Identifying Frequent User Tasks from Application Logs

Himel Dev
Department of Computer Science
University of Illinois at Urbana-Champaign
hdev3@illinois.edu

Zhicheng Liu
Adobe Research
San Francisco, CA
leoli@adobe.com

ABSTRACT

In the light of continuous growth in log analytics, application logs remain a valuable source to understand and analyze patterns in user behavior. Today, almost every major software company employs analysts to reveal user insights from log data. To understand the tasks and challenges of the analysts, we conducted a background study with a group of analysts from a major software company. A fundamental analytics objective that we recognized through this study involves identifying frequent user tasks from application logs. More specifically, analysts are interested in identifying operation groups that represent meaningful tasks performed by many users inside applications. This is challenging, primarily because of the nature of modern application logs, which are long, noisy and consist of events from high-cardinality set. In this paper, we address these challenges to design a novel frequent pattern ranking technique that extracts frequent user tasks from application logs. Our experimental study shows that our proposed technique significantly outperforms state of the art for real-world data.

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous; H.2.8. Database Management: Database Applications

Author Keywords

User Task; Application Log; Frequent Pattern Mining; Pattern Ranking

INTRODUCTION

Application log data contain valuable information about user behavior that can inform technical or business decisions. With effective analysis of such data, marketers may be able to correlate user behavior with marketing goals (e.g. successful purchase) and improve promotion strategies; application developers may better prioritize features in product roadmaps, discover potential bugs and make automatic recommendations without interrupting users' workflow. These valuable applications drive companies and organizations to employ data analysts to reveal user insight from log data.

IUI 2017, March 13–16, 2017, Limassol, Cyprus.

http://dx.doi.org/10.1145/3025171.3025184

To understand the tasks and challenges of the analysts, we conducted a background study with a group of log analysts from a major software company. Based on our study, we find that identifying meaningful and frequent user tasks is an important milestone in many log analysis scenarios. For example, to recognize what features needs to be prioritized in product roadmaps, product mangers urge to identify the major user requirements, which can be revealed by examining user tasks. Similarly, to automatically recommend workflows, it is crucial to recognize user intent, which can be modeled based on frequent tasks.

While identifying frequent user task is important, doing so for large-scale, complex log data is challenging. The challenges arise from two main sources: the volume and complexity of the data, and the diversity in user behavior and domain context. The current practice of identifying frequent user tasks involves applying frequent pattern mining techniques such as frequent itemset mining and sequential pattern mining [3, 4, 5, 12]. These techniques, however, are not customized for frequent tasks and can not address the aforementioned challenges.

In this paper, we present a novel frequent pattern ranking technique to extract frequent user tasks from application logs. The key idea of our approach is to rank patterns based on *membership based cohesion*, which prioritizes the patterns whose events appear contagiously in the supporting sequences with no or few outliers (events not belonging to the pattern). We apply our technique on real-world log data and conduct a user study to evaluate the effectiveness of our approach. Our experimental study shows that our approach significantly outperforms state of the art for a variety of standard metrics such as NDCG and P/R@k.

In summary, our contributions are as follows:

- We conduct a three-phase background study to understand the vitals of log analytics . Our study reveals many idiosyncrasies of log analytics that we present in the Objectives, Data and Techniques sections.
- We formulate the frequent task identification problem using a set of example-driven assumptions and propose a novel frequent pattern ranking technique to solve this problem. We present the details in the Identifying Frequent User Tasks section.
- We conduct a user study to evaluate the effectiveness of our approach using real-world data. We compare our approach with state of the art using a standard set of metrics. We provide the details in Evaluation section.

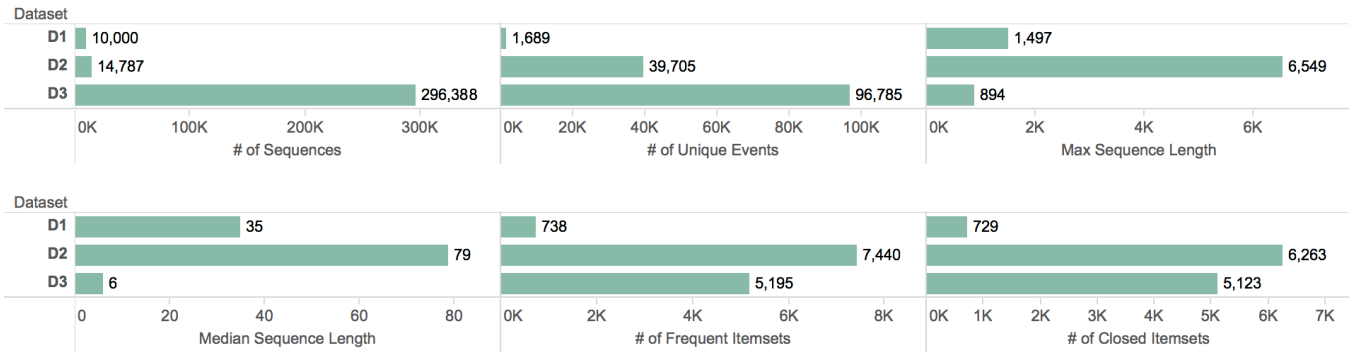

Figure 1. Statistics on the datasets we have analyzed

BACKGROUND STUDY

To understand the scale and complexity of log data and the techniques and challenges of associated analysis, we conducted a mixed-methods study consisting of three phases. In the first phase, we conducted in-depth interviews to understand the vitals of log analytics including the primary objectives, characteristics of data and state-of-the-art techniques. In the second phase, we performed some of the analysts' work routines to get a first-hand experience on both the data and techniques. In the third phase, we conducted a pilot study to get user feedback on the findings of our analysis. Note that we recruited the same group of participants in the first and third phase of study.

Participants

As this study requires understanding the vitals of log analytics, the domain knowledge of participants is of critical importance. To address this concern, we recruited two groups of people who are practitioners and stakeholders of log analytics. The first group consists of three analysts who work for a major software company that sells multimedia and creativity software products. The daily jobs of the analysts involve analyzing and reporting user behavior on the company's software applications. They execute queries to retrieve log data from Hadoop servers, wrangle and analyze the data, and prepare presentations on any insights they have found. The second group consists of two product managers who are domain experts on the software applications. We included product managers because they work with a much larger set of users and offer a high-level perspective on the common tasks shared by the analysts.

OBJECTIVES, DATA AND TECHNIQUES

In this section, we discuss the findings of our background study, which reveal many idiosyncrasies of log analytics.

Objectives

Based on our interviews, we recognize *frequent task identification* as one of primary objectives of log analytics. For example, some analysts want to understand what users are doing at a certain phase/period: *"What do most users do in their early projects?"*. Likewise, some want to understand the evolution of users: *"Are users performing similar/different tasks in their early and later projects?"*. Again, some care about the elements of a successful project: *"What are some common tasks present in successful (published) projects?"*. We note that many such analytics goals require identifying and understanding frequent user tasks. Though we recognize other relevant objectives of log analytics, such as determining usefulness of a particular application feature, we focus on identifying frequent user tasks as it is fundamental to both analysts and managers we interviewed.

Data

Based on our first-hand analysis and the interviews, we identify the following distinguishing characteristics of log data.

- **The cardinality of event set is large.** The number of possible user operations in a modern software application can range from hundreds to tens of thousands. Each of these operations triggers one or more events that are recorded in application logs. Typically, logs are pre-processed to reflect the user operations.
- **A session can have an arbitrary length.** While most users use software applications for a few minutes or hours in a session, others leave them open for days. It is common for a session to contain several hundred events, albeit the upper bound is much higher.
- **There is substantial variety in user behavior.** Users may perform the same task in a variety of ways. This variety arises from the presence of different operation combinations that can be used to achieve the same outcome.
- **Application logs are noisy.** A sizable fraction of users execute a rather *hodgepodge of operations* to perform an intended task, which often includes operations that are not required for performing the task. These misplaced operations act as a source of noise. Noise can also arise from mistakes, i.e., unintended user operations.

Techniques

Because we focus on frequent task identification, we shall only discuss the techniques that are relevant to this problem. Based on our study, we find that the analysts apply frequent pattern mining techniques such as frequent itemset mining [3] and sequential pattern mining [4] on log data to identify frequent user tasks. A major challenge of applying frequent

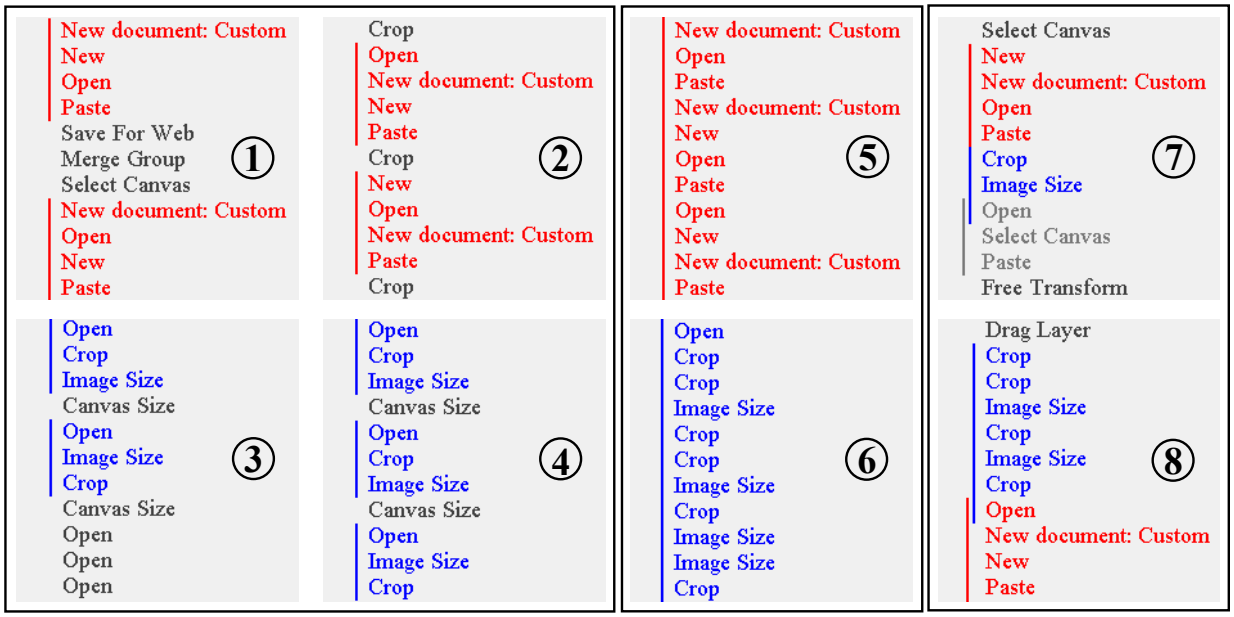

Figure 2. Event sequence fragments from eight distinct logs (corresponding to a photo editing application) shows instances of T_1 (version management task) and T_2 (cropping task). The instances of T_1 are highlighted in red font, with red vertical lines outlining the instance boundaries. The instances of T_2 are highlighted in blue font, with blue vertical lines outlining the instance boundaries. For T_2, `Paste` implies `Copy` followed by `Paste`.

pattern mining techniques is the fact that such techniques often generate a huge number of patterns [2]. For example, we applied several classes of frequent pattern mining techniques on real-world datasets, and the results (Figure 1) show that the number of patterns can be overwhelming, even for very strict classes such as closed patterns. Further, most of these patterns are not very useful to the analysts as they do not represent user tasks. Consequently, analysts face the formidable challenge of manually exploring the patterns and their supporting logs to identify the useful patterns representing user tasks. We extensively study the literature of frequent pattern mining in search of technique(s) that can address this problem. While we find numerous works addressing the concern of finding useful patterns, none entirely solves this problem, primarily due to the concept of *pattern usefulness* being context dependent. Indeed, patterns that are useful in one context, may not be relevant in another. For this reason, the problem of finding appropriate patterns representing user tasks remains open to us.

IDENTIFYING FREQUENT USER TASKS

In this section, we discuss our problem formulation, review the relevant concepts of frequent pattern mining and present our solution to resolve the frequent task identification problem.

Problem Formulation

While the idea of *user task* is easy to understand, a formal formulation in log analytics is absent. For example, one could define it as a *sequence* of user operations to actualize user intent. Another way is to define it as a *set* of user operations to achieve some milestone. While the former considers ordering of operations, the latter does not. There are other such factors (e.g., adjacency of operations) that one needs to consider while defining user task. We define user task based on a set of example-driven assumptions that we recognized during our first-hand analysis and later confirmed with the analysts. Before we report these assumptions ($\mathbf{A_1}$ to $\mathbf{A_4}$), we present two well-acknowledged tasks ($\mathbf{T_1}$ and $\mathbf{T_2}$) that we shall use to explain the assumptions.

T_1: This task involves creating a new file from an existing one via copy pasting. This is a naive version management strategy commonly used by novice users to preserve the content of a file while attempting exploratory operations. In Figure 2, the instances of this task are highlighted in red font, with red vertical lines outlining the instance boundaries.

T_2: This task involves cropping an image and checking the size of its dimensions. This is yet another common task performed by many users to generate an image with specific length and/or width. In Figure 2, the instances of this task are highlighted in blue font, with blue vertical lines outlining the instance boundaries.

A_1: Operations corresponding to a task may or may not have any associated ordering.

The first two event sequence fragments in Figure 2 show different operation orderings for $\mathbf{T_1}$. There are two alternative operation sequences that can be executed to perform this task. The first alternative is to (i) open the existing file, (ii) copy items from the existing file, (iii) create a new file, and (iv) paste items in the new file. The second alternative is to (i) create a new file, (ii) open the existing file, (iii) copy items from the existing file, and (iv) paste items in the new file. While the operations corresponding to the task have some partial ordering (open followed by copy followed by paste, new followed by paste), there is no absolute order. The third and fourth event sequence fragments in Figure 2 show two

operation orderings for $\mathbf{T_2}$. For this task, users may check the image size either before or after cropping. Our discussion with the analysts reveal that there are many such tasks where the operation ordering is variable. While the aforementioned tasks have two alternative operation orderings, other tasks may have more than two alternatives.

$\mathbf{A_2}$: A user may execute a required operation multiple times within the duration of a task.

The fifth and sixth event sequence fragments in Figure 2 show repetition of operations for $\mathbf{T_1}$ and $\mathbf{T_2}$. Our first hand analysis reveals that users often repeat operations in a loop till they achieve the intended results. Within the task duration, a user may repeat a required subset of operations in a loop.

$\mathbf{A_3}$: A user may perform multiple tasks in a single session.

The seventh and eigth event sequence fragments in Figure 2 show users performing at least two different tasks ($\mathbf{T_1}$ and $\mathbf{T_2}$) in a session. In fact, most users perform many different tasks in a session.

$\mathbf{A_4}$: To perform a task, a user executes the corresponding operations contiguously, with no or few *outliers* (operations that are not part of the task).

In all eight event sequence fragments of Figure 2, we see users executing the operations of a task contiguously, without any outlier (unnecessary operation). In some cases, users may execute one or more unnecessary operations within a task by mistake, yet, such occurrences should be few.

Frequent Pattern Mining

In this subsection, we review the concepts of frequent pattern mining. There has been extensive research on this topic and we essentially discuss the concepts relevant to our problem.

Preliminaries

We will be using the following definitions to discuss different classes of frequent patterns.

DEFINITION 1. *An **event sequence** $S = [E^1, E^2, \ldots, E^m]$ ($E^i \in E$) is an ordered list of events, where E denotes the event dictionary and i denotes the order of event E^i in S.*

DEFINITION 2. *A **sequence database** $D = \{S_1, S_2, \ldots, S_n\}$ is an unordered set of sequences.*

DEFINITION 3. *In our discussion, a **pattern** P is either (i) a set of events whose members appear in random order, or (ii) a sequence of events that appear as subsequence(s), in one or more sequences in an event sequence database.*

DEFINITION 4. *The **support set** D_P of a pattern P in a sequence database D is the largest subset of D where P appears in all sequences belonging to D_p. The **support** of P is quantified as the percentage ratio of the size of D_p and D.*

DEFINITION 5. *__Frequent patterns__ $F = \{P_1, P_2, \ldots, P_f\}$ is a set of patterns (of same type) where the support of each pattern in a given database is no less than a user-specified threshold Θ.*

DEFINITION 6. *The **occurrence window** $W_{P,S}$ of a pattern P in a sequence S refers to the interval(s) within S that contains P.*

DEFINITION 7. *The **minimum length occurrence window** or **minimum occurrence window** $W_{P,S}^{(L-)}$ of a pattern P in a sequence S refers to the minimum length interval(s) within S that contains P. Here, the function $L()$ returns length and the superscript $(L-)$ denotes minimum length.*

$$W_{P,S}^{(L-)} = \arg\min_{W_{P,S}} L(W_{P,S}) \quad (1)$$

Frequent Pattern Classes

We discuss different classes of frequent patterns (with examples in Table 1) based on two dimensions relevant to our problem.

Order Dimension: Frequent patterns extracted from an event sequence database can either be sets or sequences (of events). A set based frequent pattern does not consider the order of events in database sequences and is called a *frequent itemset* [3]. In contrast, a sequence based frequent pattern considers the order of events in database sequences and is called a *sequential pattern* [4].

ID	Sequence
S1	[A,C,D]
S2	[B,C,E]
S3	[E,A,B,C]
S4	[B,E]
S5	[E,B,A,C]

(a) Input sequences

Class	Patterns from Input Sequences
Frequent Itemset	{A},{B},{C},{E},{A,B},{A,C},{A,E}, {B,C},{B,E},{C,E},{A,B,C},{A,B,E}, {A,C,E},{B,C,E},{A,B,C,E}
Cohesive Itemset	{A},{B},{C},{E},{A,B},{A,C},{B,C}, {B,E},{A,B,C},{A,B,E},{A,B,C,E}
Sequential Pattern	[A],[B],[C],[E],[A,C],[B,C],[B,E], [E,A],[E,B],[E,C]
2-gram	[B,C]
Episode	[A],[B],[C],[E],[A,C],[B,C]

(b) Patterns with $>= 40\%$ support. For cohesive itemsets and episodes, we used the ratio of pattern length (number of elements in pattern) and occurrence window length as the cohesion parameter, and set the threshold value to 1.

Table 1. Examples of frequent patterns

Cohesion Dimension: Cohesion or adjacency of pattern elements (events) in support set (supporting database sequences) is a central concept in our problem. The pattern classes that address the concern of cohesion includes N-gram, episode and cohesive itemset. N-gram and episode are subclasses of sequential pattern, whereas cohesive itemset is a subclass of frequent itemset. An *N-gram* is a sequential pattern whose elements appear contiguously in supporting sequences, with

no outlier. An *episode* is a sequential pattern, for which the average length of minimum occurrence windows in supporting sequences is smaller than a user-specified threshold [12]. A *cohesive itemset* is similar except that it is a frequent itemset [5].

Limitations

We study the state-of-the-art frequent pattern mining techniques to determine if any of these techniques can solve the problem of frequent task identification. We find that the existing techniques fail to satisfy one or more assumptions that we reported (Table 2), and consequently fall short in solving the problem.

- Order-sensitive patterns such as sequential patterns and frequent episodes fail to satisfy assumption $\mathbf{A_1}$. More specifically, if a task has many alternative operation orderings, then none of these orderings could be frequent for a given threshold.
- Cohesion-insensitive patterns such as frequent itemsets and sequential patterns fail to satisfy assumption $\mathbf{A_4}$. More specifically, these patterns do not distinguish between the set of events or operations that appear adjacently and the set of operations that appear randomly in supporting sequences. For example, the subsets of top k popular operations may appear in many sequences, however, the operations of these sets may or may not be executed as a group to achieve some milestone.
- Cohesion-insensitive patterns such as frequent itemsets and sequential patterns also fail to satisfy assumption $\mathbf{A_3}$. In particular, these patterns may contain operations from several tasks.
- Finally, the existing cohesion-sensitive patterns such as frequent episodes and cohesive itemsets fail to satisfy assumption $\mathbf{A_2}$. In these patterns, cohesion is determined in terms of the lengths of minimum occurrence windows, which fail to satisfy the assumption of a required operation to be repeated arbitrary number of times.

State-of-the-Art	$\mathbf{A_1}$	$\mathbf{A_2}$	$\mathbf{A_3}$	$\mathbf{A_4}$
Frequent Itemset	✓	✓	×	×
Sequential Pattern	×	✓	×	×
Frequent Episode	×	×	✓	✓
Cohesive Itemset	✓	×	✓	✓

Table 2. Limitations of state-of-the-art techniques in terms of four assumptions. $\mathbf{A_1}$: Operations corresponding to a task may or may not have any associated ordering. $\mathbf{A_2}$: A user may execute a required operation multiple times within the duration of a task. $\mathbf{A_3}$: A user may perform multiple tasks in a single session. $\mathbf{A_4}$: To perform a task, a user executes the corresponding operations contiguously, with no or few outliers.

Membership Based Cohesion for Patterns

To address the limitation of existing approaches, we introduce the idea of *membership based cohesion* for frequent patterns. To introduce this idea, we first present the concept of *outlier based minimum occurrence window*.

DEFINITION 8. *The **outlier based minimum occurrence window** $W_{P,S}^{(O-)}$ of a pattern P in a sequence S refers to the interval(s) within S that contains P while containing minimum possible outliers (i.e., elements not belonging to the pattern P). Here, the function $O()$ returns the number of outliers and the superscript $(O-)$ denotes minimum outlier.*

$$W_{P,S}^{(O-)} = \arg\min_{W_{P,S}} O(W_{P,S}) \tag{2}$$

DEFINITION 9. *The **minimum outlier based maximum length occurrence window** or **minimum outlier based maximum occurrence window** $W_{P,S}^{(O-)(L+)}$ of a pattern P in a sequence S refers to the maximum length interval(s) within S that contains P while containing minimum possible outliers. In other words, the minimum outlier based maximum occurrence window refers to the interval(s) that contain(s) P, and includes as many elements of P as possible without including any element not belonging to P. Here, the function $L()$ returns length and the superscript $(L+)$ denotes maximum length.*

$$W_{P,S}^{(O-)(L+)} = \arg\max_{W_{P,S}^{(O-)}} L(W_{P,S}^{(O-)}) \tag{3}$$

For example, consider the sequences in Table 3. If we perform frequent itemset mining on these sequences with support threshold $\Theta = 0.5$, we get itemsets such as {A,B,C} and {D,E,F}. The minimum occurrence windows (according to DEFINITION 7) of itemset {A,B,C} in S1, S2 and S3 are marked with underlines. Contrast these to the minimum outlier based maximum occurrence windows (according to DEFINITION 9) which are marked with overlines. The former involves minimizing window length, whereas the latter involves first minimizing outlier count and then maximizing window length.

ID	Sequence
S1	[A,A,A,B,B,B,B,B,C,C,C,D,**B**,**B**,E,F]
S2	[G,H,I,J,A,B,A,B,A,B,A,B,C]
S3	[D,F,**X**,**Y**,E,B,A,B,A,B,A,C,C]

Table 3. Example sequences

Membership Based Cohesion

We formulate the cohesion of a pattern as the signed difference between the pattern's length (number of elements in pattern) and the median outlier count in its minimum outlier based maximum occurrence windows or outlier based minimum occurrence windows.

For example, consider the sequences in Table 3. If we perform frequent itemset mining on these sequences with threshold $\Theta = 0.5$, we get itemsets such as {A,B,C} and {D,E,F}. The length of both {A,B,C} and {D,E,F} is 3. There are no outliers in the outlier based minimum occurrence windows of {A,B,C} in S1, S2 and S3, and therefore, the median outlier count is 0. According to our formulation, the cohesion score of itemset {A,B,C} is 3 - 0 = 3. In contrast, the number of outliers in the outlier based minimum occurrence windows

of {D,E,F} in both S1 and S3 is 2, and therefore, the median outlier count is 2. According to our formulation, the cohesion score of itemset {D,E,F} is 3 - 2 = 1.

Notice that, the average length of minimum occurrence windows for itemset {D,E,F} is 5, which is smaller compared to that of itemset {A,B,C}. According to state-of-the-art cohesive pattern ranking, itemset {D,E,F} is more cohesive compared to itemset {A,B,C}. However, if we consider the median outlier count in outlier based minimum occurrence windows, the order is opposite. Our formulation of membership based cohesion uses the second ordering. In other words, it prefers *small outlier count over small window length*.

Our formulation of membership based cohesion uses pattern length. There are two key reasons why we use pattern length in cohesion formulation. The first one is to allow salient patterns/tasks with more events/operations to get priority over the short patterns/tasks. The second one is based on the fact that the tasks with higher number of operations have higher room for error/outlier, which needs to be accounted for accordingly.

Frequent Task Identification

To identify frequent user tasks, we first perform frequent itemset mining on log dataset with a small threshold (e.g., 5%, 10%). Then, we rank the itemsets in descending order based on the membership based cohesion score. As a matter of fact, frequent itemset mining and the proposed ranking can be performed simultaneously. We hypothesize that the top itemsets with high membership based cohesion are likely to be frequent user tasks. Notice that the top itemsets satisfy all four assumptions ($\mathbf{A_1}$ to $\mathbf{A_4}$) that we reported. In particular, the use of outlier count in determining cohesion score resolves the concern of assumption $\mathbf{A_2}$, without failing the other assumptions.

EVALUATION

We evaluate the effectiveness of our proposed frequent task identification technique using real-world data. Our evaluation focuses on the following aspects of effectiveness:

- Q_1. How effective is our pattern ranking technique in identifying frequent user tasks?
- Q_2. How effective is our ranking technique compared to the state of the art?
- Q_3. How meaningful are the resultant patterns, i.e., the potential frequent user tasks?

Dataset

We applied our frequent task identification technique on a real-world log dataset. The logs are from a desktop based photo editing application that has several million users. Two of the analysts from our background study work with similar logs originating from the same application. We took a representative sample of the logs to compile our dataset. Our dataset has 10,000 event sequences and accommodates 1497 unique events. The median length of sequence in the dataset is 35.

Method

Our evaluation method consists of two phases. In the first phase, we performed frequent itemset mining and ranked the resultant itemsets based on two different ranking criteria. In the second phase, we selected a representative sample from these itemsets and conducted a user study to evaluate the potential of the selected itemsets to represent frequent user tasks.

Phase I: Frequent Itemset Mining and Ranking

We applied frequent itemset mining on our log dataset using 10% support threshold and acquired 738 itemsets. From these itemsets, we selected 540 itemsets with *length >= 3* and ranked those based on membership based cohesion. We used standard competition ranking, i.e., "1224" ranking to rank the itemsets [22] . We found that in the ranked itemsets cohesion score rapidly dropped after the first few entries and based on this observation, we determined a cut-off score (*cut-off score = 2*) to categorize the itemsets into two groups. The first group consists of the top 16 itemsets that are highly likely to be frequent user tasks and the second group consists of the remaining itemsets. The rationale behind selecting 2 as a cut-off score was to maintain a reasonable number of itemsets in user evaluation as these evaluations are challenging for users.

In addition to our ranking, we ranked the 540 itemsets with *length >= 3* based on state of the art cohesive itemset mining/ranking technique proposed at [5]. We used standard competition ranking for this procedure as well.

Phase II: User Study

Based on the two rankings mentioned above, we selected 36 itemsets covering each of the following groups: (i) top 16 itemsets based on our ranking, (ii) top 16 itemsets based on state of the art ranking, (iii) randomly selected itemsets from remaining (*rank > 16* in both ranking). We conducted a user study to evaluate the potential of these selected itemsets to represent frequent user tasks.

Participants: We conducted our study with ten participants, users of the creativity application, who have intermediate to expert level experience with the application. Some of these users are log analysts. We recruited participants by sending invitation via email within an organization that heavily use the application. Each of our participants had a minimum one year experience with the application.

Tool: To conduct our user study, we developed a tool that allows a user to browse the supporting sequences of a selected pattern. The tool marks the minimum outlier based maximum occurrence windows of a selected pattern using vertical lines. It also highlights the elements belonging to the pattern. Figure 3 shows a screenshot of the tool.

Method: Our user study was a combination of exploration and survey. In the exploration phase, we asked participants to browse the supporting sequences of a selected itemset to understand the usage of the corresponding operation group. For each itemset, we asked participants to browse at least 10 supporting sequences while recommending them to browse more sequences if required. After browsing phase, in survey phase, we asked participants to rate the likelihood of the itemset (operation group) to represent frequent user task in a scale of 1 to 5. We also asked them to explain their decision and

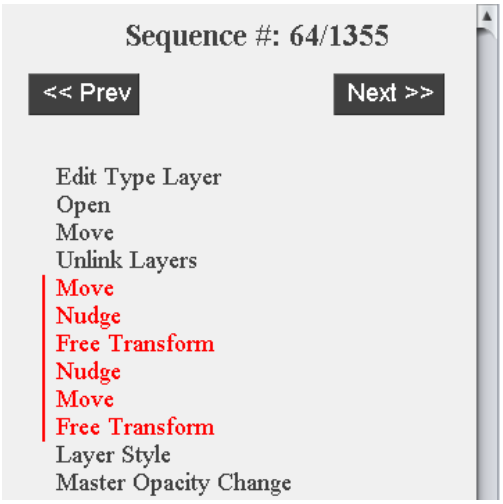

Figure 3. Screenshot of our user study tool

explain the task itself in a few sentence(s). We repeated this procedure for all 36 itemsets.

Evaluation Metrics

We use the following evaluation metrics to investigate the three questions that we reported.

- M_1. To evaluate the effectiveness of our pattern ranking technique, we compare the task scores for two groups of itemsets, (i) top 16 and (ii) remaining.
- M_2. To compare the effectiveness of our ranking with that of state of the art, we use information retrieval based metrics such as normalized discounted cumulative gain (NDCG) and precision/recall at rank k (P/R@k).
- M_3. To evaluate the interpretability of our results, we perform in-depth analysis on the itemsets/operation-groups that potentially represent frequent user tasks.

Normalized Discounted Cumulative Gain (NDCG)

NDCG is the standard metric to measure a model's ranking quality in information retrieval. It measures the performance of a ranking technique based on the graded goodness of the ranked entities. It varies from 0.0 to 1.0, with 1.0 representing the ideal ranking of the entities. Here we give a brief background on NDCG.

The cumulative gain (CG) of a ranking model's ordering is the sum of goodness scores over the ranked entities. CG at a particular rank position p is defined as:

$$CG_p = \sum_{1}^{p} rel_i \tag{4}$$

where rel_i is the relevance of the result at position i.

Discounted cumulative gain (DCG) is the sum of each ranked entity's goodness score discounted by its position in ranking. DCG is therefore higher when top quality entities are ranked higher in results, and lower when they are ranked lower. DCG is defined as:

$$DCG_p = \sum_{1}^{p} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{5}$$

NDCG is defined as the ratio of a model's DCG and the ideal DCG (IDCG):

$$nDCG_p = \frac{DCG_p}{IDCG_p} \tag{6}$$

Precision/Recall at Rank k (P/R@k)

Precision and recall are set-based metrics for binary classification tasks. In a ranking context, the relevant entities should be present within the top k entries. Therefore, to evaluate a ranking technique, measuring precision and recall score at rank k can serve as a good metric. Another advantage of using P/R@k is the scores are easy to interpret.

Experimental Results

In this subsection, we report the results of our experimental evaluation.

Top 16 vs Remaining

Based on the user ratings from our study, we calculate the task score for each selected itemset as the average rating provided by the users. We compare the task scores for the top 16 itemsets and the remaining itemsets, for the two rankings. Figure 4 shows the boxplot summary of task scores for the two groups of itemsets for our ranking, whereas Figure 5 shows similar plot for state of the art ranking.

Notice that for our ranking, the task scores for the top 16 itemsets are much higher compared to that of remaining itemsets. Among the top 16 itemsets, 14 itemsets received task scores ranging from 4.3 to 4.8, whereas the other 2 received 3.6 and 3.5. Among the 20 itemsets outside top 16, only 2 itemsets received more than 3 (3.4 and 3.5). As a matter of fact, these 2 itemsets with high scores are reasonably well ranked (rank 17 and 22) in our ranking.

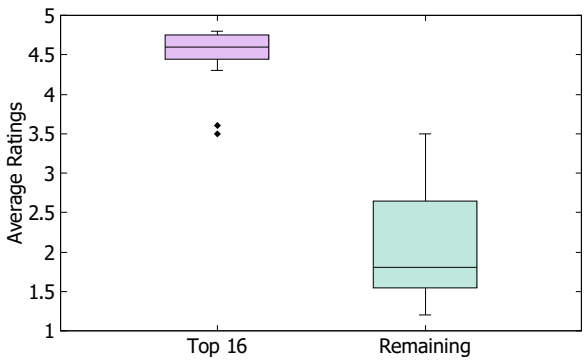

Figure 4. Boxplot summary of task scores (average user ratings) for two groups of itemsets based on our ranking

For state of the art ranking, the difference of task scores for the two groups of itemsets is not clear. While the average task score is higher for the top 16 itemsets, there are many itemsets outside top 16 that received high task scores.

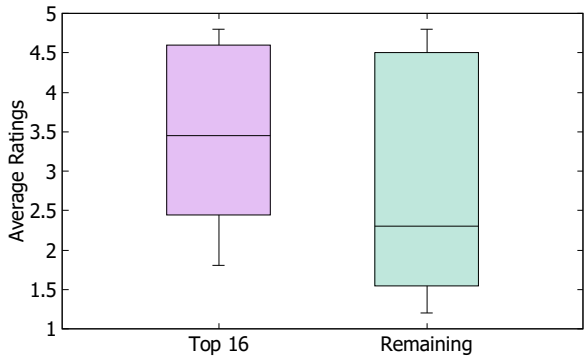

Figure 5. Boxplot summary of task scores (average user ratings) for two groups of itemsets based on state of the art ranking

Comparison with State of the Art

To compare our ranking with state of the art, we compute NDCG and P/R@k for both ranking.

NDCG: To compute NDCG, we only use the itemsets that are within top 16 either in our ranking or in state of the art ranking. There are 26 such itemsets. For these itemsets, we use the average user ratings as graded relevance scores. Further, instead of using the actual ordering of these itemsets in a ranking, we use relative ordering of these 26 itemsets as per the ranking. We do this to avoid ranking gaps for the itemsets with *rank* > *16* in both rankings.

Ranking Method	DCG	nDCG
Our Method	187.77	0.95
State of the Art	135.27	0.69

Table 4. DCG and nDCG score for two rankings

Table 4 shows the DCG and nDCG scores for the two rankings. Notice that the NDCG score for our ranking (0.95) is much higher compared to that of state of the art (0.65). This implies that our ranking is more effective in placing relevant itemsets on top.

P/R@k: We compute P/R@k for the two rankings, for different k values. To compute P/R@k, we classify the itemsets with average ratings greater than 3 as potential tasks, whereas the other itemsets as not likely to be tasks.

k	Our Method		State of the Art	
	P@k	R@k	P@k	R@k
2	1.0	0.11	1.0	0.11
4	1.0	0.22	0.5	0.11
6	1.0	0.33	0.5	0.17
8	1.0	0.44	0.63	0.28
10	1.0	0.56	0.6	0.33
12	1.0	0.67	0.58	0.39
14	1.0	0.78	0.64	0.5
16	1.0	0.89	0.56	0.5

Table 5. P/R@k for two rankings

Table 5 shows the P/R@k scores for the two rankings, for different k values. Notice that for most k values, precision and recall scores for our ranking is much higher compared to that of state of the art.

Potential Frequent Tasks

Based on the average ratings from our user study, we identify the itemsets with average ratings greater than 3 as potential tasks, whereas the other itemsets as not likely to be tasks. In Table 6, we present user provided explanation of these potential tasks.

As per our user study, both intermediate and expert users are familiar with many of the potential tasks: *"That makes sense."*, *"This is an obvious task."*, *"This is what I expect. This is 100% a task."*.

In fact, they could correlate to some of these tasks: *"I can imagine this being a task. I do this a lot."*, *"This is something I do a lot too."*.

Yet, at times some users recognized an itemset as a task based on the example operation sequences: *"I am identifying this as a task based on the example logs."*, *"I am not familiar with 'Nudge'. This is based on the sequences I have seen."*

While many of the potential tasks didn't surprise the users, some did: *"I don't understand why they are deleting the layer."*, *"Weird, so many people are doing it."*.

Users also validated our assumption regarding the ordering of operations in a task (assumption $\mathbf{A_1}$): *"Fair mix of orders; I am not sure if there is any order for this task."*, *"This is pretty consistent, the sequence doesn't matter."*

IMPLICATIONS

Our work draws from, and has implications for, several research threads.

Implications for User Behavior Modeling

Understanding user behavior is crucial for the design and operation of modern software applications. A number of early studies analyzed log data to model user behavior. For example, [1, 14] study web revisitation behavior of users by analyzing web logs. [15] studies image search behavior of users by analyzing query logs. [20, 19] built system(s) to capture user behavior from clickstream data. More specific log data based user behavior models appear in [6, 18].

This paper addresses a fundamental problem in the log data based user behavior model space. We assert that understanding user task is often a crucial first step in understanding user behavior. Tasks are semantically meaningful units that offer better user insight compared to the raw action sequences.

Implications for Temporal Event Sequence Analysis

Temporal event sequence data is pervasive in many application domains, including electronic commerce and digital marketing [10, 21], user workflow and behavior analysis [20, 9], online education [17] and healthcare [16, 13]. In recent years, we have seen several research works that utilize frequent pattern mining techniques to analyze temporal event sequence data [10, 7, 16].

The concepts introduced in this paper can be useful in temporal event sequence analysis. For example, membership based

Id	Itemset/Operation-Group	User Description	R_1	R_2	$\overline{U}$
I_1	{Crop,Open,Image Size}	Cropping an image and checking the size of its dimensions	2	2	4.8
I_2	{New,Open,New Document:Custom,Paste}	Creating a new file from an existing one via copy-pasting	1	20	4.8
I_3	{Open,New Document:Custom,Paste}	Creating a new file from an existing one via copy-pasting	8	22	4.8
I_4	{Open,Paste,Select Canvas}	Selecting the entire canvas before pasting; mostly done by novice users	8	25	4.8
I_5	{Free Transform,Nudge,Move}	Applying different move operations to put something in correct position	2	7	4.7
I_6	{New,Open,New Document:Custom}	Opening a file	2	1	4.6
I_7	{Edit Type Layer,New Type Layer,Move}	Editing and moving text	2	11	4.6
I_8	{Free Transform,Edit Type Layer,Move}	Editing text and adjusting text size	2	14	4.6
I_9	{Free Transform,Paste,Move}	Scaling and rotating something during copy-pasting	2	19	4.6
I_{10}	{Layer Order,Free Transform,Move}	Moving something and changing the ordering of layers	8	5	4.5
I_{11}	{Free Transform,Open,Drag Layer}	Positioning something via Drag Layer & making a movement to scene graph	8	18	4.5
I_{12}	{New,Open,Paste}	Creating a new file from an existing one via copy-pasting	8	21	4.5
I_{13}	{Free Transform,Edit Type Layer,Nudge}	Editing text and adjusting text size	8	23	4.4
I_{14}	{New,New Document:Custom,Paste}	Pasting stuff into a new file from a file belonging to a different application	8	24	4.3
I_{15}	{Free Transform,Move,Delete Layer}	Trying to fix a mistake; not an actual task as the goal is not by choice	8	7	3.6
I_{16}	{Rectangular Marquee,Free Transform,Deselect}	Doing a pointless thing; not a real task	17	9	3.5
I_{17}	{Crop,New,Open,New Document:Custom}	Creating a cropped version of an image while preserving the original one	8	26	3.5
I_{18}	{Free Transform,Open,Nudge}	Small tweak to put something in a layer	22	11	3.4

Table 6. User provided description of itemsets (potential frequent user tasks) with (i) rank based on our ranking (R_1); (ii) rank based on state of the art ranking (R_2); (iii) average user rating ($\overline{U}$). The itemsets with $rank >= 16$ in our ranking are highlighted in gray.

cohesion may reveal useful structural patterns for understanding relationships among temporal events. Such patterns can serve as a special class of behavioral motifs.

Implications for Clickstream Analysis and Visualization

Clickstream data is a valuable source in understanding user behavior. Researchers have proposed a variety of techniques that can be used for analyzing and visualizing [8, 13, 23] clickstream data. More recently, researchers have integrated visual exploration with analytics to develop visual analytics system for clickstreams [9, 10, 20, 21]. Visual analytics system for temporal event sequence data [11, 16] can also be used for clickstreams.

User task identification is a useful exercise in clickstream analysis and visualization. In particular, converting click sequences into coarse grained task sequences can be extremely useful. For example, clustering users based on task sequences has two benefits over clustering based on click sequences. First, tasks are semantically meaningful and consequently, the clusters are easily interpretable. Second, tasks eliminate noise (e.g., mistakes or unintended actions) and consequently, the clusters are robust. Click sequence to task sequence conversion can also be useful in clickstream visualization. In particular, visualizing task sequence instead of click sequence can address the challenge of dealing with long clickstreams that are hard to visualize.

DISCUSSION

In this section, we discuss different dimensions of our proposed pattern ranking technique.

Dropping Assumptions

The four assumptions that we report form the basis of our approach. Yet, one or more of these assumptions may not be applicable for certain applications.

For example, some applications may require extracting tasks with defined operation ordering, which implies dropping assumption $\mathbf{A_1}$. Notice that our definition of outlier based minimum occurrence window and accordingly the formulation of membership based cohesion is applicable to both itemsets and sequential patterns. Thus, our pattern ranking technique can extract frequent tasks in presence of strict ordering constraint. In recent times, the concept of soft pattern has emerged in many application domains. In contrast to the strict ordering constraint of sequential patterns, soft patterns have flexible ordering constraint. Our method can be applied with soft patterns to uncover the partial ordering of operations within tasks.

If we drop assumption $\mathbf{A_2}$, state-of-the-art cohesive itemset mining can reveal frequent user tasks. Notice that cohesive itemsets satisfy all assumptions, except $\mathbf{A_2}$.

Dropping assumption $\mathbf{A_3}$ will ease the frequent user task identification problem.

Finally, dropping assumption $\mathbf{A_4}$ will harden the frequent user task identification problem. Under this setting, all frequent itemsets or sequential patterns are likely to be tasks.

Threshold Θ

The value of support threshold Θ affects the result of frequent pattern mining and consequently, pattern ranking. We applied frequent pattern mining on our log datatset with different support thresholds such as 1%, 2%, 5% and 10%. In addition, we ranked the resultant patterns using membership based cohesion. We find that, with lower support threshold, we get more fine grained potential tasks (operation-groups).

Semantic Factors

As the frequent pattern mining techniques are syntactical, they ignore the semantic issues such as task equivalence. For example, in Table 6, I_8 and I_{13} represent the same logical task. However, it is not possible for a syntax based technique to capture such semantic similarity.

Containment

Containment is a central concept in frequent pattern mining. Notice that, if a pattern is frequent, each of its subpatterns is frequent as well. To address this issue, the idea of closed and maximal frequent patterns came into play. A *closed frequent pattern* is a frequent pattern that includes as many events as possible without compromising support. The idea of maximal frequent pattern is even stricter: a *maximal frequent pattern* is a frequent pattern that is not contained within another frequent pattern. Both frequent itemsets and sequential patterns have corresponding closed and maximal class. It is interesting that we can not use closed or maximal patterns for identifying frequent tasks. These pattern classes eliminate candidates based on containment, which is not a relevant criterion in our problem formulation. As a result, using these pattern classes escorts the risk of eliminating potential tasks from the ranking pool.

Interpretability

There are different types of high-level structures that we can extract from event sequences. From a human centred perspective, *interpretability* is a key factor in deciding which type to use. While machine learning based models are effective for prediction, such models are often difficult for humans to understand. In contrast, frequent patterns offer easy to understand patterns.

CONCLUSION

In this paper, we propose a novel frequent pattern ranking technique to extract frequent user tasks from application logs. Our technique is based on *membership based cohesion*, which prioritizes the patterns whose events appear contagiously in the supporting sequences with no or few outliers. We apply our technique on a real-world log dataset and conduct a user study to evaluate its effectiveness. Our experimental study shows that our technique outperforms state of the art for a variety of standard metrics such as NDCG and P/R@k.

ACKNOWLEDGEMENTS

We thank Matthew Hoffman, Hidy Kong, Stephen Nielson, Manoj Ravi, and John Thompson for their valuable feedback on this project. We also thank our user study participants for their time and feedback.

REFERENCES

1. Eytan Adar, Jaime Teevan, and Susan T. Dumais. 2008. Large Scale Analysis of Web Revisitation Patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 1197–1206. DOI: http://dx.doi.org/10.1145/1357054.1357241
2. Charu C. Aggarwal and Jiawei Han. 2014. *Frequent Pattern Mining*. Springer Publishing Company, Incorporated.
3. Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining Association Rules Between Sets of Items in Large Databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD '93)*. ACM, New York, NY, USA, 207–216. DOI:http://dx.doi.org/10.1145/170035.170072
4. Rakesh Agrawal and Ramakrishnan Srikant. 1995. Mining Sequential Patterns. In *Proceedings of the Eleventh International Conference on Data Engineering (ICDE '95)*. IEEE Computer Society, Washington, DC, USA, 3–14. http://dl.acm.org/citation.cfm?id=645480.655281
5. Boris Cuke, Bart Goethals, and Celine Robardet. *A new constraint for mining sets in sequences*. 317–328. DOI: http://dx.doi.org/10.1137/1.9781611972795.28
6. R. Stuart Geiger and Aaron Halfaker. 2013. Using Edit Sessions to Measure Participation in Wikipedia. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work (CSCW '13)*. ACM, New York, NY, USA, 861–870. DOI: http://dx.doi.org/10.1145/2441776.2441873
7. Brian C. Keegan, Shakked Lev, and Ofer Arazy. 2016. Analyzing Organizational Routines in Online Knowledge Collaborations: A Case for Sequence Analysis in CSCW. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '16)*. ACM, New York, NY, USA, 1065–1079. DOI: http://dx.doi.org/10.1145/2818048.2819962
8. Joseph B Kruskal and James M Landwehr. 1983. Icicle plots: Better displays for hierarchical clustering. *The American Statistician* 37, 2 (1983), 162–168.
9. Heidi Lam, Daniel M. Russell, Diane Tang, and Tamara Munzner. 2007. Session Viewer: Visual Exploratory Analysis of Web Session Logs. In *Symposium on Visual Analytics Science and Technology (VAST)*. 147–154.

10. Zhicheng Liu, Yang Wang, Mira Dontcheva, Matt Hoffman, Seth Walker, and Alan Wilson. 2017. Patterns and Sequences: Interactive Exploration of Clickstreams to Understand Common Visitor Paths. *IEEE Transactions on Visualization and Computer Graphics* 23, 01 (Janaury 2017).
11. Sana Malik, Fan Du, Megan Monroe, Eberechukwu Onukwugha, Catherine Plaisant, and Ben Shneiderman. 2015. Cohort Comparison of Event Sequences with Balanced Integration of Visual Analytics and Statistics. In *Proceedings of the 20th International Conference on Intelligent User Interfaces (IUI '15)*. ACM, New York, NY, USA, 38–49. DOI: http://dx.doi.org/10.1145/2678025.2701407
12. Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. 1997. Discovery of Frequent Episodes in Event Sequences. *Data Min. Knowl. Discov.* 1, 3 (Jan. 1997), 259–289. DOI: http://dx.doi.org/10.1023/A:1009748302351
13. Megan Monroe, Rongjian Lan, Hanseung Lee, Catherine Plaisant, and Ben Shneiderman. 2013. Temporal event sequence simplification. *Visualization and Computer Graphics, IEEE Transactions on* 19, 12 (2013), 2227–2236.
14. Hartmut Obendorf, Harald Weinreich, Eelco Herder, and Matthias Mayer. 2007. Web Page Revisitation Revisited: Implications of a Long-term Click-stream Study of Browser Usage. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 597–606. DOI: http://dx.doi.org/10.1145/1240624.1240719
15. Jaimie Y. Park, Neil O'Hare, Rossano Schifanella, Alejandro Jaimes, and Chin-Wan Chung. 2015. A Large-Scale Study of User Image Search Behavior on the Web. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 985–994. DOI: http://dx.doi.org/10.1145/2702123.2702527
16. Adam Perer and Fei Wang. 2014. Frequence: Interactive Mining and Visualization of Temporal Frequent Event Sequences. In *Proceedings of the 19th International Conference on Intelligent User Interfaces (IUI '14)*. ACM, New York, NY, USA, 153–162. DOI: http://dx.doi.org/10.1145/2557500.2557508
17. Huamin Qu and Qing Chen. 2015. Visual Analytics for MOOC Data. *IEEE Computer Graphics and Applications* 35, 6 (Nov 2015), 69–75. DOI: http://dx.doi.org/10.1109/MCG.2015.137
18. Jeffrey M. Rzeszotarski and Aniket Kittur. 2011. Instrumenting the Crowd: Using Implicit Behavioral Measures to Predict Task Performance. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 13–22. DOI: http://dx.doi.org/10.1145/2047196.2047199
19. Gang Wang, Tristan Konolige, Christo Wilson, Xiao Wang, Haitao Zheng, and Ben Y. Zhao. 2013. You Are How You Click: Clickstream Analysis for Sybil Detection. In *Proceedings of the 22Nd USENIX Conference on Security (SEC'13)*. USENIX Association, Berkeley, CA, USA, 241–256. http://dl.acm.org/citation.cfm?id=2534766.2534788
20. Gang Wang, Xinyi Zhang, Shiliang Tang, Haitao Zheng, and Ben Y. Zhao. 2016. Unsupervised Clickstream Clustering for User Behavior Analysis. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 225–236. DOI: http://dx.doi.org/10.1145/2858036.2858107
21. Jishang Wei, Zeqian Shen, Neel Sundaresan, and Kwan-Liu Ma. 2012. Visual cluster exploration of web clickstream data. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*. IEEE, 3–12.
22. Wikipedia. 2017. Ranking — Wikipedia, The Free Encyclopedia. (2017). https://en.wikipedia.org/wiki/Ranking [Online; accessed 9-January-2017].
23. Jian Zhao, Zhicheng Liu, Mira Dontcheva, Aaron Hertzmann, and Alan Wilson. 2015. MatrixWave: Visual Comparison of Event Sequence Data. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 259–268. DOI: http://dx.doi.org/10.1145/2702123.2702419