Unsupervised Clickstream Clustering for User Behavior Analysis

Gang Wang, Xinyi Zhang, Shiliang Tang, Haitao Zheng, Ben Y. Zhao
Department of Computer Science, UC Santa Barbara, Santa Barbara, CA, 93106
{gangw, xyzhang, shiliang_tang, htzheng, ravenben}@cs.ucsb.edu

ABSTRACT

Online services are increasingly dependent on user participation. Whether it's online social networks or crowdsourcing services, understanding user behavior is important yet challenging. In this paper, we build an unsupervised system to capture dominating user behaviors from clickstream data (traces of users' click events), and visualize the detected behaviors in an intuitive manner. Our system identifies "clusters" of similar users by partitioning a similarity graph (nodes are users; edges are weighted by clickstream similarity). The partitioning process leverages *iterative feature pruning* to capture the natural hierarchy within user clusters and produce intuitive features for visualizing and understanding captured user behaviors. For evaluation, we present case studies on two large-scale clickstream traces (142 million events) from real social networks. Our system effectively identifies previously unknown behaviors, *e.g.*, dormant users, hostile chatters. Also, our user study shows people can easily interpret identified behaviors using our visualization tool.

ACM Classification Keywords

H.5.0 Information Interfaces and Presentation (e.g., HCI): General; J.4 Computer Applications: Social and Behavioral Sciences; K.6.5 Management of Computing and Information Systems: Security and Protection

Author Keywords

Clickstream Analysis; User Behavioral Model; Visualization

INTRODUCTION

The next generation of Internet services is driven by user participation. Whether it's online social networks, online review services (Yelp, TripAdvisor), content sharing communities (Reddit) or crowdsourcing systems (Amazon Turk, TaskRabbit), their functionalities rely on active and well-behaved user participation.

Yet for these and many other systems, understanding user behavior is a complex and difficult challenge. In systems with millions of users, how can system builders understand the factors that drive each user's behavior? Understanding such factors can dramatically improve a user's experience, either through better performance, customized user interface features, or better targeted ads. Take for example the LinkedIn social network. LinkedIn is used by different types of users ranging from students not yet on the job market, happily employed professionals, professionals seeking new positions, and recruiters. Each user type uses the service differently, and yet rarely identifies their usage type explicitly in their profile data or elsewhere.

The intuitive solution is to survey users on how they use these systems through well-designed user studies [3, 34]. Unfortunately, this approach is limited by three factors. First, detailed user studies are limited in scale because of their significant cost to conduct and analyze. Studies sacrifice scale for depth on a small sample of the user population. Second, users may not be willing or able to self-identify into different user categories. Finally, user surveys rely on known questions or hypotheses. Unknown or new user behaviors cannot be anticipated in these studies.

These issues can be addressed by a data-driven approach to understanding user behavior. With improving data mining tools, today's online services collect all traces of user activity to produce *clickstreams*, sequences of timestamped events generated by user actions. For web-based services, these might include detailed HTTP requests. For mobile apps, clickstreams can include everything from button clicks, to finger swipes and text or voice input. Compared to user studies, clickstream analysis can scale to large user populations, identify behaviors without user assistance, and identify previously unknown behaviors.

Yet identifying common user behaviors in clickstreams is very challenging. Early works on clickstreams are limited, and focused on users' "navigation paths" within a website [10, 23, 27], or use Markov Chain models to predict popular webpages [15, 21]. To identify user behaviors today, we need a sophisticated clickstream analysis system that meets three requirements. *First*, it must scale and function well on large, noisy clickstream datasets. *Second*, the system should be able to capture previously unknown user behavior, *i.e.* capture behavior without categories or labels defined a priori. This is critical, because users often utilize popular services in unexpected ways, and adapting to these behaviors can determine the long-term viability of a service. *Finally*, the system should be interactive, and help others understand user behavior by presenting detected behaviors in an intuitive and understandable way. In contrast, current tools usually treat user models as a "black box" for classification tasks, while offering little explanations on how users behave and why [8, 29].

CHI 2016, May 7–12, 2016, San Jose, California, USA.

http://dx.doi.org/10.1145/2858036.2858107

In this paper, we present the design and evaluation of a practical and scalable clickstream tool for user behavior analysis. At a high level, our system uses similarity metrics between clickstreams to build similarity graphs that capture behavioral patterns between users. Edges capture similarity distance between clickstreams of users, and clusters represent user groups with similar behavior. We use a hierarchical clustering approach to detect the most popular behavior patterns, and use an *iterative feature pruning* technique to remove the influence of dominating features from each subsequent layer of clusters. The result is a hierarchy of behavioral clusters where higher-level clusters represent more general user behavior patterns, and lower-level clusters further identifying smaller groups that differ in key behavioral patterns. We can further use Chi-square statistics to identify statistical features that can be used to categorize and label behavior clusters.

Our system provides an easy way for service providers to analyze and understand groups and patterns in user behavior. First, the hierarchy of behavior clusters presents a compressed view of the most dominant user behavior patterns. In addition, because our approach does not rely on prior knowledge of categories or labels, it is able to capture any behavior patterns, both known and unknown. Finally, we integrate an *interactive visualization tool* to help service providers to examine the clustering results in real time.

To demonstrate the effectiveness of our system, we perform case studies using two large-scale, real-world clickstream datasets. One clickstream captures 135 million smartphone app events from 100K users on Whisper, a popular anonymous social network app. A second dataset comes from Renren (China's Facebook) and contains 7 million click events from 16K normal and malicious users. Our tool produces user behavioral models and reveals key insights about users on both networks. First, we identify patterns that capture different levels of "dormant users" on Whisper, and effectively predict dormant users based on neighboring behavior clusters. Second, we study user blocking behavior on Whisper and show that much of the blocking behavior is bidirectional, often following private message sessions, and is often correlated with sexually suggestive messages (sexting). On our Renren dataset, our system not only accurately identifies fake accounts with 95% accuracy, but also reveals subgroups that utilize different attack strategies. For example, we identify attacker subgroups that try to emulate normal users by intentionally slowing down their attack speed to avoid detection.

Finally, we evaluate our tool on two key benchmarks: First, we evaluate whether the algorithm-generated behavioral cluster are easy to understand with a controlled user study. We let participants summarize the corresponding user behaviors in a given cluster by examining cluster features. We find that most participants can interpret the semantic meaning of the user behavior and their summaries reach a high level of *consistency*. Second, we evaluate the clustering quality produced by our algorithm, in comparison to existing clustering methods (*e.g.*, K-means). Results show that our approach reaches a higher *accuracy* in detecting and grouping similar users.

Our paper makes three key contributions.

- We propose a novel unsupervised method to model online user behaviors. By building and partitioning a clickstream similarity graph, we capture the detailed user behavior models as hierarchical clusters in the graph. In addition, our tool automatically produces intuitive features to interpret the meaning of the behavioral clusters.
- We perform real-world case studies on two large-scale clickstream traces (142 million click events in total). We demonstrate that our tool can effectively help service providers to identify unexpected user behaviors (malicious accounts in Renren, hostile chatters in Whisper) and even predict users' future actions (dormant users in Whisper).
- Finally, we perform benchmark evaluations on our tool. The results show that the algorithm-generated cluster labels are easy to understand, and our tool produces highly accurate user behavioral models.

RELATED WORK

User Behavior Modeling in Online Services. Understanding user behavior is important to the design and operation of online services. Recent works analyze network traffic to understand online users' browsing habits [1, 18]. Researchers also built more specific user behavioral models to study users' search intent [19] and Wikipedia editing patterns [7], to predict crowdsourcing worker performance [20], and to detect malicious accounts in online social networks [29].

Clickstream Analysis. Earlier research used clickstream data for Web Usage Mining [23]. Researchers applied simple methods such as Markov Chains to capture users' navigation paths within a website [2, 15, 21]. However, these models focus on the simple aspects of user behavior (*e.g.*, user's favorite webpage), and are incapable of modeling more sophisticated user behavior. Other approaches use clustering techniques to identify user groups that share similar clickstream activities [8, 25, 27, 29]. The resulting clusters can be used to infer user interests [25] or predict future user behaviors [8]. However, existing clustering based models are largely supervised (or semi-supervised), requiring large samples of ground-truth data to train or fine-tune the model parameters [21, 27, 29]. Also, many behavioral models are built as "black boxes" for classification tasks, offering little explanations on how users behave and why [8,29]. Our work seeks to build unsupervised clickstream behavioral models and produce intuitive explanations on the models.

Clickstream Visualization. Researchers have developed interactive interfaces to visualize and inspect clickstream data. Existing tools generally focus on visualizing raw user clicks [16], click event sequences [35] or click transitions [32]. Instead, we build a tool to visualize the clickstream behavioral clusters produced by our system, providing hints for understanding key user behavior patterns.

CLICKSTREAM DATASETS

In this work, we seek to build a clickstream tool for user behavioral modeling in online services. To provide context, we first describe the clickstream datasets used in our study. We obtained server-side clickstream data from two large-scale

Dataset	Time	# of Users	# of Events
Whisper	Oct.13–Nov.26 2014	99,990	135,208,159
Renren-Normal	Mar.31–Apr.30 2011	5,998	5,856,941
Renren-Sybil	Feb.28–Apr.30 2011	9,994	1,008,031

Table 1. Clickstream datasets from Whisper and Renren.

online social networks: Whisper and Renren. In the following, we introduce the two online social networks and the clickstream datasets. Note that we have taken careful precautions to avoid any personally identifiable information in our datasets, and our study has been approved by our local IRB under protocol #COMS-ZH-YA-010-6N.

Whisper

Whisper is a popular smartphone app for anonymous social messaging. It allows users to share confessions and secrets under anonymous nicknames without worrying about privacy [6, 30]. As of April 2015, Whisper has reached 10 million users. Unlike traditional social networks, Whisper does not maintain user profiles or social connections. Its key function is messaging: the app overlays a user's short text message on top of a background picture selected by keywords. The resulting *whisper* message is posted to the public stream where other users can read, reply or heart (like) it. In addition, the app provides a chat feature to facilitate direct communication. Any user can start a private chat with the whisper author. Finally, users browse whispers from several public lists.

We collect detailed clickstream data from Whisper in collaboration with Whisper's Data Science team. The dataset contains 136 million click events from 99,990 users over 45 days in 2014 (Table 1). Users were randomly selected from the Whisper user population as a representative sample. Each click event is characterized by userID, timestamp, event type and event parameter. The userID in our dataset (including Renren data) is globally unique and has been fully anonymized to protect user privacy. We obtained userIDs from each company through internal collaborators. The Whisper dataset contains 33 types of events that can be grouped into 6 categories. These categories are:

- **Browsing:** Browsing whispers, visiting the public whisper feeds (popular/nearby/latest list).
- **Account:** Creating a user account and login the app.
- **Posting:** Posting original whispers and replies, hearting/unhearting a whisper, sharing whispers, and tagging a whisper to a topic.
- **Chatting:** Initiating a chat, blocking other users in a chat, and being blocked in a chat.
- **Notification:** Receiving notifications about hearts/replies on their whispers, and whisper recommendations.
- **Spam:** Whisper being examined or deleted by system admins, flagging other people' whispers. Events in this category are all below 1% (omitted from Table 2).

Among the 33 event types, 25 are user-initiated events corresponding to the user performing an action on the app (*e.g.*, "posting a whisper"). The rest 8 events are system events which don't require user action (*e.g.*, "receiving notifications"). Table 2 shows the most popular events and the absolute number (in thousands) and the percent of clicks. Overall,

Category	Event Type	Events # (K)	Events %	Initiated By User?
Browsing	View whisper	52437	38	Yes
	View popular feed	16008	12	Yes
	View nearby feed	5354	4	Yes
	View latest feed	2346	2	Yes
	View other feed	196	1	Yes
Account	Login	16994	12	Yes
Posting	Heart whisper	2156	2	Yes
	Upload image	1325	1	Yes
	Create whisper	1308	1	Yes
Chatting	Being blocked in chat	3271	3	No
	Block user in chat	3271	3	Yes
	Start a chat	2238	2	Yes
Notification	Receive notification	9680	7	No
	Whisper recommendation	2530	2	No

Table 2. Event types in the Whisper dataset. # of click events are presented in thousands. Events that are <1% are omitted for brevity.

Category	Event Type	Sybil Events # (K)	Sybil Events %	Normal Events # (K)	Normal Events %
Friending	Send request	417	41	16	0
	Accept invitation	20	2	13	0
	Invite from guide	16	2	0	0
Photo	Visit photo	242	24	4,432	76
	Visit album	25	2	330	6
Profile	Visit profiles	160	16	214	4
Sharing	Share content	27	3	258	4
Message	Send IM	20	2	99	2
Blog	Visit/reply blog	12	1	103	2
Notification	Check notification	8	1	136	2

Table 3. Event types in the Renren dataset. # of click events are presented in thousands for Normal and Sybil users. Events with <1% of clicks are omitted for brevity. All of the events are user-initiated events.

the most prevalent events are related to content consumption such as viewing whispers. Interestingly, under the chatting category, the most prevalent events are "blocking users" and "being-blocked" by others. Intuitively, anonymous environment is more likely to foster abusive behaviors (*e.g.*, bullying) [26]. Later, we investigate this behavior in greater details using behavioral models.

Our dataset also contains the content of the public whispers (about 1 million) posted by these users. This content data is not used to construct clickstreams, but used to understand specific user behavior and user intent later in our analysis.

Renren

Renren is one of the largest online social networks in China with 223 million users as of December 2014. Renren offers similar functionalities as Facebook, allowing users to maintain a personal profile and build social connections. Users can post status updates, write blog entries, share photos, and interact with other users' content (*e.g.*, liking and commenting). The key difference between Renren and Facebook is Renren's "footprint" feature, which allows users to see who have recently visited their profiles.

Our Renren dataset contains 5998 normal users and their clickstream traces over two months in 2011 (Table 1). In addition, it also includes 9994 Sybil accounts (*i.e.*, malicious accounts suspended by Renren), which can serve as the "ground-truth" data to evaluate our system. There are 55 types of events in our Renren dataset that can be grouped into 8 primary categories. These categories are:

- **Friending:** Sending friend requests, accepting or denying those requests, and un-friending.
- **Photo:** Uploading photos, organizing albums, tagging friends, browsing photos, and writing comments.
- **Profile:** Browsing user profiles. Profiles on Renren can be browsed by anyone, but the displayed information is restricted by the owner's privacy settings.
- **Sharing:** Users posting URLs linking to videos, blogs or photos in/outside Renren.
- **Message:** Status updates and instant-messages.
- **Blog:** Reading/writing blogs, and commenting.
- **Notification:** Users actively clicking on the notifications.
- **Like:** Users liking (or unliking) content.

Unlike Whisper, Renren traces do not contain any system events such as "receiving notifications." All events are initiated by users. Table 3 displays the most popular events. Note that the percentages for click events are calculated for Sybils and normal users separately, *i.e.*, each "%" column sums to 100%. We find Sybils and normal users behave differently. Normal users spend most of their clicks on viewing photos (76%), albums (6%), and sharing (4%). In contrast, Sybils are skewed to making friend requests (41%), viewing photos (24%), and browsing profiles (16%). Later we will analyze specific Sybil attack strategies using behavioral models.

UNSUPERVISED USER BEHAVIOR MODELING

In this section, we describe our unsupervised method to build user behavior models from clickstream data. At the high level, our system assumes that human behavior naturally forms clusters. Despite users' differences in personalities and habits, their behavioral patterns within a given service cannot be entirely disparate. Our goal is to identify such natural clusters as behavioral models. In addition, user behavior is likely multi-dimensional. We expect user clusters to fall into a tree hierarchy instead of a one-dimensional structure (Figure 1). In this hierarchy, most prominent features are used to place users into high-level categories while less significant features characterize detailed sub-structures.

To these ends, we design an algorithm to captures hierarchical clickstream clusters with *iterative feature pruning*. At a high level, we map users into a similarity graph where nodes are users and edges are weighed on the similarity between user clickstreams. We partition the similarity graph to identify clusters of users with similar clickstream activities. To capture the hierarchical structure, we recursively partition newly generated clusters, while *pruning* the feature set used to measure clickstream similarity. Intuitively, by identifying and pruning dominating features in higher-level clusters, we allow the secondary features to manifest and discover more fine-grained subclusters. Also, the pruned features are indicative of why this cluster is formed, which can help service providers to understand the behavioral model.

In the following, we first introduce the notion of clickstreams and similarity graph. Then we describe the feature-pruning algorithm to identify clusters in the similarity graph. Finally, we build a visualization tool to help service providers examine and understand behavioral clusters.

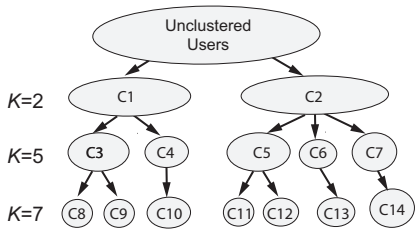


Figure 1. Hierarchy of the behavioral clusters.

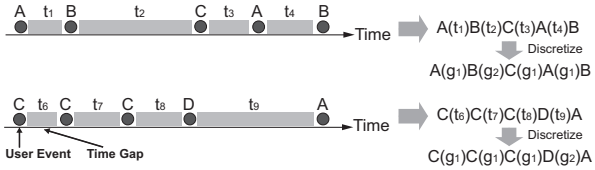


Figure 2. Discretizing two sample clickstreams into event sequences.

Clickstream and Similarity Graph

Formatting User Clickstream. For each user, we gather all her click events to form a single clickstream. It is a sequence of discrete events sorted by the order of arrival, capturing both different *event types* in the stream and the magnitude of *time gaps* between them. For example, take the clickstreams in Figure 2: $A(t_1)B(t_2)C(t_3)A(t_4)B$ and $C(t_6)C(t_7)C(t_8)D(t_9)A$ where A, B, C, D are event types, and t_i is the time interval between the i_{th} and $(i+1)_{th}$ event. To make time gaps discrete, we replace precise time gap values with discrete identifiers that represent a range of time gap, *i.e.*, a "bucket". For our system, we map the time gap into the following five discrete time buckets: $< 1s, [1s, 1min], (1min, 1h], (1h, 1day], > 1day$, represented by g_1, g_2, g_3, g_4 and g_5, respectively.[1] The above two clickstreams are further discretized to $A(g_1)B(g_2)C(g_1)A(g_1)B$ and $C(g_1)C(g_1)C(g_1)D(g_2)A$.

Clickstream Similarity Graph. Our clustering algorithm is based on *similarity graph*, where each node is a user and edges are weighted on similarity between two users' clickstreams [29]. We identify behavioral clusters by partitioning the similarity graph. To do so, we need a metric to measure the similarity (or distance) between any two clickstreams.

Our method is to extract subsequences from the clickstreams as features to compare similarity. More specifically, we formalize a clickstream as a sequence $S = (s_1 s_2 ... s_i ... s_n)$, where s_i is the i^{th} element in the sequence (either a click event or time gap event) and n is the total number of events in the sequence. We define T_k as the set of all possible k-grams (k consecutive elements) in sequence S: $T_k(S) = \{k\text{-}gram | k\text{-}gram = (s_j s_{j+1} ... s_{j+k-1}), j \in [1, n+1-k]\}$. To compute the distance between two sequences, we consider both the common k-grams in the two sequences and their *count*. For S_1, S_2 and a chosen k, we first compute the set of all possible k-grams from both as $T = T_k(S_1) \cup T_k(S_2)$. Next, we count the normalized frequency of each k-grams

[1] We use uneven bucket sizes to handle the "long tail" distribution of time gaps between clicks. In our dataset, the majority of time gaps are short (e.g. minutes) while long gaps (e.g. days) are rare.

within each sequence l ($l = 1, 2$) as array $[c_{l1}, c_{l2}, ..., c_{ln}]$ where $n = |T|$. Finally, their distance is computed as the normalized *Polar Distance* between the two arrays: $D(S_1, S_2) = \frac{1}{\pi} \cos^{-1} \frac{\sum_{j=1}^{n} c_{1j} \times c_{2j}}{\sqrt{\sum_{j=1}^{n}(c_{1j})^2} \times \sqrt{\sum_{j=1}^{n}(c_{2j})^2}}$. The value ranges from 0 to 1, and small distance indicates high similarity between two clickstreams. We choose Polar distance over other alternatives (*e.g.*, Euclidean distance) because Polar distance is more suitable to handle highly sparse vectors: it compares the "directionality" of the vectors rather than "magnitude" [11].

We set $k = 5$ for our system, after testing different Ks from 1 to 10. We find larger K values do not give us benefits (*e.g.*, Sybil detection accuracy reaches diminishing returns after K$= 5$). Intuitively, large K's will capture long sequences that are unlikely to repeat as a pattern. Also, the number of features (and associated computational costs) increases exponentially with K.

Feature Pruning based Clickstream Clustering

A similarity graph dominated by very few features gives little insight on subtle differences between users. The generated clusters may only describe the broadest user categories, while interesting and detailed behavioral patterns remain hidden. We recognize that similarity graph has the capability to capture user behavior at different levels of granularity. We implement *iterative feature pruning* as a means of identifying fine-grained behavioral clusters within existing clusters, and recursively partitioning the similarity graph. In the following, we first introduce the key steps of our clustering algorithm and feature pruning. Then we describe using pruned features to interpret the meaning of the clusters, and the technical details to determine the number of clusters.

Iterative Feature Pruning & Clustering. We explain how our algorithm works using the example in Figure 1. We start with a similarity graph of all users, where clickstream similarity is measured based on the full feature set (union of all k-grams). By partitioning the similarity graph, we get the top-level clusters C_1 and C_2. The partitioning algorithm we use is Divisive Hierarchical Clustering [13], which can work on arbitrary metric space and find clusters of arbitrary shapes.

To identify more fine-grained subclusters within C_1 or C_2, we perform feature pruning: We identify the primary features that are responsible for forming the parent cluster, remove them from the feature set, and use the remaining secondary features to further partition the parent. For example, suppose C_1 is the current parent cluster. We first perform feature selection to determine the key features (*i.e.*, k-grams) that classify users into C_1. Then to partition C_1, we remove those top k-grams from the feature set, and use the remaining k-grams to compute a new similarity graph for C_1. In this way, secondary features can step out to partition C_1 into C_3 and C_4 (by running Divisive Hierarchical Clustering on the new similarity graph). For each of the newly generated clusters (*e.g.*, C_3 and C_4), we recursively run the same process to produce more fine-grained subclusters. Our algorithm stops when all the new partitions cannot be further split, *i.e.* clustering quality reaches a minimal threshold. The result is a tree hierarchy of behavioral clusters.

The key step of feature pruning is finding the primary features responsible for forming the parent cluster. We select features based on Chi-square statistics (χ^2) [33], a classic metric to measure feature's discriminative power in separating data instances of different classes. For a given cluster, *e.g.*, C_1, we measure the χ^2 score for each feature based the distribution of users in C_1 and those outside C_1. We sort and select the top features with the highest χ^2 scores. Our empirical data shows χ^2 distribution usually exhibits "long-tail" property — only a small number of dominating features have high χ^2 scores. We automatically select top features by identifying the sweet point (or turning point) in the χ^2 distribution [22].

Understanding the Behavioral Clusters. We can infer the meaning of the clusters based on the selected features during feature pruning phase. A feature is selected because users in this cluster are distinct from users outside the cluster on this particular feature dimension. Thus it can serve as explanations for why a cluster has formed and which user behaviors the cluster encompasses. Later we construct a visualization tool to help service providers interpret behavioral clusters.

Determining the Number of Subclusters. For each parent cluster (and its similarity graph), our system identifies the natural number of subclusters within. To do so, we monitor the changes of the overall *clustering quality* while continuously partitioning the graph to more subclusters. We stop when generating more subclusters will no longer improve the clustering quality. The metric to assess clustering quality is the widely-used *modularity*, which measures the density of edges inside clusters to edges outside clusters [4]. The modularity value ranges from -1 to 1, with a higher value indicating a better clustering quality.

Cluster Visualization

We build a visualization tool for service providers to examine and understand user behavioral clusters generated by our algorithm. The tool allows service providers to answer key questions about their users, *e.g.*, what are the major behavioral categories? Which behavior is more prevalent? What's the relationship between different types of behavior?

Visualization Interface. Figure 3 shows a screenshot of our tool displaying the behavioral clusters of Whisper (best viewed in color). We build this tool using *D3.js*, a JavaScript library for data visualization. By default, we display the cluster hierarchy using Packed Circle [31], where child clusters are nested within their parent cluster. This gives a clear view of the hierarchical relationships of different clusters. Circle sizes reflect the number of users in the cluster, which allows service providers to quickly identify the most prevalent user behaviors. Finally, the visualization tool is zoomable and easy to navigate among clusters. We also implemented other interfaces such as Treemaps [12], Sunburst [24] and Icicle [14]. Service providers can choose any of these based on personal preference (Figure 5). We use Packed Circle as default because it leaves more space between clusters, making it easier to visually separate different clusters.

To understand the user behavior in a specific cluster, we can click the cluster to pop-up an information window. Take

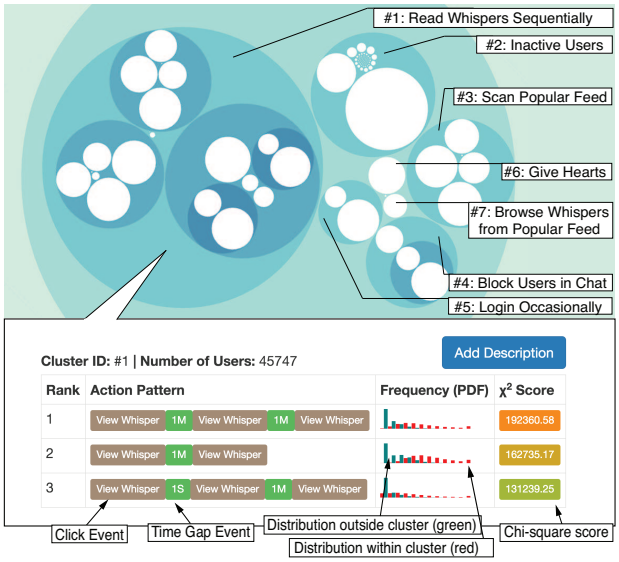


Figure 3. Whisper behavioral clusters. Cluster labels are manually input based on results of each cluster. The pop-up window shows users in Cluster #1 tend to sequentially read whispers.

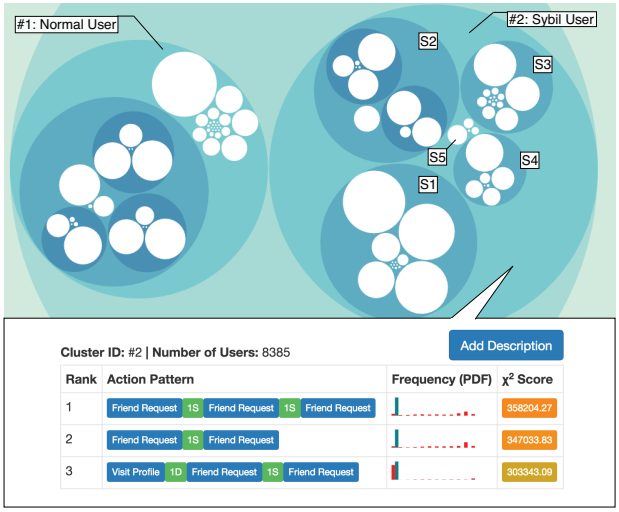


Figure 4. Renren behavioral clusters. The pop-up window shows users in Cluster #2 focus on sending friend requests and browsing user profiles.

the one in Figure 3 for example: we show the basic cluster information on top, including clusterID and the number of users. Below is a list of "Action Patterns" (k-grams) selected by our Feature Pruning algorithm to describe how users behave. Each row contains one pattern, ranked by χ^2 score (brighter color indicates higher score). The "Frequency (PDF)" column shows how frequently each action pattern appears among users of this cluster. The red bar indicates the pattern frequency (probability density function) inside the cluster, and the green bar denotes frequency outside of this cluster. Intuitively, the more divergent the two distributions are, the more distinguishing power the pattern has. In this example, the first pattern shows users viewing whispers sequentially with a time interval of one minute or less. The red bar is much more skewed to the right, indicating users in this cluster perform this action more often than users outside. Finally, service providers can "add descriptions" to the cluster using the button in blue.

Figure 6 shows the control panel to configure the visualization tool. Users can switch the visualization interface and set maximum level of clusters to display. Users can also change the cluster coloring scheme. By default, the cluster color indicates the level (or depth) of the cluster. Alternatively, users can enable a color overlay to denote the "compactness" of the cluster (*i.e.*, the ratio of average inter-cluster distance over the average intra-cluster distance) or the cluster modularity.

Visualizing Whisper and Renren Clusters. We run our system on Whisper and Renren datasets and display the behavioral clusters in Figure 3–4. We apply the same configuration on both runs: the partitioning of a cluster stops if the modularity reaches a threshold 0.01 (insignificant cluster structure). We intentionally set a loose threshold to let the algorithm dig out very detailed sub-clusters. In practice, service providers can tune this parameter depending on how detailed behavioral clusters they need. Details regarding algorithm implementation and complexity are in the Appendix. For our Whisper dataset, our system produces a tree hierarchy of 107 clusters (root included) with 95 leaf clusters. The maximum tree depth is 4. For Renren, the hierarchy contains 108 clusters (95 leaf clusters) with a maximum depth of 4.

Note that our visualization tool only displays the selected features for each cluster. As shown in Figure 7, 80% of the clusters have less than 5 selected features, and 90% of the clusters have less than 10. This indicates that the prevalent user behavior can be characterized by a small number of key feature dimensions. Also, this makes it possible for people to understand the cluster without looking through the full feature set (*e.g.*, Whisper data has 80903 unique kgrams as features).

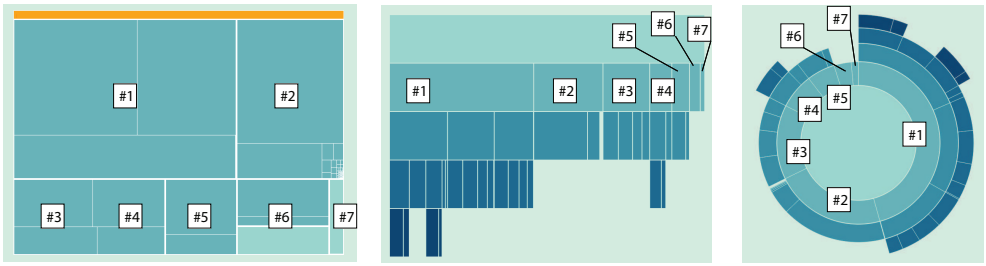
EVALUATION: CLUSTER LABELS

In the following, we analyze the behavioral clusters in Whisper and Renren, and demonstrate their effectiveness in identifying unexpected behavior and predicting future activities. Our evaluation contains three steps. First, To evaluate the ease of understanding and labeling behavioral clusters, we run a user study. We ask the participant to read cluster information and describe the corresponding user behavior. Then we examine whether different people give consistent descriptions. Second, we perform in-depth case studies on the unusual behavioral clusters, and provide new insights to both networks. Third, we evaluate cluster quality, *i.e.*, how well behavioral clusters capture similar users.

User Study to Interpret Clusters

User behavioral models need to be intuitive and understandable to the service providers. Thus we conduct a user study to answer two key questions: Are these behavioral clusters understandable to humans? How consistently do different people interpret the corresponding user behaviors?

In this user study, we ask participants to browse behavioral clusters using our visualization tool (Packed Circle interface).



(a) Treemaps (b) Icicle (c) Sunburst

Figure 5. Whisper hierarchical clusters displayed with different visualization methods. We mark the cluster number of the top-level clusters in the text box.

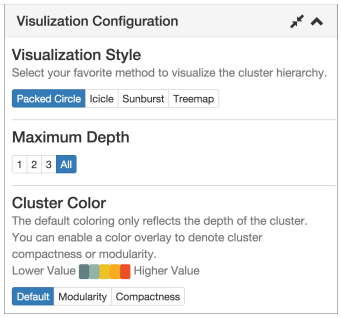


Figure 6. Screenshot of the configuration panel of the visualization tool.

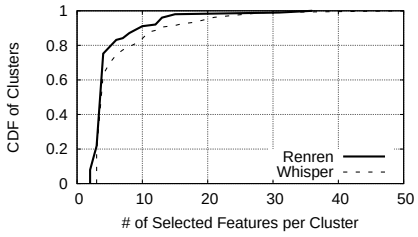


Figure 7. # of Selected features per cluster.

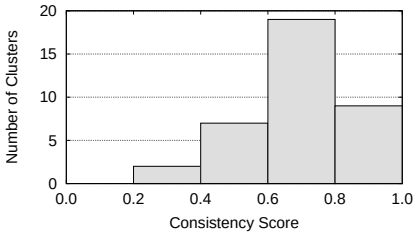


Figure 8. Distribution of consistency score.

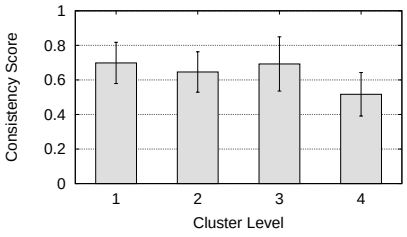


Figure 9. Consistency score vs. cluster level.

For each cluster, the participant is asked to describe the user behavior using her own words (in one sentence) based on the information displayed. If a cluster is not understandable to the participant, she can mark it as "N/A". Since our tool is designed for service providers, we expect they will have basic technical backgrounds. Our participants include 15 graduate students in Computer Science who have basic knowledge in online social networks. To best utilize participants' time, we only use the Whisper clusters (Figure 3), and the participants only look at top clusters that cover 90% of users at each level of the hierarchy (37 clusters in total). Before the test, we ask the participants to use the Whisper app for at least 10 minutes to get familiar with it. Each participant also goes through a quick instruction session to learn how to use the visualization tool and how to read the information in the pop-up window.

User Study Results

We gathered a total of 555 descriptions from the participants on the 37 clusters (15 descriptions per cluster). We find that the behavioral clusters are generally understandable to the participants. Out of the 555 descriptions, 530 (95.5%) are valid descriptions about user behaviors (others are "N/A" marks). In addition, most participants can finish the task within a reasonable amount of time. The average completion time is 28.7 minutes (46 seconds per cluster).

To understand the "consistency" of the descriptions, we let 3 external experts independently read and assess the collected descriptions. These experts are graduate students recruited outside of our research group (to avoid bias) and none of them participated in labeling clusters in the first round. For each cluster, an expert reads all 15 descriptions and assigns a consistency score (0 to 1), which is the ratio of the maximum number of consistent descriptions over all descriptions. For example, if 10 out of the 15 descriptions are consistent, the score is 10/15=0.667. The final consistency score is averaged over three experts. Figure 8 shows the consistency score distribution. The most common scores range from 0.6 to 0.8. The score distribution skews heavily to the right. This indicates that most clusters can be interpreted consistently.

Upon examining clusters with low consistency scores, we have two key observations. First, lower-level clusters are more difficult to interpret. As shown in Figure 9, average consistency scores decrease as we move further along the tree hierarchy. Intuitively, lower-level clusters represent more specific or even outlier behavior that is difficult to describe. Second, we find clusters with more selected features are harder to interpret. We perform correlation analysis between the consistency score and the number of selected features per cluster, and find they correlate negatively (Pearson coefficient r =-0.1, p =0.5). Noticeably, the consistency score also correlates negatively with the unique event types in selected features (Pearson coefficient r =-0.4, p =0.02).

Finally, we add short labels to the top-level clusters in Whisper and Renren based on the descriptions from user study and our own interpretations. The labels are shown in Figure 3 and Figure 4 respectively.

EVALUATION: CASE STUDIES

Next, we present in-depth analysis on a few behavioral clusters as case studies. We have two goals. First, by analyzing the user behavior in these clusters, we validate the correctness of our cluster labels. Second, we explore the interesting (or unexpected) user behavior, and demonstrate the prediction power of the user behavioral models. Due to space limitation, we focus on two clusters from Whisper (Cluster#2 and Cluster#4), and one from Renren (Sybil Cluster).

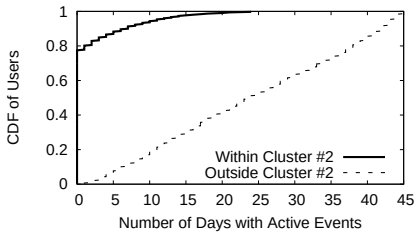


Figure 10. Number of days when users have active events in their clickstreams.

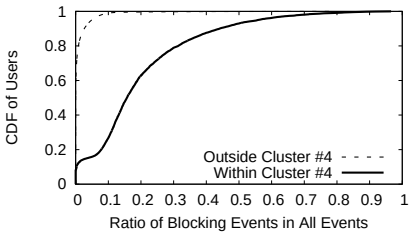


Figure 11. Ratio of blocking event over all click events in a user's clickstream.

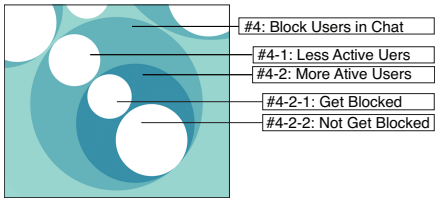


Figure 12. The sub-clusters within Cluster#4.

Case Study 1: Inactive Whisper Users

We start with Cluster#2, which is labeled as inactive users. The selected action patterns of this cluster consist almost entirely of "receiving notification" events, indicating these users have not been actively engaged with the app. This is also confirmed in Figure 10: users in Cluster#2 have far fewer active days (when users actively generate clicks) than the rest of users. A remarkable 80% of users in Cluster#2 did not generate any active events through the 45 days, representing completely dormant users. In fact, our algorithm successfully groups dormant users into a separate subcluster (Figure 3, the biggest subcluster in Cluster#2).

Contrary to expectation, inactive users are not outliers. Cluster#2 is the second largest cluster with 21,962 users (20% of all users). From the perspective of service providers, it is important to identify the early signals of user disengagement, and implement mechanisms to re-gain user activities.

Predicting Dormant Users. We demonstrate the effectiveness of our behavioral models in predicting future user dormancy. The high-level idea is simple: Whisper can build behavioral models using users' most recent clickstreams, and update the models at regular intervals (*e.g.*, every month). Our hypothesis is that users placed in the "inactive" cluster are more likely to turn completely dormant. Thus we can use the inactive cluster to predict future dormant users.

We validate this hypothesis by investigating whether users in the "inactive" cluster will migrate to the "dormant" cluster over time. To do so, we split our clickstream data by date into three snapshots: Oct.13–27, Oct.28–Nov.12 and Nov.13–26. Then we generate behavioral clusters for each snapshot. The inactive cluster can be easily pinpointed within each snapshot based on selected activity patterns (*i.e.*, notification events). Also, we consistently find the following sub-structures within the inactive cluster: a big "dormant" cluster in which users have zero active events, alongside several "semi-dormant" clusters in which users are occasionally active.

In Table 4, we compare clusters from two adjacent snapshots to determine the likelihood of users migrating into the dormant cluster. The results confirm our hypothesis: Users in semi-dormant clusters are more likely to migrate to the dormant cluster than others. For example, 17% of semi-dormant users in snapshot-2 end up in the dormant cluster in snapshot-3, while only 1.1% of other users do so. Users already within the dormant cluster are highly likely to remain there through future snapshots (94%-99%). This result shows that our behavioral models can successfully track and predict the dormancy of Whisper users. It allows service providers to make timely interventions before losing user participation.

Cluster	# (%) of Users Join the Dormant Cluster	
	Snap 1 → Snap 2	Snap 2 → Snap 3
Dormant Cluster	15873/16872 (94%)	16161/16314 (99%)
Semi-dormant Clusters	363/9383 (4%)	2026/11773 (17%)
Other Clusters	63/73735 (0.09%)	804/71903 (1.1%)

Table 4. Users becoming dormant over time. We split the clickstream data into three snapshots, and report the number of users who migrate to the dormant cluster over two adjacent snapshots.

Actions per day	Statistics: Mean (STD)		T-statistics (p value)
	Inside C#4	Outside C#4	In vs. Out
Whisper Posted	1.25 (1.77)	0.65 (1.46)	27.43 (p<0.001)*
Replies Received	0.70 (4.09)	0.26 (1.41)	8.89 (p<0.001)*
Heart Received	2.39 (48.68)	0.69 (5.34)	2.93 (p=0.0034)*
Chats Initiated	2.20 (10.93)	1.18 (3.98)	7.79 (p<0.001)*

Table 5. Activity statistics for users inside and outside Cluster#4. *The difference is statistically significant based on Welch two-sample t-tests.

Case Study 2: Hostile Behaviors of Whisper Chatters

Next, we analyze Cluster#4, which contains 7026 users who tend to block other people during private chat. As shown in Figure 11, users in this cluster perform blocking actions much more frequently. 80% of users spend more than 10% of their total clicks on blocking events. In contrast, only 1% of users outside Cluster#4 achieve this ratio.

Next, we explore the possible causes to the blocking events. A private chat is initiated by the user who wants to talk to whisper authors. Our hypothesis is that users in Cluster#4 are more likely to post whispers which attract unwanted chatters to harass them. To validate this, we list behavioral statistics for users inside and outside Cluster#4 in Table 5. Users in Cluster#4 are more active in posting public whispers, which attract more hearts and replies from others (statistically significant based on Welch t-tests). These users are likely to experience harassment as a side effect.

Users may attract unwanted chat messages due to the topics they write about. We analyze users' whisper content in Cluster#4 and find they often consist of sexually explicit messages (sexting). Table 6 lists top keywords from users in and outside Cluster#4. Keywords are ranked based on how strongly they are associated with the cluster. For each keyword, we compute a simple correlation ratio for ranking, as the number of whispers in Cluster#4 containing this keyword divided by the total number of whispers with this word. We exclude

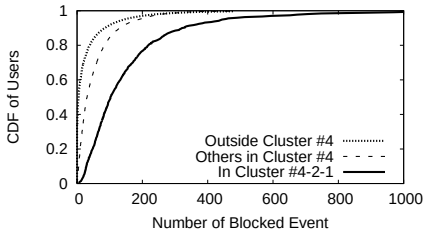
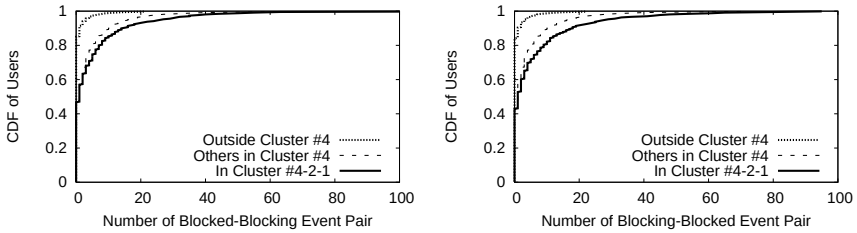


Figure 13. Number of being-blocked events per user.



(a) Blocked→Blocking Event (b) Blocking→Blocked Event

Figure 14. Number paired blocking and blocked events per user. We match blocking and blocked events under the same whisper with time interval $<$ 1 hour.

Users	Top 30 Keywords
Inside C#4	20f, 19f, 18f, 17f, 29, f, roleplay, daddy, wet, role, lesbians, 17, lesbian, kinky, trade, bored, kik, weakness, nude, threesome, bestfriend, msg, shower, boys, chubby, nipples, horny, female, dirty, message
Outside C#4	religion, que, bullshit, 18m, personally, bible, eventually, faith, sign, plenty, hilarious, congratulations, gender, brain, idiot, dumbass, ignorant, quite, depends, animals, google, society, loss, count, health, sexuality, em, business, sound, foot

Table 6. Top whisper keywords for users in Cluster#4 and users outside Cluster#4.

ID	Cluster Label	# of Users	FrdReq per Day	ProfileReq per Day	In/out FrdReq
S_1	Friending in bulks	4064	**25.13**	0.30	0.002
S_2	Friending quickly	1891	19.81	2.08	0.004
S_3	Crawl profiles	1348	11.41	**6.44**	0.050
S_4	Friending slowly	899	**8.76**	1.93	0.00004
S_5	Receive FrdReq	129	25.65	3.43	**0.286**
#1	Normal users	6141	1.65	2.80	1.06

Table 7. Characteristics of users the 5 biggest Sybil clusters (S_1–S_5) and the normal user cluster. We add the cluster label based on the selected action patterns per cluster. "FrdReq" stands for "friend requests."

common stopwords [5] and low frequency words to avoid statistical outliers. A mere glance at Table 6 reveals that Cluster#4 users are focused on exchanging sexual content. Terms like "20f", "f","17" and "lesbian" indicate age, gender (f = female) and sexual orientation. Other frequently used words are associated with the exchange of nude photos ("trade", "shower", "nipples") or more general erotic terms.

Users Who Get Blocked. Within Cluster#4, we find a subcluster of 1412 users who often get blocked by others (Cluster#4-2-1 in Figure 12). As shown in Figure 13, these users have more "being-blocked" events in their clickstreams. In the meantime, as members of Cluster#4, these users are also highly likely to block other users.

Then the question is how often blocks are "bidirectional", *i.e.*, user X blocks Y and then Y immediately blocks X. Unfortunately, our dataset cannot directly measure bidirectional blocks. For a blocking event, the known information includes the whisperID where two users chat, the userID issuing the block, but not the userID being blocked. Thus we take an approximation approach to match potential "bidirectional" blocks (as upper bound). For each user, we group her blocking and being-blocked events under the same whisperID as a *pair* if their time interval is within a short time window (*e.g.*, one hour). This approximates immediate blocking back after getting a block. Figure 14 shows the matching result using time window as 1-hour. Users in Cluster#4, particularly in Cluster#4-2-1 have a higher number of paired blocking events. It is likely these users are easily offended or often offend other users during private chat, suggesting a strong hostile behavior. We also test 10 minutes and 1-day time window and have similar conclusion.

Case Study 3: Renren Sybil Accounts

Finally, we analyze the Sybil cluster in Renren (Cluster#2 in Figure 4). Our system groups Sybil accounts into one single cluster with high accuracy. 95% of true Sybils are clustered into the cluster and only 0.74% of normal users are misclassified. The selected features indicate Sybils are more likely to engage in sending friend requests. This makes sense because a Sybil must first befriend a user before accessing private information or spamming.

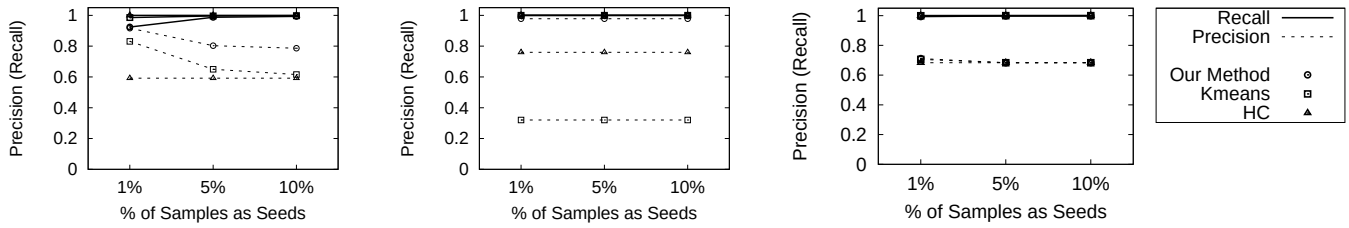
In addition, our system uncovers more fine-grained subclusters within the Sybil cluster, representing different attack strategies. Here we focus on the largest 5 (out of 8 subclusters), which encompass 99.36% of Sybil accounts. Table 7 shows their behavioral statistics. First, S_3 appears to describe "crawlers" who specialize in collecting user information and photos for sale on the black market [17]. Second, S_1, S_2 and S_4 all focus on "sending friend requests." Sybils in S_1 send requests in bulks via Renren's friend recommendation system, resulting in a high volume of friend requests per day (25.13). On the other hand, Sybils in S_4 tend to build social connections slowly (8.76 requests per day), possibly to avoid being detected. Finally, Sybils in S_5 are likely to *receive* friend requests. The ratio of incoming friend requests over outgoing ones is notably higher (0.286) than other Sybil clusters (< 0.05). One possible explanation is that these Sybils are controlled by a single attacker to befriend with each other to bootstrap their social connections.

EVALUATION: CLUSTER QUALITY

Finally, we evaluate the quality of behavioral clusters produced by our system by examining how well they capture similar users. For this analysis, we compare our algorithm with existing clustering methods.

Clustering Quality

At the high-level, an effective clustering algorithm should accurately group similar users together while separating different ones. We evaluate the quality of our behavioral clusters by testing how well they capture similar users. More specif-



(a) Sybils (Renren) (b) Dormant Users (Whisper) (c) Blocked Users (Whisper)

Figure 15. The precision and recall of using the behavioral clusters to detect certain type of users. We compare our method with K-means and Hierarchical Clustering algorithm (HC).

ically, given a small sample of known users, how accurately can they retrieve other users of the same type?

Experiment Setups. We first explain our experiment method, using Sybil detection in Renren as an example. Suppose a small sample of Sybils are known to us (x%). To detect the rest of the Sybil accounts, we use the known samples as *seeds* to color Renren's behavioral clusters. Any cluster that contains a known Sybil will be colored as Sybil-cluster (uncolored ones as normal). We evaluate the accuracy using two metrics: *Precision* (percentage of users in Sybil-clusters that are true Sybil accounts) and *Recall* (percentage of true Sybils that are captured by Sybil-clusters). A higher precision and recall indicate a better clustering quality. We vary the parameter x (1%, 5%, 10%) and repeat each experiment 10 times.

To perform this experiment on Whisper dataset, we need to construct known groups of users. We use the two types of users identified in earlier analysis: *Dormant* users who have zero active events (16688 users) and *Blocked* users who have been blocked at least once in a private chat (68302 users).

Comparison Baselines. Our baselines are two widely used clustering algorithms: K-means [9] and Hierarchical Clustering (HC) [13]. We run both algorithms to cluster the full similarity graph (without feature pruning). At the high-level, K-means divides users into K clusters where each user is assigned to the nearest cluster (center). The number of clusters K must be pre-defined. Here we generate multiple versions of K-means clusters, and pick the K with the highest clustering quality (modularity). As a result, K-means generates 10 clusters on the Renren dataset and 10 for Whisper. In the same way, HC generates 7 clusters for Whisper and 2 clusters for Renren.

Results. First, for Sybil detection on Renren, our algorithm is highly accurate with a precision of 93% and a recall of 94% (1% ground-truth as seed) as shown in Figure 15(a). Using more seeds (*e.g.* 5%) produces a higher recall (99%) but reduces precision (82%). Nonetheless, the overall performance is better than K-means and HC (precision 67% and 61%). On the Whisper dataset, our algorithm achieves accurate results (98% precision, 100% recall) in identifying dormant users (Figure 15(b)). K-means and HC have a much lower precision (32% and 78%) with the same recall. Finally, all three algorithms achieve similar accuracy in detecting blocked users (73% precision and 99% recall). These results indicate that our system produces high quality clusters to capture similar users.

CONCLUSION & FUTURE WORK

In this work, we describe a practical clickstream tool to model online user behavior. Our tool captures complex human behaviors while presenting them in a simple and intuitive manner. For a given clickstream dataset, it automatically identifies clusters of users with similar clickstream activities, and captures the natural hierarchical structure for user clusters. With a visualization tool, service providers can explore dominating user behaviors and categories as an overview, while tracking fine-grained user behavioral patterns along each category. Our tool does not require prior knowledge or assumptions of user categories (unsupervised), thus it can effectively capture unexpected or previously unknown behaviors. We demonstrate its effectiveness using case studies on two large-scale online social networks. Our tool accurately identifies unusual behaviors (malicious Sybils, hostile users) and even predicts users' future activities (dormant users). Finally, we shared our tool and results with the Whisper Data Science team. While we are awaiting more detailed comments, the initial feedback was extremely positive.

Broader Applications. We believe our proposed techniques are generalizable beyond online social networks. To obtain clickstream traces, service providers can extract "user events" from their HTTP logs. In this paper, we define user events based on social network features. For other services, specific events will depend on the service functionalities. For example, Wikipedia, News or Q&A sites [28] might extract events based on the category or topic of the pages. E-commerce web sites can define user events based on the functionality of the clickable links or product categories. Crowdsourcing sites can define click events based on the crowdsourcing workflow. In future work, we will explore broader applications of clickstream behavioral models, and expand our tool to other user-driven systems.

Acknowledgments

We would like to thank Ulas Bardak, Sarita Schoenebeck, Megan McQueen and the anonymous reviewers for their helpful comments. This project was supported by NSF grants CNS-1527939 and IIS-1321083. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any funding agencies.

REFERENCES

1. E. Adar, J. Teevan, and S. T. Dumais. 2008. Large Scale Analysis of Web Revisitation Patterns. In *Proc. of CHI*.
2. F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida. 2009. Characterizing User Behavior in Online Social Networks. In *Proc. of IMC*.
3. N. Bhatti, A. Bouch, and A. Kuchinsky. 2000. Integrating user-perceived quality into Web server design. *Computer Networks* 33, 1–6 (2000), 1 – 16.
4. V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. 2008. Fast unfolding of communities in large networks. *JSTAT* 2008, 10 (2008).
5. A. Brahaj. 2009. English Stop Words. `http://xpo6.com/list-of-english-stop-words/`. (2009).
6. D. Correa, L. A. Silva, M. Mondal, F. Benevenuto, and K. P. Gummadi. 2015. The Many Shades of Anonymity: Characterizing Anonymous Social Media Content.. In *Proc. of ICWSM*.
7. R. S. Geiger and A. Halfaker. 2013. Using Edit Sessions to Measure Participation in Wikipedia. In *Proc. of CSCW*.
8. S. Gündüz and M. T. Özsu. 2003. A Web page prediction model based on click-stream tree representation of user behavior. In *Proc. of SIGKDD*.
9. J. Hartigan and M. Wong. 1979. Algorithm AS 136: A k-means clustering algorithm. *Applied statistics* (1979), 100–108.
10. J. Heer and E. H. Chi. 2002. Separating the swarm: categorization methods for user sessions on the web. In *Proc. of CHI*.
11. M. E. Houle, H. Kriegel, P. Kröger, E. Schubert, and A. Zimek. 2010. Can Shared-neighbor Distances Defeat the Curse of Dimensionality?. In *Proc. of SSDBM*.
12. B. Johnson and B. Shneiderman. 1991. Tree-Maps: A Space-filling Approach to the Visualization of Hierarchical Information Structures. In *Proc. of VIS*.
13. L. Kaufman and P. Rousseeuw. 2009. *Finding groups in data: an introduction to cluster analysis*. Vol. 344. John Wiley & Sons.
14. J. B. Kruskal and J. M. Landwehr. 1983. Icicle plots: Better displays for hierarchical clustering. *The American Statistician* 37, 2 (1983), 162–168.
15. L. Lu, M. Dunham, and Y. Meng. 2005. Mining significant usage patterns from clickstream data. In *Proc. of WebKDD*.
16. J. Matejka, T. Grossman, and G. Fitzmaurice. 2013. Patina: Dynamic Heatmaps for Visualizing Application Usage. In *Proc. of CHI*.
17. M. Motoyama, D. McCoy, K. Levchenko, S. Savage, and G. M. Voelker. 2011. An Analysis of Underground Forums. In *Proc. of IMC*.
18. H. Obendorf, H. Weinreich, E. Herder, and M. Mayer. 2007. Web page revisitation revisited: implications of a long-term click-stream study of browser usage. In *Proc. of CHI*.
19. J. Y. Park, N. O'Hare, R. Schifanella, A. Jaimes, and C. Chung. 2015. A Large-Scale Study of User Image Search Behavior on the Web. In *Proc. of CHI*.
20. J. M. Rzeszotarski and A. Kittur. 2011. Instrumenting the Crowd: Using Implicit Behavioral Measures to Predict Task Performance. In *Proc. of UIST*.
21. N. Sadagopan and J. Li. 2008. Characterizing Typical and Atypical User Sessions in Clickstreams. In *Proc. of WWW*.
22. S. Salvador and P. Chan. 2004. Determining the Number of Clusters/Segments in Hierarchical Clustering/Segmentation Algorithms. In *Proc. of ICTAI*.
23. J. Srivastava, R. Cooley, M. Deshpande, and P. N. Tan. 2000. Web usage mining: discovery and applications of usage patterns from Web data. *SIGKDD Explor. Newsl.* 1, 2 (2000), 12–23.
24. J. Stasko and E. Zhang. 2000. Focus+ context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *Proc. of InfoVis*.
25. Q. Su and L. Chen. 2015. A method for discovering clusters of e-commerce interest patterns using click-stream data. *ECRA* 14, 1 (2015), 1–13.
26. J. Suler and W. L. Phillips. 1998. The Bad Boys of Cyberspace: Deviant Behavior in a Multimedia Chat Community. *Cyberpsy., Behavior, and Soc. Networking* 1, 3 (1998), 275–294.
27. I. Ting, C. Kimble, and D. Kudenko. 2005. UBB Mining: Finding Unexpected Browsing Behaviour in Clickstream Data to Improve a Web Site's Design. In *Proc. of ICWI*.
28. G. Wang, K. Gill, M. Mohanlal, H. Zheng, and B. Y. Zhao. 2013a. Wisdom in the Social Crowd: an Analysis of Quora. In *Proc. of WWW*.
29. G. Wang, T. Konolige, C. Wilson, X. Wang, H. Zheng, and B. Y. Zhao. 2013b. You Are How You Click: Clickstream Analysis for Sybil Detection. In *Proc. of USENIX Security*.
30. G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, and B. Y. Zhao. 2014. Whispers in the Dark: Analysis of an Anonymous Social Network. In *Proc. of IMC*.
31. W. Wang, H. Wang, G. Dai, and H. Wang. 2006. Visualization of Large Hierarchical Data by Circle Packing. In *Proc. of CHI*.
32. J. Wei, Z. Shen, N. Sundaresan, and K. Ma. 2012. Visual cluster exploration of web clickstream data. In *Proc. of VAST*.
33. Y. Yang and J. O. Pedersen. 1997. A Comparative Study on Feature Selection in Text Categorization. In *ICML*.

34. Z. Yang, S. Cai, Z. Zhou, and N. Zhou. 2005. Development and validation of an instrument to measure user perceived service quality of information presenting Web portals. *Information & Management* 42, 4 (2005), 575 – 589.

35. J. Zhao, Z. Liu, M. Dontcheva, A. Hertzmann, and A. Wilson. 2015. MatrixWave: Visual Comparison of Event Sequence Data. In *Proc. of CHI*.

APPENDIX – IMPLEMENTATION AND SCALABILITY

Our system is implemented in python, and runs on 9 servers (HP DL360P Gen8). For Whisper dataset (100K users), it takes about 58 hours to generate the complete behavioral clusters. For Renren dataset (16K users), it only takes 47 minutes. This performance is already sufficient for practical usage to build behavioral models on a weekly basis. Because these servers are a shared resource with other research teams, we only take 4 threads per server. Potentially, we can speed this up by increasing the server utility (*e.g.*, 40 threads).

The major computational bottleneck is constructing the similarity graph where we compute pair-wise clickstream similarity. For a dataset of n users, the time complexity is $O(n^2)$. To scale up the system for even larger datasets (*e.g.*, 10 million), one cannot simply add more machines to handle the $O(n^2)$ complexity. A promising approach is *incremental clustering*. The key idea is to take a small sample from a clickstream dataset to build the initial clusters (*e.g.*, K clusters). Then we incrementally assign the rest of the users to existing clusters, based on their distance to the "centers" of existing clusters. In this way, we only need to compute each node's distance to the K cluster centers. The time complexity becomes $O(nK)$ where K is a small number. We leave the implementation and evaluation of this approach to future work.