 2001. doi:10.1109/INFVIS.2001.963293. → page 16

[7] M. Burch, F. Beck, and S. Diehl. Timeline trees: Visualizing sequences of transactions in information hierarchies. In *Proc. Working Conf. on Advanced Visual Interfaces (AVI)*, pages 75–82. ACM, 2008. doi:10.1145/1385569.1385584. → page 16

[8] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Visualization of navigation patterns on a web site using model-based clustering. In *Proc. Intl. Conf. Knowledge Discovery and Data Mining (KDD)*, pages 280–284. ACM, 2000. doi:10.1145/347090.347151. URL http://doi.acm.org/10.1145/347090.347151. → page 15

[9] B. C. M. Cappers and J. J. van Wijk. Exploring multivariate event sequences using rules, aggregations, and selections. *IEEE Trans. Visualization and Computer Graphics*, 24(1):532–541, 2018. ISSN 1077-2626. doi:10.1109/TVCG.2017.2745278. → pages 16, 39

[10] Y. Chen, P. Xu, and L. Ren. Sequence synopsis: Optimize visual summary of temporal event data. *IEEE Trans. Visualization and Computer Graphics*, 24(1):45–55, 2018. doi:10.1109/TVCG.2017.2745083. → page 15

[11] E. H. Chi. Improving web usability through visualization. *IEEE Internet Computing*, 6(2):64–71, 2002. ISSN 1089-7801. doi:10.1109/4236.991445. → page 16

[12] J. A. Fails, A. Karlson, L. Shahamat, and B. Shneiderman. A visual interface for multivariate temporal data: Finding patterns of events across multiple histories. In *Proc. IEEE Symp. Visual Analytics Science and Technology (VAST)*, pages 167–174, 2006. doi:10.1109/VAST.2006.261421. → page 16

[13] D. Gotz and H. Stavropoulos. DecisionFlow: Visual analytics for high-dimensional temporal event sequence data. *IEEE Trans. Visualization and Computer Graphics*, 20(12):1783–1792, 2014. doi:10.1109/TVCG.2014.2346682. → pages 11, 16

[14] S. Guo, K. Xu, R. Zhao, D. Gotz, H. Zha, and N. Cao. EventThread: Visual summarization and stage analysis of event sequence data. *IEEE Trans. Visualization and Computer Graphics*, 24(1):56–65, 2018. ISSN 1077-2626. doi:10.1109/TVCG.2017.2745320. → page 15

[15] J. Heer, J. Mackinlay, C. Stolte, and M. Agrawala. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *IEEE Trans. Visualization and Computer Graphics*, 14(6):1189–1196, 2008. ISSN 1077-2626. doi:10.1109/TVCG.2008.137. → page 8

[16] I. hsien Ting, C. Kimble, and D. Kudenko. Visualizing and classifying the pattern of users browsing behavior for website design recommendation. In *Proc. Intl. ECML/PKDD Workshop on Knowledge Discovery in Data Stream*, pages 101–102, 2004. → page 16

[17] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: Interactive visual specification of data transformation scripts. In *Proc. ACM Conf. on Human Factors in Computing Systems (CHI)*, pages 3363–3372. ACM, 2011. doi:10.1145/1978942.1979444. → page 6

[18] S. Kannappady, S. P. Mudur, and N. Shiri. Visualization of web usage patterns. In *Proc. Intl. Database Engineering and Applications Symposium (IDEAS)*, pages 220–227, 2006. doi:10.1109/IDEAS.2006.52. → page 15

[19] G. M. Karam. Visualization using timelines. In *Proc. ACM SIGSOFT Intl. Symp. Software Testing and Analysis (ISSTA)*, pages 125–137. ACM, 1994. ISBN 0-89791-683-2. doi:10.1145/186258.187157. → page 16

[20] J. Krause, A. Perer, and H. Stavropoulos. Supporting iterative cohort construction with visual temporal queries. *IEEE Trans. Visualization and Computer Graphics*, 22(1):91–100, 2016. ISSN 1077-2626. doi:10.1109/TVCG.2015.2467622. → pages 11, 16

[21] M. Krstajic, E. Bertini, and D. Keim. CloudLines: Compact display of event episodes in multiple time-series. *IEEE Trans. Visualization and Computer Graphics*, 17(12):2432–2439, 2011. ISSN 1077-2626. doi:10.1109/TVCG.2011.179. → page 16

[22] J. B. Kruskal and J. M. Landwehr. Icicle plots: Better displays for hierarchical clustering. *The American Statistician*, 37(2):162–168, 1983. doi:10.2307/2685881. → page 16

[23] B. C. Kwon, J. Verma, and A. Perer. Peekquence: Visual analytics for event sequence data. In *Proc. ACM SIGKDD Workshop on Interactive Data Exploration and Analytics (IDEA)*, 2016. → page 15

[24] H. Lam, D. Russell, D. Tang, and T. Munzner. Session Viewer: Visual exploratory analysis of web session logs. In *Proc. IEEE Symp. Visual Analytics Science and Technology (VAST)*, pages 147–154, 2007. doi:10.1109/VAST.2007.4389008. → pages 11, 16, 39

[25] J. Lee, M. Podlaseck, E. Schonberg, and R. Hoch. Visualization and analysis of clickstream data of online stores for understanding web merchandising. 5: 59–84, 2001. → page 16

[26] Z. Liu, B. Kerr, M. Dontcheva, J. Grover, M. Hoffman, and A. Wilson. CoreFlow: Extracting and visualizing branching patterns from event sequences. *Computer Graphics Forum*, 36(3):527–538, 2017. ISSN 0167-7055. doi:10.1111/cgf.13208. URL https://doi.org/10.1111/cgf.13208. → page 15

[27] Z. Liu, Y. Wang, M. Dontcheva, M. Hoffman, S. Walker, and A. Wilson. Patterns and sequences: Interactive exploration of clickstreams to

understand common visitor paths. *IEEE Trans. Visualization and Computer Graphics*, 23(1):321–330, 2017. ISSN 1077-2626. doi:10.1109/TVCG.2016.2598797. → pages 6, 11, 15, 28

[28] A. Makanju, S. Brooks, A. N. Zincir-Heywood, and E. E. Milios. LogView: Visualizing event log clusters. In *Proc. Conf. on Privacy, Security and Trust*, pages 99–108, 2008. doi:10.1109/PST.2008.17. → page 15

[29] S. Malik, B. Shneiderman, F. Du, C. Plaisant, and M. Bjarnadottir. High-volume hypothesis testing: Systematic exploration of event sequence comparisons. *ACM Trans. Interact. Intell. Syst.*, 6(1):9:1–9:23, 2016. ISSN 2160-6455. doi:10.1145/2890478. → page 15

[30] M. Monroe, R. Lan, H. Lee, C. Plaisant, and B. Shneiderman. Temporal event sequence simplification. *IEEE Trans. Visualization and Computer Graphics*, 19(12):2227–2236, 2013. ISSN 1077-2626. doi:10.1109/TVCG.2013.200. → page 16

[31] P. Mui. Introducing flow visualization: visualizing visitor flow. https://analytics.googleblog.com/2011/10/introducing-flow-visualization.html, 2011. URL https://analytics.googleblog.com/2011/10/introducing-flow-visualization.html. → page 16

[32] A. Perer and D. Gotz. Data-driven exploration of care plans for patients. In *Extended Abstracts of ACM Conf. Human Factors in Computing Systems (CHI)*, pages 439–444. ACM, 2013. doi:10.1145/2468356.2468434. → page 16

[33] A. Perer and F. Wang. Frequence: Interactive mining and visualization of temporal frequent event sequences. In *Proc. Intl. Conf. on Intelligent User Interfaces (IUI)*, pages 153–162, 2014. ISBN 978-1-4503-2184-6. doi:10.1145/2557500.2557508. URL http://doi.acm.org/10.1145/2557500.2557508. → pages 11, 15

[34] J. Pitkow and K. A. Bharat. Webviz: A tool for world-wide web access log analysis. In *Proc. Intl. Conf. World Wide Web (WWW)*, pages 271–277, 1994. → page 16

[35] C. Plaisant, R. Mushlin, A. Snyder, , J. Li, D. Heller, and B. Shneiderman. Lifelines: using visualization to enhance navigation and analysis of patient records. In *Proc. Symp. American Medical Informatics Association (AMIA)*, pages 76–80, 1998. → page 16

[36] A. Sarikaya, E. Zgraggen, R. Deline, S. Drucker, and D. Fisher. Sequence pre-processing: Focusing analysis of log event data. In *IEEE VIS The Event Event: Temporal & Sequential Event Analysis Workshop*, 2016. URL https://pdfs.semanticscholar.org/1d65/8b25094f6215c681208945f7103f3ea2d030.pdf. → page 6

[37] Z. Shen, J. Wei, N. Sundaresan, and K. L. Ma. Visual analysis of massive web session data. In *Proc. IEEE Symp. Large Data Analysis and Visualization (LDAV)*, pages 65–72, 2012. doi:10.1109/LDAV.2012.6378977. → pages 11, 16, 40

[38] C. Shi, S. Fu, Q. Chen, and H. Qu. VisMOOC: Visualizing video clickstream data from massive open online courses. In *Proc. IEEE Symp. Pacific Visualization (PacificVis)*, pages 159–166, 2015. doi:10.1109/PACIFICVIS.2015.7156373. → page 16

[39] K. Vrotsou, J. Johansson, and M. Cooper. ActiviTree: Interactive visual exploration of sequences in event-based data using graph similarity. *IEEE Trans. Visualization and Computer Graphics*, 15(6):945–952, 2009. doi:10.1109/TVCG.2009.117. → page 16

[40] G. Wang, X. Zhang, S. Tang, H. Zheng, and B. Y. Zhao. Unsupervised clickstream clustering for user behavior analysis. In *Proc. ACM Conf. on Human Factors in Computing Systems (CHI)*, pages 225–236, 2016. ISBN 978-1-4503-3362-7. doi:10.1145/2858036.2858107. URL http://doi.acm.org/10.1145/2858036.2858107. → pages 11, 15

[41] T. D. Wang, C. Plaisant, A. J. Quinn, R. Stanchak, S. Murphy, and B. Shneiderman. Aligning temporal data by sentinel events: Discovering patterns in electronic health records. In *Proc. ACM Conf. Human Factors in Computing Systems (CHI)*, pages 457–466. ACM, 2008. doi:10.1145/1357054.1357129. → page 16

[42] S. J. Waterson, J. I. Hong, T. Sohn, J. A. Landay, J. Heer, and T. Matthews. What did they do? understanding clickstreams with the WebQuilt visualization system. In *Proc. Working Conf. on Advanced Visual Interfaces (AVI)*, pages 94–102. ACM, 2002. doi:10.1145/1556262.1556276. → page 16

[43] J. Wei, Z. Shen, N. Sundaresan, and K. L. Ma. Visual cluster exploration of web clickstream data. In *Proc. IEEE Conf. Visual Analytics Science and Technology (VAST)*, pages 3–12, 2012. doi:10.1109/VAST.2012.6400494. → page 15

[44] K. Wongsuphasawat and D. Gotz. Exploring flow, factors, and outcomes of temporal event sequences with the outflow visualization. *IEEE Trans. Visualization and Computer Graphics*, 18(12):2659–2668, 2012. ISSN 1077-2626. doi:10.1109/TVCG.2012.225. → page 16

[45] K. Wongsuphasawat and J. Lin. 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)*, pages 113–122, 2014. doi:10.1109/VAST.2014.7042487. → pages 11, 16

[46] K. Wongsuphasawat, J. A. Guerra Gómez, C. Plaisant, T. D. Wang, M. Taieb-Maimon, and B. Shneiderman. LifeFlow: Visualizing an overview of event sequences. In *Proc. ACM Conf. Human Factors in Computing Systems (CHI)*, pages 1747–1756. ACM, 2011. doi:10.1145/1978942.1979196. → page 16

[47] E. Zgraggen, S. M. Drucker, D. Fisher, and R. DeLine. (s,Qu)Eries: Visual regular expressions for querying and exploring event sequences. In *Proc. ACM Conf. on Human Factors in Computing Systems (CHI)*, pages 2683–2692, 2015. ISBN 978-1-4503-3145-6. doi:10.1145/2702123.2702262. URL http://doi.acm.org/10.1145/2702123.2702262. → page 16

[48] X. Zhang, H.-F. Brown, and A. Shankar. Data-driven personas: Constructing archetypal users with clickstreams and user telemetry. In *Proc. ACM Conf. on Human Factors in Computing Systems (CHI)*, pages 5350–5359. ACM, 2016. doi:10.1145/2858036.2858523. URL http://doi.acm.org/10.1145/2858036.2858523. → page 15

[49] J. Zhao, Z. Liu, M. Dontcheva, A. Hertzmann, and A. Wilson. MatrixWave: Visual comparison of event sequence data. In *Proc ACM Conf. Human Factors in Computing Systems (CHI)*, pages 259–268, 2015. ISBN 978-1-4503-3145-6. doi:10.1145/2702123.2702419. URL http://doi.acm.org/10.1145/2702123.2702419. → page 15

Appendix A

Supporting Materials

The following supporting material provides additional screenshots of Segmentifier to further clarify the visual encodings and linked-view interaction, and to further document how the clickstream analysis framework is instantiated through the interface. We show full screenshots of every step of Analysis #1 and Analysis #2 of the case study with our domain expert. Figures A.1 to A.12 describe Analysis #1 where 200,000 per-session sequences are loaded into the tool, each representing one user *session*. Figures A.13 to A.21 describe Analysis #2 where 200,000 per-client sequences are loaded each representing all actions performed by that *user*.

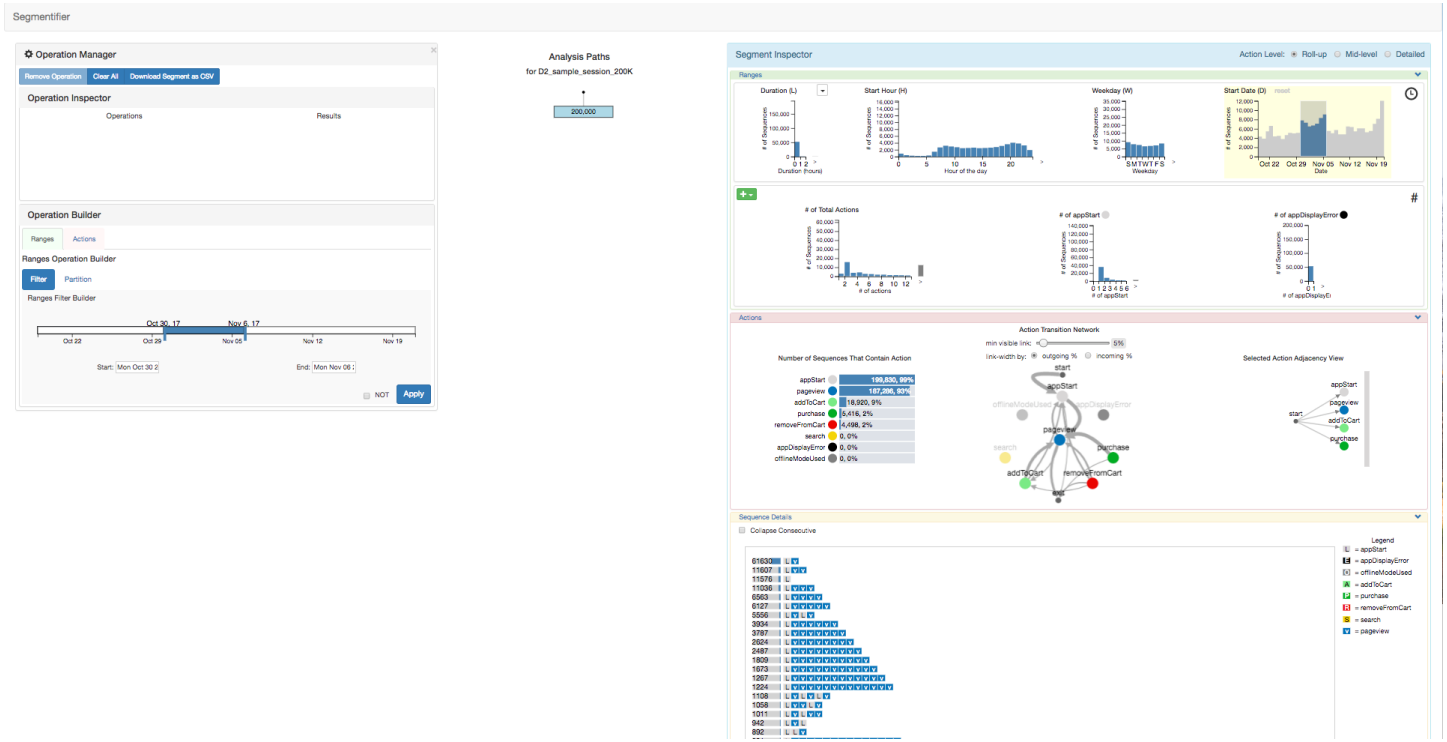

Figure A.1: Case Study Analysis #1 (CS-A1): The initial state of the interface when 200,000 sequences, each representing a user session, are loaded in. The initial segment including the 200,000 sequences is shown as a blue rectangle in the *Analysis Paths* view in the middle. The *Segment Inspector* view on the right shows information about the underlying data of the segment including ordinal and categorical segment attributes and the actual sequences. The analyst has thus far chosen to show only two per-action histograms, for APPSTART and APPDISPLAYERROR.

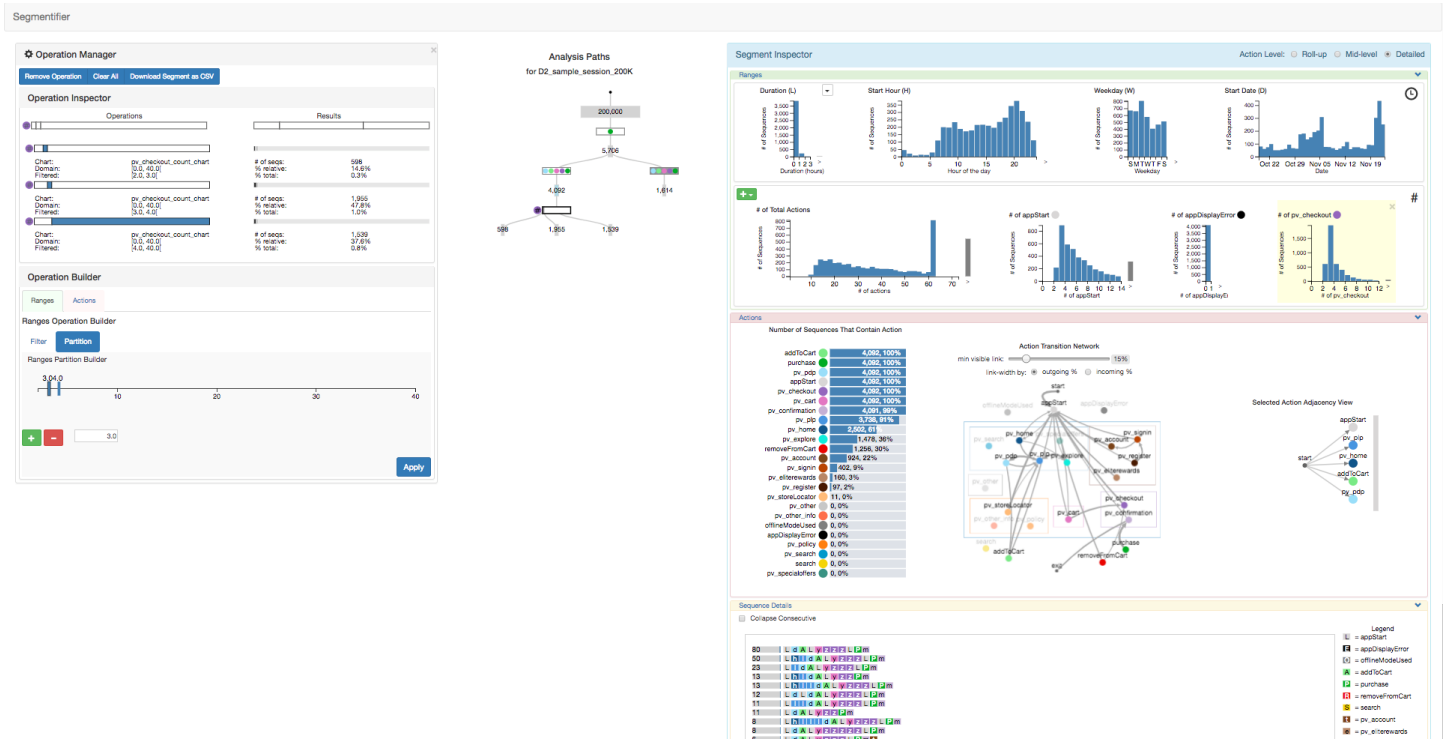

Figure A.2: The analyst first narrows down to purchasers by filtering segments that contain a PURCHASE action and from those they filter out segments that follow all five actions required to purchase an item and those that do not. They add the CHECKOUT chart to the *Ranges* section on the right, notice that 3 pages seem to be required to properly check out from this page, and click on that chart to start creating a partition operation in the *Ranges Operation Builder* on the left. They build a partition operation to determine which sequences had fewer than, equal to, or more than that expected number of checkout pages and apply it to the segment containing all five purchasing actions. The Operation Inspector view on the left shows that 15% of sessions contain fewer, 48% equal to, and 37% more.

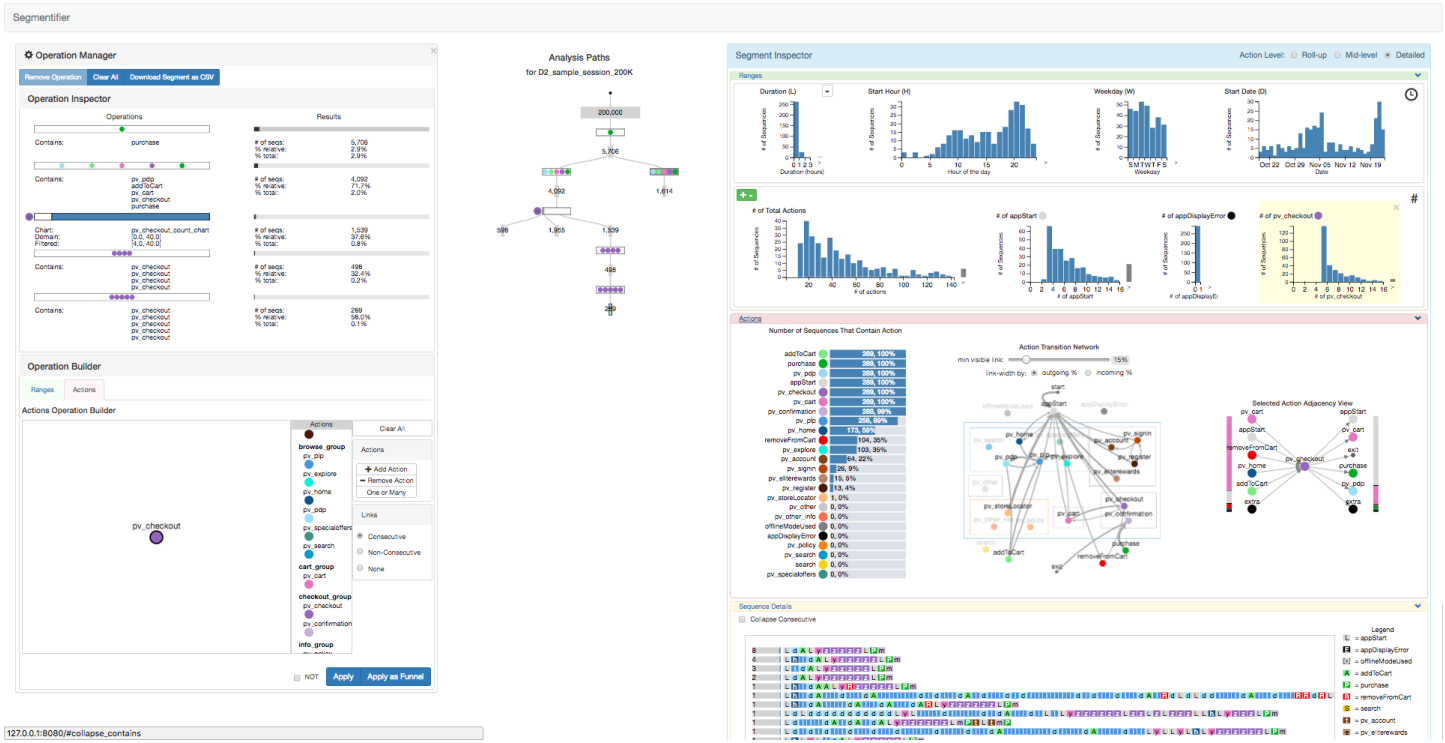

Figure A.3: The analyst decides to further analyze those who viewed more checkout pages than expected by determining those that viewed four and then five in a row. They used the *Actions Operation Builder* to specify that many consecutive actions, as we can tell from the glyphs in *Operations Inspector* in the upper left showing the path to the selected segment at the bottom of this analysis path. It contains only 289 sequences, and the raw sequences in the *Sequence Detail* view on the bottom right are quite long for this uncommon behavior.

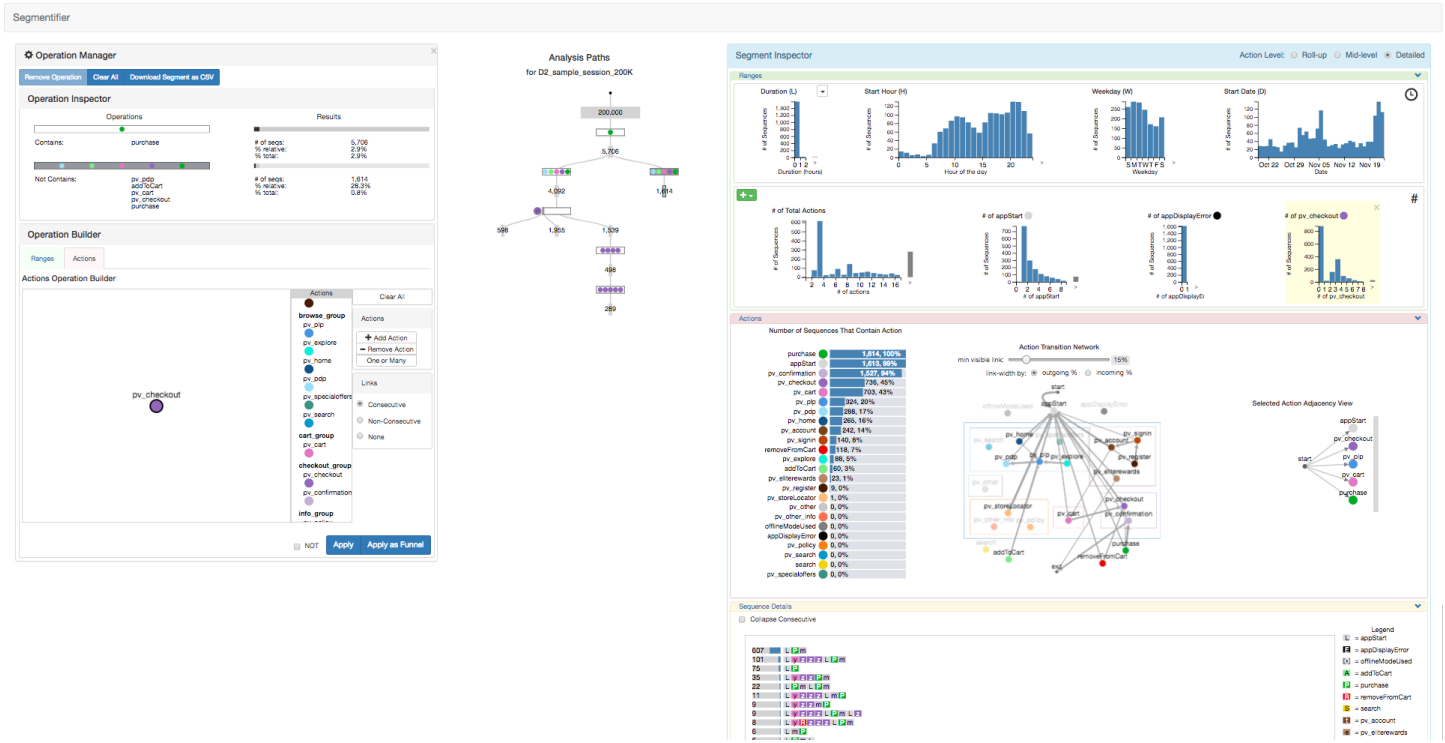

Figure A.4: The analyst decides to go back up the tree to select the segment on the right side of the *Analysis Paths* view to check the number of sessions that include a purchase action but not all five actions of the purchasing funnel. The Operation Inspector shows them that 28% of users do not complete a purchase in one session but return in a later session to do so.

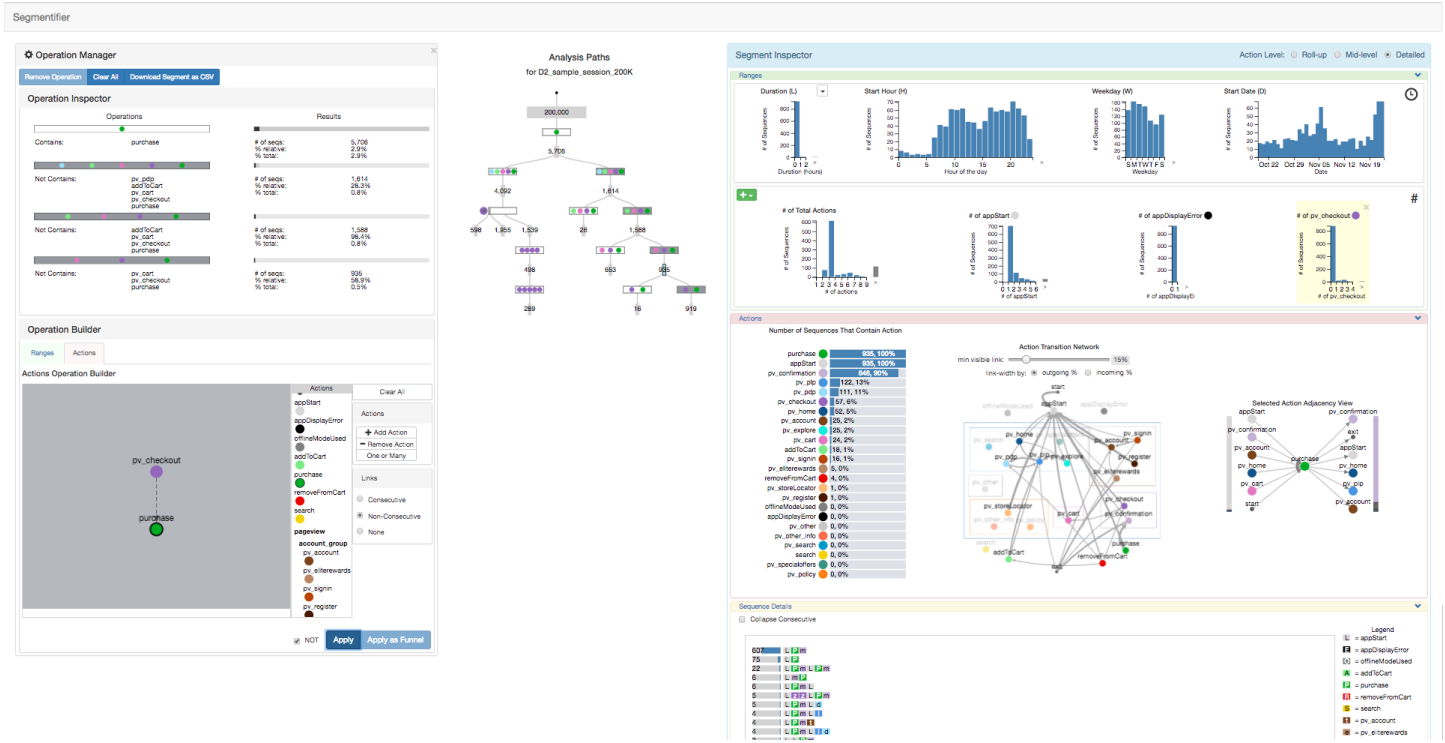

Figure A.5: The analyst continues their inquiry by investigating how many of the five actions in the purchasing funnel are completed in one session through multiple rounds of splits using the *Actions Operation Builder* to indicate which actions are not contained in the segment. They notice that 16% of users that make it to checkout must have returned after a previous session, since they PURCHASE without having conducted the other required actions in this session.

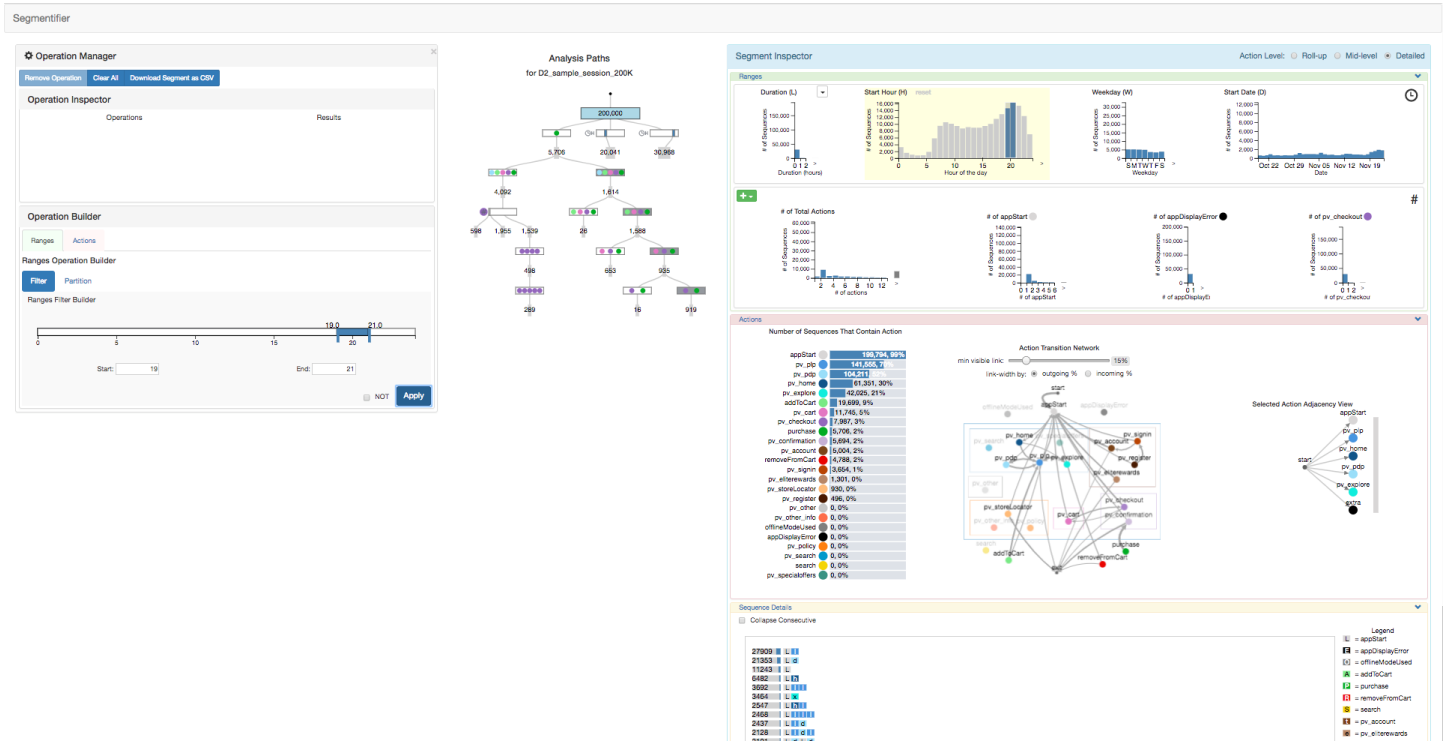

Figure A.6: The analyst decides to investigate a hypothesis that behavior changes depending on the time of day so they return to the root segment of the *Analysis Paths* tree to start new analysis paths. They click on the *Start Hour* chart from the *Ranges* section of the *Segment Inspector* view in the upper right and select a two-hour segment from 7 to 9am, which automatically populates the *Filter* tab of the *Ranges Operation Builder* accordingly, and they hit the *Apply* button to create that new segment. They then do the same for the 7-9pm range, and they see there are more such sequences: around 31K at night vs. around 20K in the morning.

Figure A.7: They first investigate the percentage of sequences that contain the full purchasing funnel with the hypothesis that users spend more time browsing and purchasing at night compared to the morning when users are usually commuting to work. However, using the *Operation Inspector*, they discover that the results are similar: although the absolute counts differ, the relative proportion of around 2% of sessions is the same for both time ranges.

Figure A.8: They then investigate the number of actions in the sequences with the similar hypothesis that users may have longer sessions at night after dinner than in the morning commute time slot. By building partition operations and applying them to both segments, using the *Operation Inspector*, they discover a similar disconfirmatory result that the size of the partitions are also similar between both times.

Figure A.9: The analyst decides to investigate sequences that contain an ADDTOCART action. However, after inspecting the data through the *Segment Inspector*, they did not discover anything worth exploring further and they abandon this analysis path.

Figure A.10: The analyst also decides to investigate sequences that contain a REMOVEFROMCART action. Using the *Selected Action Adjacency View* on the right, they notice that 18% of the time this action results in the end of a session.

Figure A.11: The analyst decides to analyze the purchasing funnel by building the pattern of five actions in the *Actions Operation Builder* on the left and clicking *Apply as Funnel*. The result is shown in the *Analysis Paths View*. After selecting the final resulting segment, the details of the dropout of each step is shown in the *Operation Inspector* on the top left.

Figure A.12: The analyst decides to investigate the sequences that got to the checkout part of the purchasing funnel but did not purchase. The *Operation Inspector* shows that this occurs 37% of the time. Using the previously found insight in Figure A.5 that 16% of users that check out are returning from a previous session, the analyst hypothesizes that about 21% of users that get to checkout never come back to purchase.

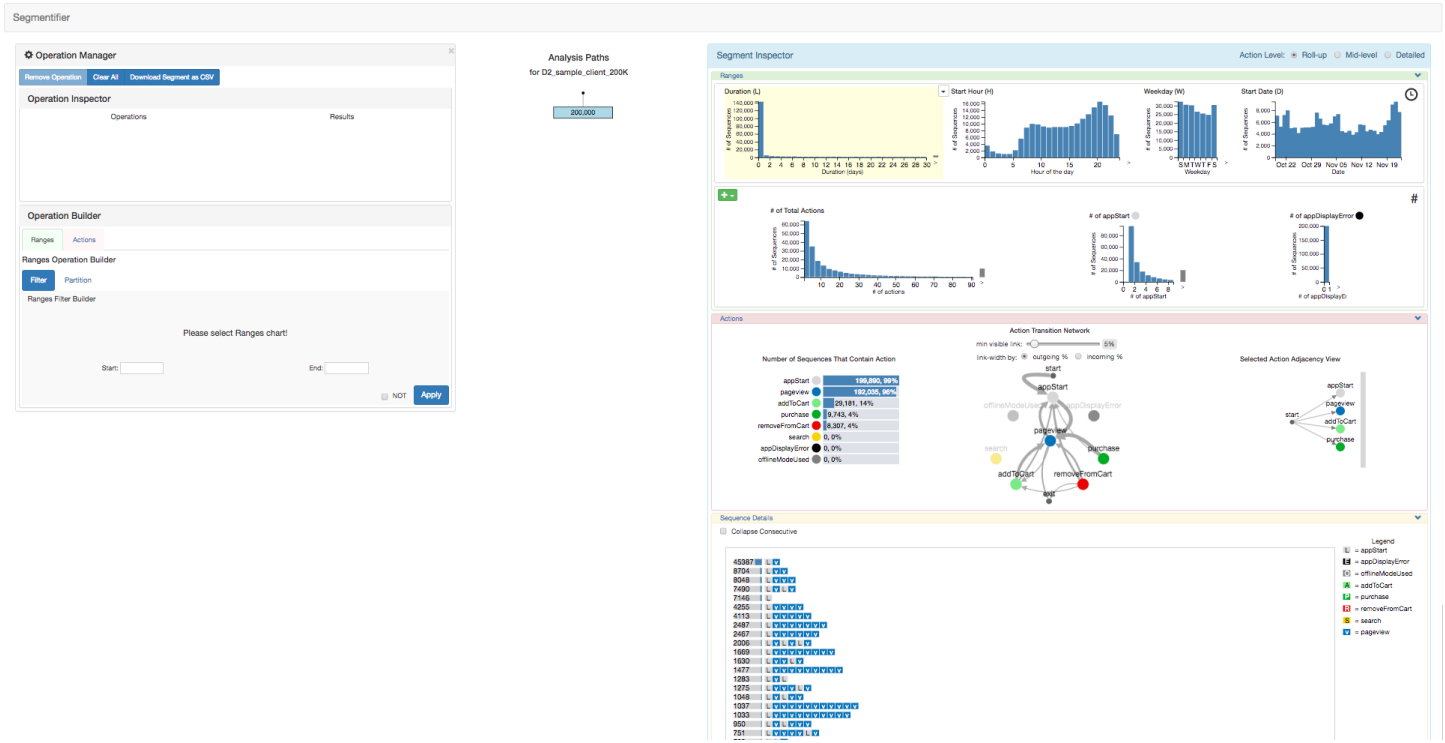

Figure A.13: Case Study Analysis #2: The initial state of the interface when 200,000 per-client sequences are first loaded, representing all actions performed by user over the entire dataset time window. The *Duration* histogram axis now extends to 30 days with sharp dropoff, in contrast to the Analysis #1 segments where this histogram was usually capped at between 15 and 20 minutes with a much more uniform distribution. The *Action Level* at the top right is set to the default of *Roll-up*.

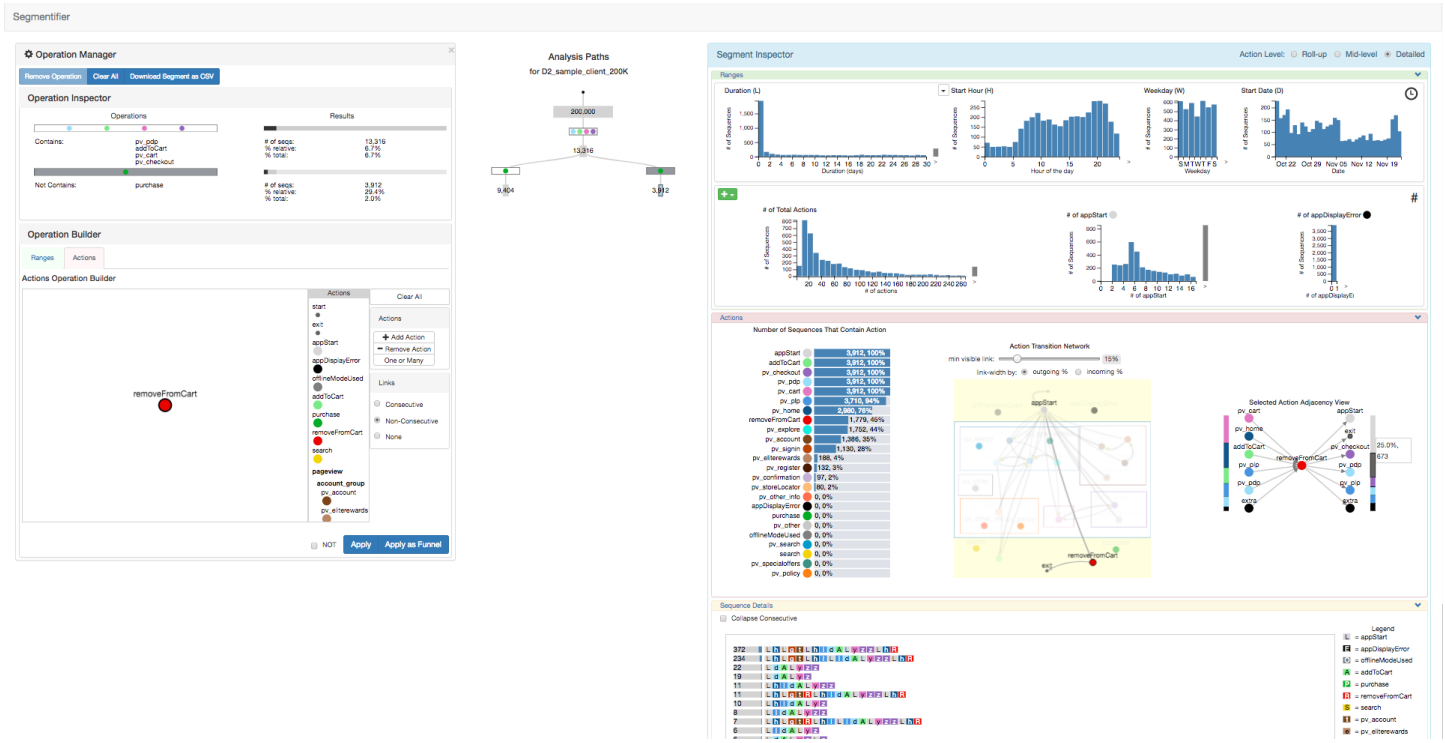

Figure A.14: The analyst switches to the *Detailed* Action Level to get further details about pages viewed. They investigate users that make it to the checkout but do not purchase. The *Select Action Adjacency View* on the right shows that 25% of those users that REMOVEFROMCART leave the site and never return.

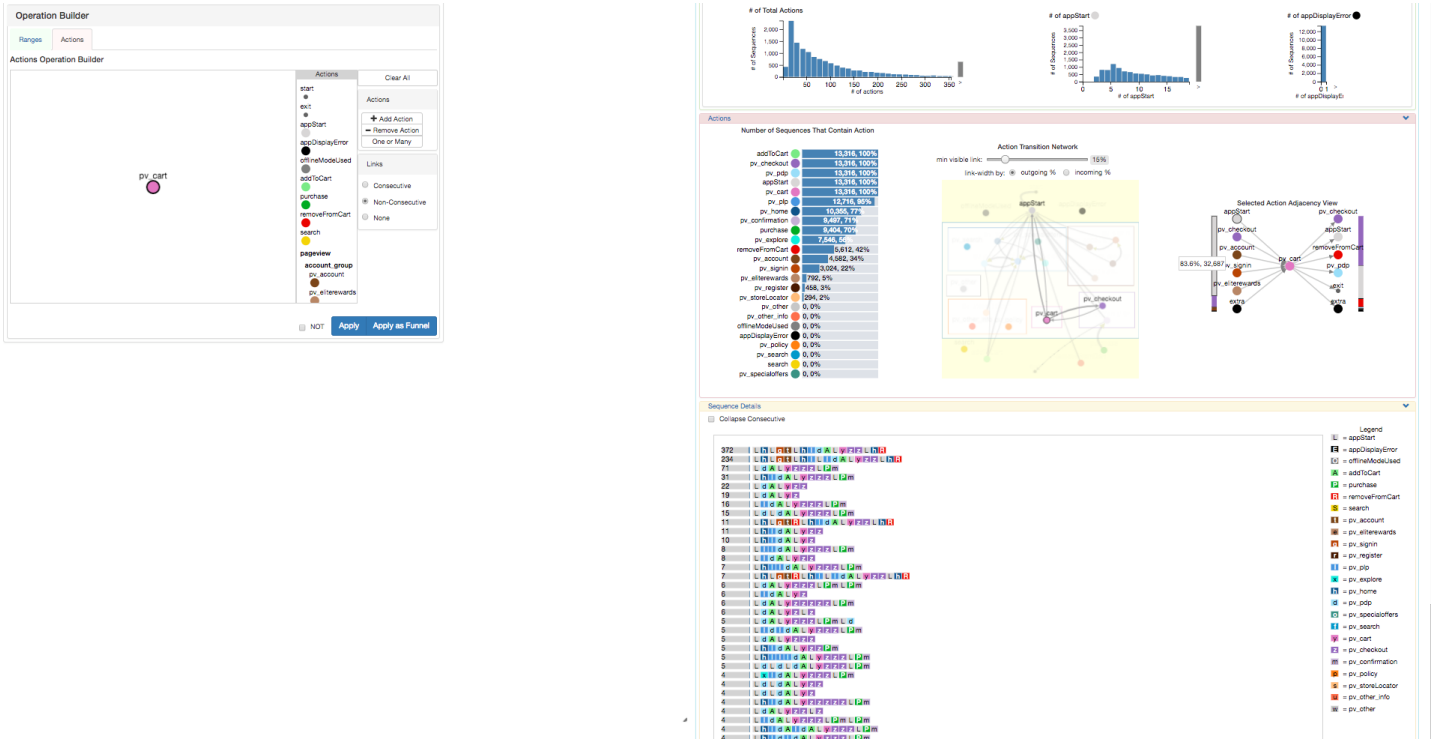

Figure A.15: The analyst scrolls down the *Segment Inspector* view on the right and notices in the *Sequence Details* view at the bottom that an APPSTART action represented by the gray glyph occurs often before the CART action represented by the pink glyph. They confirm their observation using the *Selected Action Adjacency Chart* which shows that 84% of the time an APPSTART action is generated before entering the cart. Followup investigation determined a problem with the cart pages of the website.

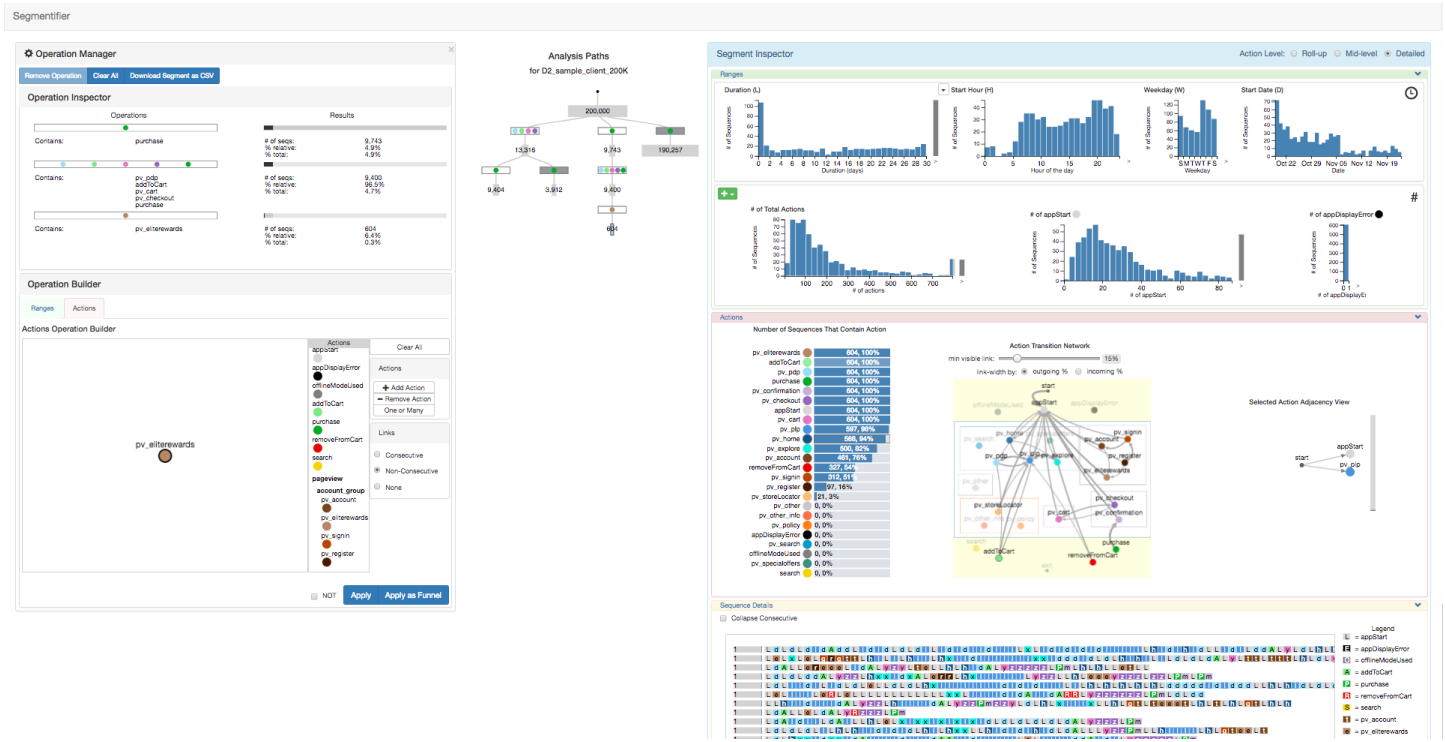

Figure A.16: The analyst wants to investigate the impact of a new awards account whose pages are stored in the ELITEREWARDS pageview action. They first look at the percentage of purchasers who signed up and discover by looking at the *Operation Inspector* that it is 6% of users.

Figure A.17: The analyst investigates at which point of the purchasing funnel the users accessed the awards account by building and filtering by patterns of actions in the *Actions Operation Builder* with the awards account action inserted at three different parts of the funnel. They discover based on the sizes of the three resulting segments that it is accessed most frequently after ADDTOCART.

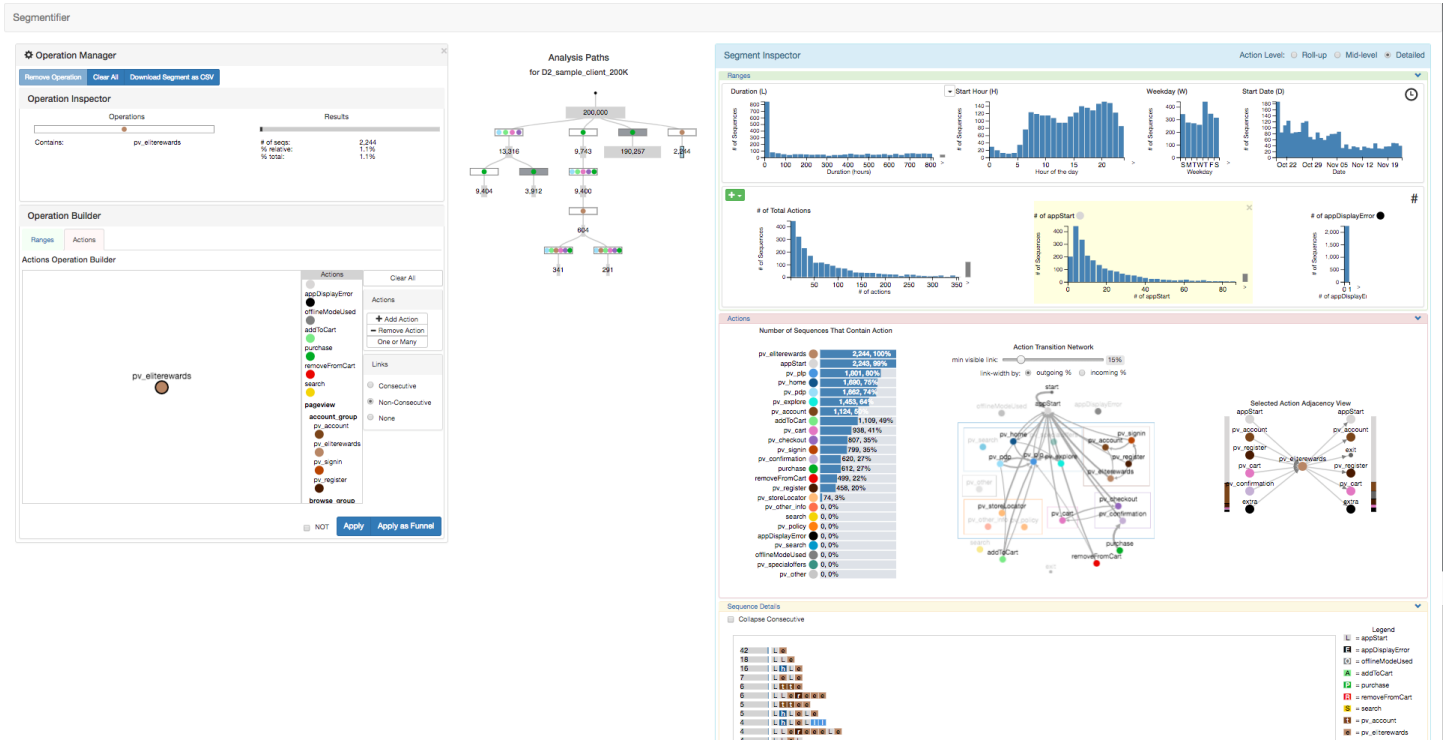

Figure A.18: The analyst creates a new segment with sequences containing the ELITEREWARDS action and uses the *Operation Inspector* view to discover that 1% of users have signed up for the awards account.

Figure A.19: They investigate further by filtering sequences that contain a PURCHASE action and those that do not, discovering that 27% of users who access the rewards account end up making a purchase.

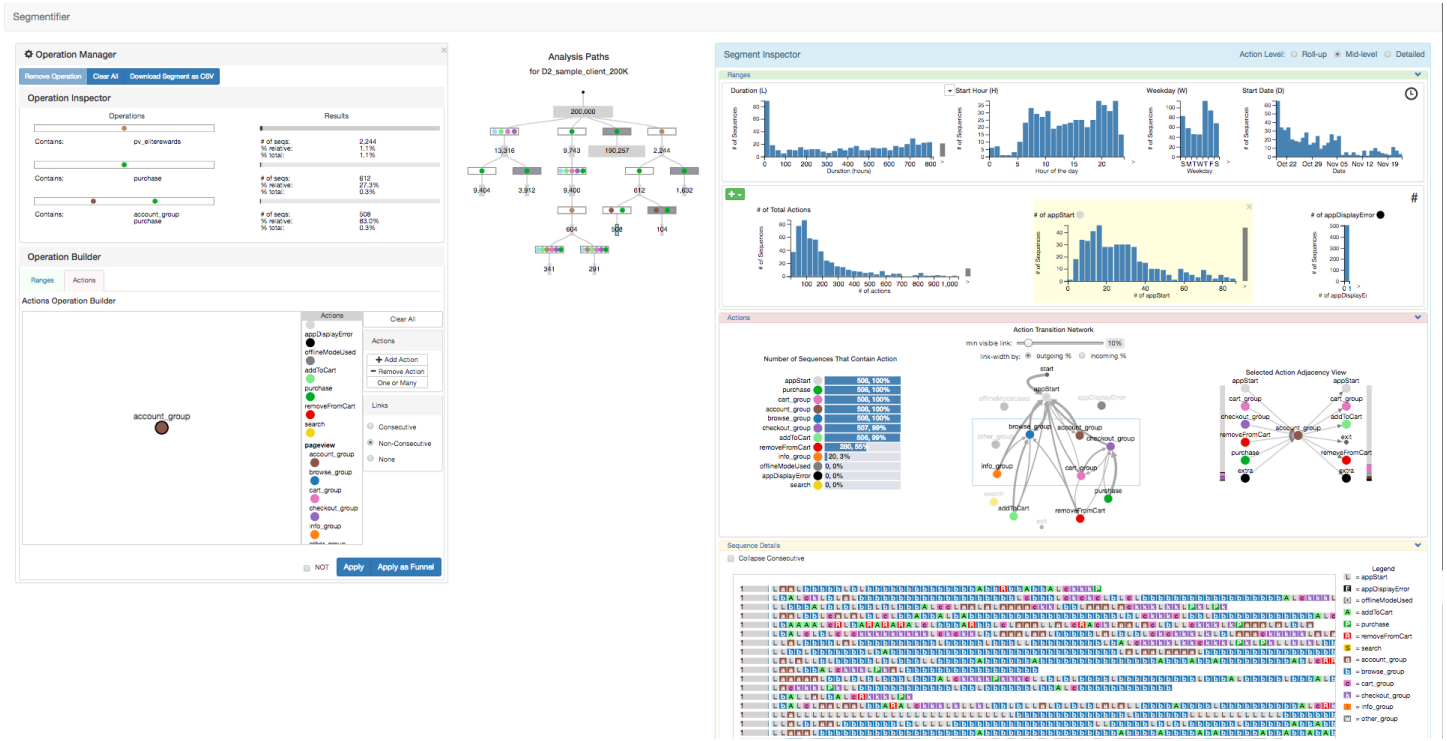

Figure A.20: The analyst switches to the *Mid-level* of the action hierarchy using the *Action Level* radio buttons at the top right corner to simplify the sequences. They notice in the *Operations Inspector* that 83% of users access any ACCOUNT page before purchasing.

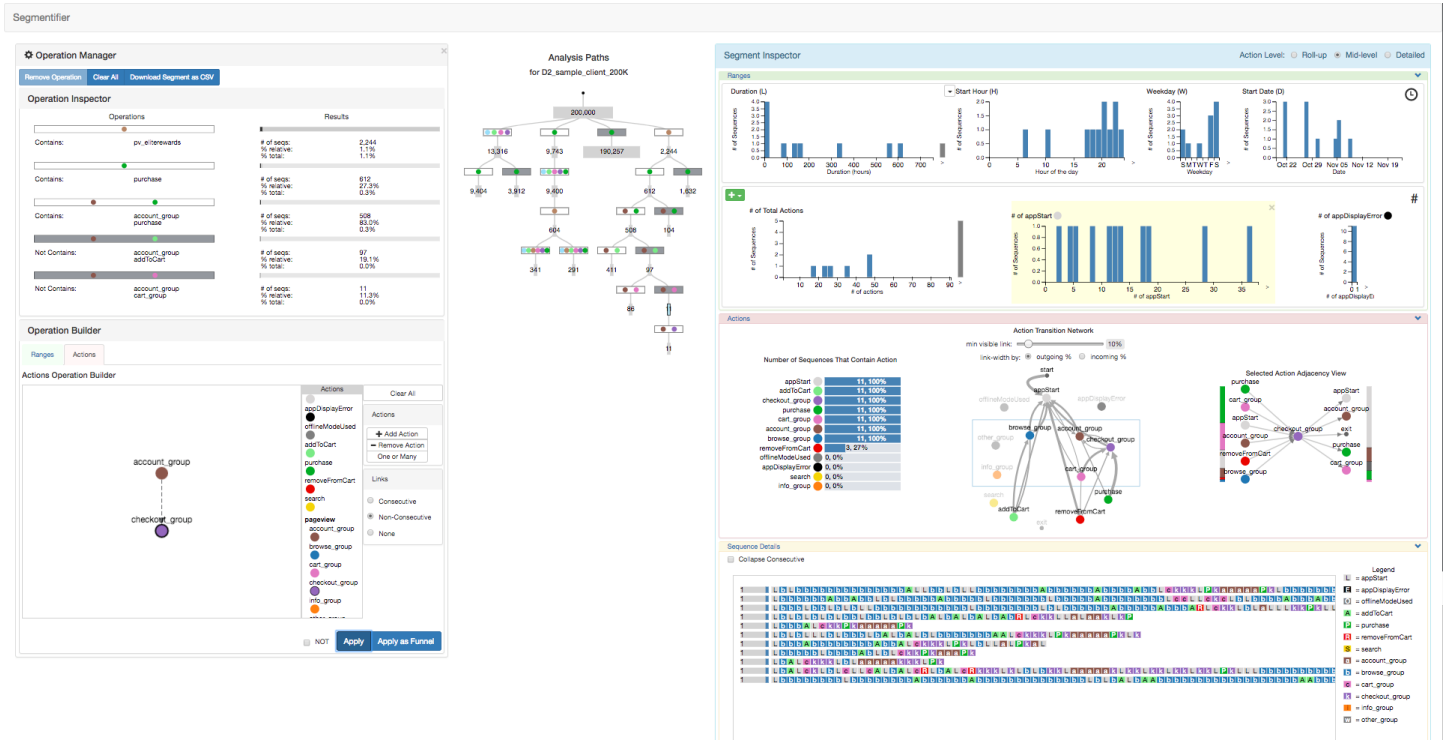

Figure A.21: They investigate further to determine at what point of the purchasing funnel users access their account. They establish that it occurs most frequently before ADDTOCART.