Mining Persistent Activity in Continually Evolving Networks

Caleb Belth
University of Michigan
cbelth@umich.edu

Xinyi Zheng
University of Michigan
zxycarol@umich.edu

Danai Koutra
University of Michigan
dkoutra@umich.edu

ABSTRACT

Frequent pattern mining is a key area of study that gives insights into the structure and dynamics of evolving networks, such as social or road networks. However, not only does a network evolve, but often the *way* that it evolves, itself evolves. Thus, knowing, in addition to patterns' frequencies, for how long and how regularly they have occurred—i.e., their *persistence*—can add to our understanding of evolving networks. In this work, we propose the problem of mining activity that persists through time in continually evolving networks—i.e., activity that repeatedly and consistently occurs. We extend the notion of temporal motifs to capture activity among *specific* nodes, in what we call *activity snippets*, which are small sequences of edge-updates that reoccur. We propose axioms and properties that a measure of persistence should satisfy, and develop such a persistence measure. We also propose PENminer, an efficient framework for mining activity snippets' *Persistence in Evolving Networks*, and design both offline and streaming algorithms. We apply PENminer to numerous real, large-scale evolving networks and edge streams, and find activity that is surprisingly regular over a long period of time, but too infrequent to be discovered by aggregate count alone, and bursts of activity exposed by their lack of persistence. Our findings with PENminer include neighborhoods in NYC where taxi traffic persisted through Hurricane Sandy, the opening of new bike-stations, characteristics of social network users, and more. Moreover, we use PENminer towards identifying anomalies in multiple networks, outperforming baselines at identifying subtle anomalies by 9.8-48% in AUC.

ACM Reference Format:
Caleb Belth, Xinyi Zheng, and Danai Koutra. 2020. Mining Persistent Activity in Continually Evolving Networks. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3394486.3403136

1 INTRODUCTION

Many networks evolve continually over time, such as traffic networks that encode current en-route traffic, networks of computers (IP-addresses) sending messages to each other, social networks of users interacting over time, and more. In order to understand these networks and the systems they represent, it is important to consider not just their structure, but also their temporal dynamics. Towards this goal, existing works have focused on counting patterns in networks. These patterns include temporal motifs (small subgraph patterns) [14, 17, 21, 24], frequent subgraphs in evolving networks [3, 4, 6], flow motifs [16], communication motifs [30], and coevolving relational motifs [5]. This line of research uses the frequency of patterns towards understanding the behavior of networks (e.g., some interaction patterns are more common in Q&A forums than in instant messengers [24]). However, many patterns in evolving networks may only last for a short period of time (e.g., bursty activity). This can lead to large aggregate counts, making a bursty anomaly falsely appear to be an intrinsic characteristic of a dynamic network. On the other hand, there may be important activity in a network that has low overall frequency, but that occurs continually and regularly [27], such as a stealthy computer-network attack. Thus, to more fully understand the dynamics of evolving networks, a pattern's *persistence*—involving how long it has occurred, how uniformly its occurrences are spread out, *and* what its frequency is—should be taken into account (Fig. 1).

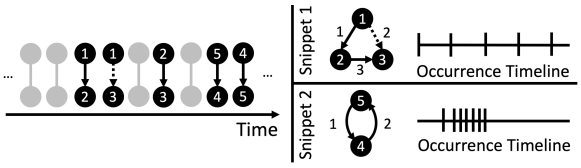

Figure 1: We seek to measure the persistence of activity snippets (i.e., reoccurring sequences of edge-updates) in continually evolving networks, which allows snippets 1 and 2 to be differentiated. While both snippets have approximately the same frequency, snippet 1 is more persistent, and snippet 2 is bursty. Activity snippets differ from temporal motifs by allowing different nodes interacting in the same way to be treated as distinct snippets (e.g., depending on the application, if snippet 2 reoccurs with nodes 6 and 7, it can either be treated as an instance of the same snippet, or of a new snippet).

In this work, we seek to mine persistent activity in continually evolving networks. We view the activity in a network as a stream of edge *updates* (insertions or deletions) over time. We introduce activity snippets by extending the idea of temporal motifs [24], which encode activity among nodes in *general*, to allow for encoding activity between or among *specific* nodes. This extension allows us to capture exact activity, (e.g., a specific edge in the network), which is important for applications like identifying which public transportation routes are used persistently, or identifying suspicious nodes or edges (e.g., nodes 4 and 5 in Fig. 1). In other applications, such as social network analysis, we may be interested in interactions between nodes in general, with no specific users in mind. In this case, activity snippets reduce to temporal motifs. In either case, activity can *repeat*, whether it be specific nodes repeatedly engaging, or different sets of nodes behaving in consistent ways. We seek to analyze this activity in terms of how persistently it occurs.

Our main contributions are as follows:

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403136

- **Precise Formulation of Persistence:** We introduce four axioms and three general properties that a measure of persistence ought to satisfy. We also provide a versatile persistence measure, and prove that it satisfies all axioms and properties.
- **Offline and Online Algorithms:** We develop PENminer, an efficient framework that uses our persistence measure to mine evolving networks. Our offline version, oPENminer, uses the measure for analyzing time-stamped sequences of edges from the past. Our online version, sPENminer, computes the measure incrementally for real-time analysis of edge-streams.
- **Extensive Experiments on Real Data:** We perform experiments on real, large scale evolving networks, which reveal real-world phenomena, including neighborhoods in NYC where taxi traffic persisted through Hurricane Sandy, the opening of new bike-stations in multiple cities, characteristics of social network users, and more. We also demonstrate that PENminer is scalable in offline and streaming settings, processing edge-updates in each stream 10K to 360K times faster than the rate of that stream. This allows both subtle and bursty activity to be identified in real time, right when it happens. PENminer also effectively identifies subtle anomalies, outperforming baselines by 9.8-48% in AUC.

Our code is available at https://github.com/GemsLab/PENminer.

2 RELATED WORK

Our work is related to motif mining, frequent subgraph mining, and persistent item mining.

Motif Mining. Network motifs, which are small, frequent subgraph patterns [23] are used in various problems, such as network summarization [22] and exploration, and dense subgraph detection [12]. Temporal motifs [17] extend static motifs to evolving networks by adding an ordering to their edges. Researchers have analyzed temporal motifs within and across a wide-range of networks [24], and have proposed sampling methods to estimate their counts [21]. Temporal motifs have also been extended to capture the flow of information among nodes in a motif [16]. Among different motifs, triangle counting has attracted significant interest [28].

A special case, communication motifs, was introduced to model temporal communication patterns of users in communication networks, for the purpose of understanding information flow [14, 30]. In [30], focusing on motif counts, the authors observed stability in the ranking of the ten most frequent motifs over time, but did not investigate subtly persistent or bursty behavior. Coevolving Relational Motifs (CRMs) are patterns that describe sets of nodes that evolve together in a consistent way [5]. In CRMs, consistency means that when a set of nodes co-evolve, it is almost always in the way specified by the motif. Thus, it considers the *relative* frequency of the motif, compared to other possible motifs over the same nodes, but does not investigate persistence.

Our work differs from motif-mining in its focus on *persistence* not just counts, and much of our analysis deals with *specific* edges or sequences of edges, rather than general activity among nodes.

Frequent Subgraph Mining. Frequent subgraph mining (FSM) has two settings: transactional FSM and single graph FSM [15]. The transactional setting attempts to identify, in a sequence of graphs (transactions), subgraphs that appear in a large number of the graphs. This setting naturally extends to evolving networks, by treating graph snapshots (corresponding to a batch of edge updates) as transactions [25]. The single graph setting seeks to identify subgraphs that have many instances within a single, large graph [10]. It has also been extended to evolving networks, including edge streams [3, 4, 6], where the goal is to adaptively *maintain* the most frequent subgraphs. While these methods seek to find *frequent subgraphs*, we seek to find *persistent activity*.

Table 1: Description of major symbols.

Notation	Description
$\mathscr{S}$	Edge stream
x	Activity snippet
$k_{\max}$, $\delta_{\max}$	Maximum size and duration for a snippet
ϕ	View of a snippet (ID, Label, or Order)
$[t_i, t_j]$, $\|[t_i, t_j]\|$	Interval of time and its width $t_j - t_i$
O_x, $\tilde{O}_x$	All and unique occurrences of x in $[t_s, t_e]$, resp.
Γ_x	Gaps (lengths of time) between unique occurrences of x

Persistent Item Mining. Motivated by scenarios of *stealthy* click-fraud or distributed denial-of-service attacks, persistent item mining in data streams was introduced in [9]. Arguing that, besides frequency, the length of the time period in which an item appears is important for understanding the dynamics of streaming data, the authors introduced a heuristic definition of a persistent item as one that occurs at least once in a large number of equally-sized observation periods. As we show, this simple definition violates some intuitive desired properties (§ 3.4). Our work goes beyond heuristics to establish a technical definition of a persistence measure, and focuses on edge-streams.

Persistent community detection, which attempts to find communities that last for long durations of time, has also been studied [19, 26]. However, these works seek to find tightly-knit subgraphs (communities) that last for a long time, while we focus on measuring *activity* that regularly *re-occurs* through time without requiring that this activity take place in contiguous chunks of time.

3 THEORY

3.1 Preliminary Definitions

We begin with preliminary definitions. Since most definitions apply beyond network settings, we first introduce the general concepts in § 3.1.1 and then the network-specific terminology in § 3.1.2.

3.1.1 Events in Time. **Interval of Time.** An interval of time $[t_s, t_e]$ (Fig. 2) is defined as $\{t : t \in \mathbb{R}_{\geq 0}, t \in [t_s, t_e]\}$. The width of the interval is $\|[t_s, t_e]\| = t_e - t_s \geq 0$. The set of all intervals is $\mathcal{I} \triangleq \{[t_s, t_e] : t_s, t_e \in \mathbb{R}_{\geq 0}, t_s \leq t_e\}$.

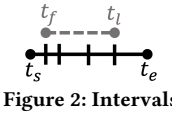

Figure 2: Intervals.

Event. An event x, with respect to an interval $[t_s, t_e]$, is something that occurs at least once in that interval. Examples include item purchases in a sequence of transactions, and measurements in a time series. The universe of events is $\mathcal{X}$.

Event Occurrence. An occurrence, $t \in [t_s, t_e]$ of x, is a timestamp. All the occurrences of x, $O_x = \{t_f, \ldots, t_l\}$, form an ordered multiset of timestamps between the first and last occurrrences, and the corresponding *interval of occurrences* is $[t_f, t_l]_x \subseteq [t_s, t_e]$ (e.g., Fig. 2). We denote the ordered set of *unique* occurrences $\tilde{O}_x$.

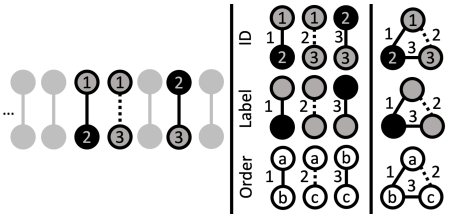

Figure 3: Left: An update sequence that is ordered but not contiguous. Right: The activity snippet for the sequence, depending on the view ϕ (ID, Label, or Order), and a graph describing the activity. Solid/dashed edges capture types of interactions (e.g., edge types or insertions vs. deletions), and the node colors denote labels.

Occurrence Gaps. The gaps (i.e., amount of time) between occurrences of x form the sequence $\Gamma_x = (g_1, \ldots, g_{|\tilde{O}_x|-1})$, where $g_i = t_{i+1} - t_i$ for $t_i, t_{i+1} \in \tilde{O}_x$. The number of gaps is $|\Gamma_x| = |\tilde{O}_x| - 1$.

3.1.2 Activity Snippets in Evolving Networks. **Graph or Network.** A graph or network $G = (V, E)$ is a set of nodes V, and a set of edges $E \subseteq V \times V$. If an edge between nodes v_1 and v_2 has a relationship type r, we denote it (v_1, r, v_2). Our proposed persistence measure and algorithms apply to general networks: labeled (nodes/edges), directed, weighted, bipartite or multi-graphs.

Edge-update. An edge update $\underline{u} = (\pm, v_1, r, v_2, t)$ to a network G is the *insertion* (+) or *deletion* (-) of an edge (v_1, r, v_2) at time t. We refer to $\underline{u}$'s timestamp with $\textsc{timestamp}(\underline{u})$.

Edge Stream. An edge stream $\mathscr{S} = (\underline{u}_1, \underline{u}_2, \ldots)$ is a time-ordered sequence of possibly infinite edge-updates to an evolving graph $\mathcal{G} = (G_1, G_2, \ldots)$. We denote the start time of $\mathscr{S}$, $\textsc{start}(\mathscr{S}) = \textsc{timestamp}(\underline{u}_1)$, its length $|\mathscr{S}|$, and if bounded, its end time $\textsc{end}(\mathscr{S}) = \textsc{timestamp}(\underline{u}_{|\mathscr{S}|})$. We call a sub-sequence of $\mathscr{S}$ that consists of all edge-updates in the last w time units a **window** $\mathscr{W}$ of width w.

Activity Snippet. Intuitively, an activity snippet x describes a sequence of activity among connected nodes in the network. Specifically, $x = (\phi(\underline{u}_i), \ldots, \phi(\underline{u}_j))$ is a sequence of ordered, but *not*-necessarily contiguous, edge updates, where the node IDs *may* be replaced by a **view** ϕ: (1) their labels or (2) the position of their first occurrence in the sequence (Fig. 3). If the node IDs are *not* replaced, the snippet is an exact sequence of activity. If the IDs are replaced by their position in the sequence, then the snippet is analogous to temporal motifs [24], but also captures edge deletions or insertions. The snippet has duration $\delta = \textsc{timestamp}(\underline{u}_j) - \textsc{timestamp}(\underline{u}_i)$ and size k equal to the number of edge-updates in the snippet. A $(\delta_{\max}, k_{\max})$ activity snippet is an activity snippet with duration $\delta \leq \delta_{\max}$ and size $k \leq k_{\max}$. In this work, events (§ 3.1.1) are activity snippets.

Problem Definition. Given these definitions, we focus on the problem of persistent activity mining:

Problem 1 (Persistent Activity Mining). *Given a network G that continually evolves via an edge-stream $\mathscr{S}$, measure the persistence of each activity snippet x, i.e., for how long, how often, and how regularly it has occurred.*

3.2 Properties of Persistence

Based on our definitions, we present axioms and properties that a persistence measure should follow. In the remaining sections, we propose a principled persistence measure (§ 3.3) and prove that it satisfies all the axioms and properties (§ 3.4). Although **our theoretical definitions, properties, proposed measure and derivations are general and apply to any event x in a stream or time series**, in the context of Problem 1, x is an activity snippet.

Definition 1 (Persistence Measure). *A persistence measure $P(x; [t_s, t_e]) : \mathcal{X} \times \mathcal{I} \to \mathbb{R}_{\geq 0}$ is a function that defines the persistence of an event $x \in \mathcal{X}$ in the interval $[t_s, t_e] \in \mathcal{I}$, and satisfies the following axioms:*

- ***A1**: It is non-negative, and 0 iff there are no occurrences. Formally, $P(x; [t_s, t_e]) \geq 0$, with equality iff $|O_x| = 0$.*
- ***A2**: As the interval becomes infinitely filled with unique occurrences, persistence tends to infinity. That is, $\lim_{|\tilde{O}_x| \to \infty} P(x; [t_s, t_e]) = \infty$.*
- ***A3**: Shifting all occurrences in time does not affect persistence. Formally, $P(x'; [t_s, t_e]) = P(x; [t_s, t_e])$, where x' is an event with occurrences $O_{x'} = \{t + c : t \in O_x\}$, for some constant $c \in \mathbb{R}$ such that the shifted points remain in $[t_s, t_e]$, i.e, $t_f + c \geq t_s, t_l + c \leq t_e$.*
- ***A4**: Shrinking the interval of measurement towards $[t_f, t_l]_x$ leads to higher persistence. Mathematically, $P(x; [t_s, t_e]) \leq P(x; [t'_s, t'_e]) \leq P(x; [t_f, t_l]_x)$, for $t_s \leq t'_s \leq t_f$ and $t_e \geq t'_e \geq t_l$. The equality holds for $t_s = t'_s = t_f$ and $t_e = t'_e = t_l$.*

Besides these axioms, there are several properties that a good persistence measures ought to follow. For use in presenting these properties, let $\tilde{x}^n$ be a special class of event with n occurrences, all of which are unique and uniformly spaced out over $[t_f, t_l]_{\tilde{x}^n}$, that is, $|O_{\tilde{x}^n}| = |\tilde{O}_{\tilde{x}^n}| = n$ and $g_i = g_j = \frac{|[t_f, t_l]_{\tilde{x}^n}|}{|\Gamma_{\tilde{x}^n}|}, \forall g_i, g_j \in \Gamma_{\tilde{x}^n}$.

- **P1**: For two events with n unique, uniformly-spaced occurrences, persistence is larger for the event with occurrences spread over a wider interval. Formally, for any $\tilde{x}^n_1$ and $\tilde{x}^n_2$ such that $|[t_f, t_l]_{\tilde{x}^n_1}| < |[t_f, t_l]_{\tilde{x}^n_2}|$, it should hold that $P(\tilde{x}^n_1; [t_s, t_e]) < P(\tilde{x}^n_2; [t_s, t_e])$.
- **P2**: For two events with unique, uniformly-spaced occurrences spread out over the same interval, persistence is larger for the event with more occurrences. Mathematically, for any $\tilde{x}^n$ and $\tilde{x}^{n+k}$ such that $[t_f, t_l]_{\tilde{x}^n} = [t_f, t_l]_{\tilde{x}^{n+k}}$, the persistence measure should satisfy $P(\tilde{x}^n; [t_s, t_e]) < P(\tilde{x}^{n+k}; [t_s, t_e]), \forall k \geq 1$.
- **P3**: The persistence of an event with n unique occurrences in an interval is maximized iff the occurrences are spread out uniformly. Formally, for any x and $\tilde{x}^n$ such that $|O_x| = |\tilde{O}_x| = |\tilde{O}_{\tilde{x}^n}| = n$ and $[t_f, t_l]_x = [t_f, t_l]_{\tilde{x}^n}$, the persistence measure should satisfy $P(x; [t_s, t_e]) \leq P(\tilde{x}^n; [t_s, t_e])$, with equality iff $\Gamma_x = \Gamma_{\tilde{x}^n}, \forall n > 2$.

3.3 Proposed Persistence Measure

Family of Persistence Measures. The axioms and properties point to three main components of persistence: the width of the interval in which occurrences fall, the frequency, and the distribution of occurrences. We thus propose a family of persistence measures

$$P(x; [t_s, t_e]) \triangleq f\Big(\underbrace{W(x; [t_s, t_e])}_{\text{width}}, \underbrace{F(x; [t_s, t_e])}_{\text{frequency}}, \underbrace{S(x; [t_s, t_e])}_{\text{spread}} \Big), \quad (1)$$

where $W(\cdot)$ is a function of the width of x's interval of occurrences $[t_f, t_l]_x$, $F(\cdot)$ is a function of the number of occurrences, $S(\cdot)$ is a function of how uniformly the points are spread out over $[t_f, t_l]_x$, and $f(\cdot)$ is a function that combines these components.

A Persistence Measure. There are a number of ways to construct the functions $W(\cdot)$, $F(\cdot)$, and $S(\cdot)$, and to combine them. We propose one intuitive persistence measure within the family described above and show that it satisfies all the axioms and properties (§ 3.4).

First, we define $W(\cdot)$ as the percentage of the interval width that is covered by the occurrences of x:

$$W\big(x;[t_s,t_e]\big) \triangleq \frac{|[t_f,t_l]_x|+1}{|[t_s,t_e]|+1}, \tag{2}$$

where we add one in the numerator and denominator so that they are non-0 when $t_f = t_l$ or $t_s = t_e$. For $|O_x| = 0$, we define $W(\cdot) = 0$.

Second, we define $F(\cdot)$, the function of frequency, as the logarithm of the number of occurrences $|O_x|$ to: (a) prevent the term from dominating over the others (which can be more than an order of mangitude smaller), and (b) naturally capture diminishing returns. Formally,

$$F\big(x;[t_s,t_e]\big) \triangleq \log_{10}\Big(|O_x|+1\Big), \tag{3}$$

where we add one to ensure that the logarithm is well-defined in the absence of occurrences.

Third, for $S(\cdot)$, to capture how regularly the occurrences are spread, we model the distribution of the *gaps* $\Gamma_x = (g_1,\ldots,g_{|\tilde{O}_x|-1})$ in a principled way via Shannon entropy [8]. Entropy measures the amount of randomness in a distribution, and when the distribution has a fixed number of outcomes in its support, this essentially means it measures the distribution's uniformity. The gaps Γ_x between occurrences (§ 3.1.1) normalized by the interval width $|[t_f,t_l]_x|$ define a probability mass function with entropy

$$H(\Gamma_x) \triangleq -\textstyle\sum_{g_i\in\Gamma_x} \frac{g_i}{|[t_f,t_l]_x|} \log \frac{g_i}{|[t_f,t_l]_x|}. \tag{4}$$

As is standard in information theory, we define $0\log 0 = 0$ (e.g., multiple occurrences at the same time giving $g_i = 0$), and we use log base 2. In order to remove the dependency on the number of occurrences (since this is captured by F) and make this a measure of spread, we normalize by the maximum entropy $\log(|\Gamma_x|)$ [8]:

$$S\big(x;[t_s,t_e]\big) \triangleq \begin{cases} \frac{H(\Gamma_x)}{\log|\Gamma_x|}+1, & |\Gamma_x|>1 \\ 1, & |\Gamma_x|\in\{0,1\} \end{cases}, \tag{5}$$

where we add one since entropy is 0 if there are 0 or 1 gaps.

Since the terms defined above have different units, we combine them into one function as follows:

$$P\big(x;[t_s,t_e]\big) \triangleq W\big(x;[t_s,t_e]\big)^{\alpha} F\big(x;[t_s,t_e]\big)^{\beta} S\big(x;[t_s,t_e]\big)^{\gamma}, \tag{6}$$

where the finite exponents $\alpha,\beta,\gamma\in(0,\infty)$ can be used for assigning different weights to the components, depending on the goals of a particular application (e.g., $\gamma>1$ can help events with low frequency but high regularity to be discovered).

3.4 Theoretical Analysis

We now show that Eq. (6) is a principled persistence measure that satisfies all axioms and properties. We also discuss a recently-proposed persistence heuristic, which violates key axioms.

Lemma 1. *If x has occurred at least once, $|O_x|>0$, then our proposed measure is positive: $P\big(x;[t_s,t_e]\big)>0, \forall\,\alpha,\beta,\gamma\in(0,\infty)$.*

Proof. $W\big(x;[t_s,t_e]\big)>0$ since $|[t_s,t_e]|+1 \geq |[t_f,t_l]_x|+1>0$. Since $|O_x|>0$, $F\big(x;[t_s,t_e]\big)>0$. If $|O_x|\in\{1,2\}$, then $|\Gamma_x|\in\{0,1\}$, so $S\big(x;[t_s,t_e]\big)=1>0$ by definition. If $|O_x|>2$, $\log|\Gamma_x|>0$ and the well-known property of entropy, $H(\Gamma_x)\geq 0$ [8], implies $S\big(x;[t_s,t_e]\big)>0$. For $\alpha,\beta,\gamma>0$, each component is still greater than 0 and so is their product $P\big(x;[t_s,t_e]\big)$. □

Theorem 1. *$P\big(x;[t_s,t_e]\big)$ as defined in Eq. (6) satisfies all the axioms, and thus is a persistence measure.*

Proof. We show that $P\big(x;[t_s,t_e]\big)$ is a persistence measure by proving that it satisfies Axioms **A1**-**A4**. Since $\alpha,\beta,\gamma\in(0,\infty)$ do not affect the results, we omit them from the proofs for readability.

• Axiom **A1.** If $|O_x|=0$, then $P\big(x;[t_s,t_e]\big)=0\cdot 0\cdot 1=0$. If $|O_x|>0$, then $P\big(x;[t_s,t_e]\big)>0$, by Lemma 1. Therefore, since $|O_x|\geq 0$, if $P\big(x;[t_s,t_e]\big)=0$, then $|O_x|=0$.

• Axiom **A2.** Since $|[t_f,t_l]_x|\leq|[t_s,t_e]|$, $W\big(x;[t_s,t_e]\big)\leq 1$. Also $H(\Gamma_x)\leq\log(|\Gamma_x|)$ [8], so $S\big(x;[t_s,t_e]\big)\leq 2$. Thus, these bounded terms do not affect the limit, and $\lim_{|\tilde{O}_x|\to\infty}\log_{10}(|O_x|+1)=\infty$. Thus, unaffected by $\alpha,\beta,\gamma\in(0,\infty)$, $\lim_{|\tilde{O}_x|\to\infty}P\big(x;[t_s,t_e]\big)=\infty$.

• Axiom **A3.** First, $W(x',[t_s,t_e])=\frac{t_l+c-t_f-c+1}{t_e-t_s+1}=\frac{t_l-t_f+1}{t_e-t_s+1}=W\big(x;[t_s,t_e]\big)$. Also, $|O_{x'}|=|O_x|$, so $F(x';[t_s,t_e])=F\big(x;[t_s,t_e]\big)$. Each new gap $g_i'=t_{i+1}+c-t_i-c=t_{i+1}-t_i=g_i$ (§ 3.1.1), so shifting the occurrences does not change their spread: $S(x';[t_s,t_e])=S\big(x;[t_s,t_e]\big)$. As a result, the persistence remains the same.

• Axiom **A4.** By definition, $t_s\leq t_s'\leq t_f$ and $t_e\geq t_e'\geq t_l$, so $t_l-t_f\leq t_e'-t_s'\leq t_e-t_s$. This implies that $W\big(x;[t_s,t_e]\big)\leq W(x;[t_s',t_e'])\leq W(x;[t_f,t_l]_x)$, with equality iff $t_f=t_s$ and $t_l=t_e$. Since all occurrences fall in $[t_f,t_l]_x$ by definition, the frequency and spread terms remain the same upon shrinking the measurement interval. Therefore, $P\big(x;[t_s,t_e]\big)\leq P(x;[t_s',t_e'])\leq P(x;[t_f,t_l]_x)$, with equality iff $t_f=t_s$ and $t_l=t_e$. □

Theorem 2. *Our persistence measure $P\big(x;[t_s,t_e]\big)$, Eq. (6), satisfies the three desired properties **P1**-**P3**.*

Proof. We prove each property separately:

• Property **P1.** First, since $|O_{\tilde{x}_1^n}|=|O_{\tilde{x}_2^n}|=n$, $F\big(\tilde{x}_1^n;[t_s,t_e]\big)=F\big(\tilde{x}_2^n;[t_s,t_e]\big)$. Also, for $|\Gamma_{\tilde{x}_1^n}|=|\Gamma_{\tilde{x}_2^n}|=n-1$ uniform gaps, $H(\Gamma_{\tilde{x}_1^n})=H(\Gamma_{\tilde{x}_1^n})=\log(n-1)$. Thus, $S\big(\tilde{x}_1^n;[t_s,t_e]\big)=S\big(\tilde{x}_2^n;[t_s,t_e]\big)$. Since $|[t_f,t_l]_{\tilde{x}_1^n}|<|[t_f,t_l]_{\tilde{x}_2^n}|$ and $|[t_s,t_e]|$ is fixed, $W\big(\tilde{x}_1^n;[t_s,t_e]\big)<W\big(\tilde{x}_2^n;|[t_s,t_e]|\big)$. Therefore, $P\big(\tilde{x}_1^n;[t_s,t_e]\big)<P\big(\tilde{x}_2^n;[t_s,t_e]\big)$.

• Property **P2.** We prove this by case. For $n=0$, $P(\tilde{x}^0;[t_s,t_e])=0<P(\tilde{x}^k;[t_s,t_e])$ by Axiom **A1**. For $n\geq 1$, $W(\tilde{x}^n;[t_s,t_e])=W(\tilde{x}^{n+k};[t_s,t_e])$, since $[t_f,t_l]_{\tilde{x}^n}=[t_f,t_l]_{\tilde{x}^{n+k}}$. If $n=1$, then $S(\tilde{x}^1;[t_s,t_e])=1\leq S(\tilde{x}^{1+k};[t_s,t_e])\leq 2$ and $F\big(\tilde{x}^1;[t_s,t_e]\big)=\log_{10}(2)<\log_{10}(2+k)=F\big(\tilde{x}^{1+k};[t_s,t_e]\big)$. For $n>1$, because of uniform gaps, $H(\Gamma_{\tilde{x}^n})=\log(|\Gamma_{\tilde{x}^n}|)$ and $H(\Gamma_{\tilde{x}^{n+k}})=\log(|\Gamma_{\tilde{x}^{n+k}}|)$ [8]. Thus, $S(\tilde{x}^n;[t_s,t_e])=S\big(\tilde{x}^{n+k};[t_s,t_e]\big)=2$. Lastly, $F\big(\tilde{x}^n;[t_s,t_e]\big)<F(\tilde{x}^{n+k};[t_s,t_e])$. By combining the three terms, we prove the claim.

• Property **P3.** Since $|O_x|=|O_{\tilde{x}^n}|$, $F\big(x;[t_s,t_e]\big)=F\big(\tilde{x}^n;[t_s,t_e]\big)$. Also, $[t_f,t_l]_x=[t_f,t_l]_{\tilde{x}^n}$, so $W\big(x;[t_s,t_e]\big)=W\big(\tilde{x}^n;[t_s,t_e]\big)$. Moreover, $H(\Gamma_x)\leq\log(|\Gamma_x|)$, with equality iff the $g_i\in\Gamma_x$'s are uniform. Thus, $S\big(x;[t_s,t_e]\big)\leq 2=S\big(\tilde{x}^n;[t_s,t_e]\big)$, with equality iff the gaps $g_i\in\Gamma_x$'s are uniform, in which case $\Gamma_x=\Gamma_{\tilde{x}_n}$. □

Persistence Heuristic in Data Streams [9]. A simple persistence heuristic was proposed in [9]: an item in a data stream is considered persistent if it occurs at least once in a large number of *predefined*, equally-sized observation periods (intervals). We show that this heuristic violates axioms **A2**-**A4** via counter-examples. As the number of unique occurrences tends to infinity, the heuristic tends to the number of intervals, not infinity (**A2** violation). If an item has two occurrences in the same interval, they can be shifted such that one falls in a new period, in which case the heuristic grows (**A3** violation). Finally, for an item with two occurrences in different intervals, after shrinking $[t_s, t_e]$ towards $[t_f, t_l]_x$ and re-dividing into the predefined number of intervals, they could still fall in different intervals. Thus persistence would not increase (**A4** violation).

4 PROPOSED METHOD: PENMINER

With our proposed persistence measure in Eq. (6), and the definitions in § 3.1, we can now define the offline and streaming (online) versions of our problem precisely.

Problem 2 (Offline/Streaming Persistent Activity Mining). *Given an edge stream, $\mathscr{S}$, a maximum duration $\delta_{\max}$, and a maximum snippet size $k_{\max}$,*

- [*Offline*] ***output*** *the persistence $P\big(x; [\textsc{start}(\mathscr{S}), \textsc{end}(\mathscr{S})]\big)$*
- [*Streaming*] ***maintain*** *the persistence $P(x; [\textsc{start}(\mathscr{S}), t])$*

of every $(\delta_{\max}, k_{\max})$-activity snippet x observed in the whole stream.

We next introduce our offline and streaming algorithms, oPENminer and sPENminer, to solve this problem. We first discuss extracting all $(\delta_{\max}, k_{\max})$-snippets from a stream $\mathscr{S}$, followed by the details of each variant. We give the main steps for PENminer, which takes as a parameter which variant to use, in Algorithm 1, and detailed pseudocode for reproducibility in § A.2.

Activity Snippet Extraction (line 2). Each time a new edge-update $\underline{u}_{\text{new}}$ arrives (at time t), the procedure ExtractNewSnippets is called. The procedure maintains the window $\mathscr{W}$, which consists of all updates $\underline{u}$ that occurred within the last $\delta_{\max}$ time units, and removes all others. It then adds the new update $\underline{u}_{\text{new}}$ (which is 0 time-units in the past). In addition to maintaining the window, the procedure extracts all valid snippets from the window. A valid snippet must be connected, have duration $\delta \le \delta_{\max}$, and size $k \le k_{\max}$. Since all stale updates have been removed from $\mathscr{W}$, $\delta_{\max}$ is already enforced. Any new snippet instance must contain $\underline{u}_{\text{new}}$. The singleton snippet containing just $\underline{u}_{\text{new}}$ is created, and then snippets of size $k = 2, \ldots, k_{\max}$ are constructed smallest to largest, such that the nodes in each snippet are connected.

Offline Algorithm (lines 4 and 8). For the offline version of Problem 2, oPENminer maintains the set of occurrences of each activity snippet extracted from the stream. Then, when the end of the stream is reached, it computes and outputs the persistence of each snippet x in $[\textsc{start}(\mathscr{S}), \textsc{end}(\mathscr{S})]$ with Eq. (6) as $P(x; [\textsc{start}(\mathscr{S}), \textsc{end}(\mathscr{S})])$.

Streaming Algorithm (line 6). For the online version of Problem 2, there are two cases to handle: (1) update the persistence of snippet x when it occurs at time t, and (2) return the correct persistence of a snippet x if it is queried at any other time t. We maintain in memory a constant amount of information on each snippet: the total number of its occurrences, $|O_x|$, the number of gaps between occurrences, $|\Gamma_x|$, the time of its first and last occurrences, t_f, t_l, and its persistence when it last occurred, $P(x; [\textsc{start}(\mathscr{S}), t_l])$.

Algorithm 1 PENminer ($\mathscr{S}$, $\delta_{\max}$, $k_{\max}$, ϕ, α, β, γ, variant)

Input: Stream $\mathscr{S}$, max snippet duration $\delta_{\max}$ and size $k_{\max}$, view ϕ, persistence exponents α, β, γ, the variant (oPENminer/sPENminer).

1: **while** $\underline{u}_{\text{new}} \in \mathscr{S}$ **do** ▷ While there is a new update in the stream
2: **for** $x \in$ ExtractNewSnippets($\mathscr{W}$, $\underline{u}_{\text{new}}$, t, ϕ) **do**
3: **if** variant is oPENminer **then**
4: Add the occurrence t of x to O_x.
5: **else**
6: Update $P(x; [\textsc{start}(\mathscr{S}); t])$ incrementally.
7: **if** variant is oPENminer **then**
8: Compute $P(x; [\textsc{start}(\mathscr{S}), \textsc{end}(\mathscr{S})])$ for each x observed.

• **Case 1. Updating Persistence Upon Occurrence.** Since there is a new occurrence at the current time t and we maintain the first occurrence of each snippet, we can compute the width function $W\big(x; [t_s, t_e]\big)^{\alpha} = \left(\frac{|[t_f, t]_x| + 1}{|[\textsc{start}(\mathscr{S}), t]| + 1}\right)^{\alpha}$ from Eq. (2). Since we maintain the number of occurrences, $|O_x|$, we can obtain the frequency $F\big(x; [\textsc{start}(\mathscr{S}); t]\big)^{\beta} = \log_{10}\left(|O_x| + 1\right)^{\beta}$ from Eq. (3). By maintaining the number of gaps between occurrences, we know whether $|\Gamma_x| \in \{0, 1\}$ (in Eq. (5)) and can compute $\log |\Gamma_x|$. In order to compute $H(\Gamma_x)$—Eq. (4)—we show how the entropy of the distribution induced by a snippet x's gap widths can be computed incrementally, as new occurrences create new gaps. Specifically, the entropy when a new gap is formed can be computed from (1) the previous entropy, (2) the previous normalizing constant Z, and (3) the new gap g_{n+1}. This is stated in Thm. 3 and proved via the following two Lemmas.

Let $p_i \triangleq \frac{g_i}{Z}$, where $Z = \sum_{i=1}^{n} g_i$, be a probability mass function on a set of n gaps, Γ_x, induced by normalizing each gap by Z.

Lemma 2. *The entropy $H(p) = \log Z - \frac{1}{Z}\sum_{i=1}^{n} g_i \log g_i$.*

Lemma 3. *For the new set of gaps $\Gamma'_x = \Gamma_x \cup \{g_{n+1}\}$, and normalizing constant $Z' = Z + g_{n+1}$, the corresponding pmf p' has entropy $H(p') = \frac{Z}{Z'}\log Z' - \frac{g_{n+1}}{Z'}\log\frac{g_{n+1}}{Z'} - \frac{1}{Z'}\sum_{i=1}^{n} g_i \log g_i$.*

Proof. Both lemmas can be derived by algebraically expanding the corresponding entropy definition and using $Z = \sum_{i=1}^{n} g_i$. □

Theorem 3. *$H(p') = H(p) + \frac{Z}{Z'}\log Z' - \log Z - \frac{g_{n+1}}{Z'}\log\frac{g_{n+1}}{Z'} + \left(\frac{1}{Z} - \frac{1}{Z'}\right)(\log Z - H(p))Z$, where $H(p)$ is the entropy of p, Z is the normalizing constant for p, and g_{n+1} is the new gap.*

Proof. Let $\Delta = H(p') - H(p)$ be the change in entropy and $X = \sum_{i=1}^{n} g_i \log g_i$. By Lemmas 2-3, we obtain $\Delta = \frac{Z}{Z'}\log Z' - \log Z - \frac{g_{n+1}}{Z'}\log\frac{g_{n+1}}{Z'} + \left(\frac{1}{Z} - \frac{1}{Z'}\right)X$. By Lemma 2, $X = (\log Z - H(p))Z$. Plugging, this into $H(p') = H(p) + \Delta$ completes the proof. □

The new gap g_{n+1} is the time from the previous occurrence of x (which we know since we maintain t_l) to the current time t. The normalizing constant Z is the time from the first occurrence (t_f) to the previous occurrence (t_l), both of which we maintain.

• **Case 2. Querying Persistence Without a New Occurrence.** Without a new occurrence, the frequency $F(x; [\textsc{start}(\mathscr{S}), t])^{\beta}$ and spread $S(x; [\textsc{start}(\mathscr{S}), t])^{\gamma}$ do not change. From Eq. (2), we compute the width function, $W(x; [\textsc{start}(\mathscr{S}), t])^{\alpha}$, by replacing $|[\textsc{start}(\mathscr{S}), t_l]|$ with $|[\textsc{start}(\mathscr{S}), t]|$ in the denominator.

5 EVALUATION

We investigate the following research questions across multiple real networks (§ 5.1):

- **RQ1** What does the relationship between frequency and persistence reveal about activity and networks?
- **RQ2** Can sPENminer find anomalies in real-time?
- **RQ3** Can sPENminer process updates to a network at least as quickly as they arrive, and how does oPENminer scale with the number of edge updates in the stream?

In § 5.2 we investigate a core value of measuring persistence, which is the relationship between frequency and persistence. Specifically, we find that activity snippets with high persistence, but low frequency tend to correspond to *subtle* yet regular activity, which would be missed by measuring frequency alone. On the other hand, snippets with high frequency, but low persistence tend to be *bursty*. In § 5.3, we show how we can make analogous insights in real-time with sPENminer, and **accurately** find both subtle and bursty anomalies right when they occur. Finally, in § 5.4, we evaluate the maximum duration $\delta_{\max}$ and maximum size $k_{\max}$ parameters, and demonstrate that sPENminer processes edges in each stream 10K to 360K times faster than the rate of that stream, and oPENminer scales linearly with the number of edge-updates processed.

5.1 Data

We utilize a diverse set of evolving networks (Tab. 2), including communication, transportation, computer, and social networks.

- The communication network Eu Email has timestamped edges denoting emails sent within a European research institute [18].
- Columbus Bike, Chicago Bike, and Boston Bike are networks encoding bike trips made in the bike-share systems in those three cities [1]. Each node is a bike station. When a bike leaves one station for another, an edge is inserted into the network; when the bike arrives at its destination, it is deleted. Thus, the networks capture *en route* bike-trips.
- Similarly, NYC Taxi [2] captures *en route* taxi trips in New York City, but in this case, nodes are city zones rather than stations.
- Edges in the social network Reddit correspond to timestamped references between subreddits (topical discussion boards) [18].
- DARPA IP [20] is an IP-IP network, where both normal network traffic and malicious attack traffic is present. Edges denote interactions between computers in the network.
- Stackoverflow [18] consists of interactions among users on the website Stackoverflow. The interactions are between users, and can be answers to questions, comments on questions, or comments on answers.

Table 2: Description of edge streams: number of edge-updates, number of nodes, whether it has edge deletions, number of unique edges, number of edge types, and average rate of the stream in updates/sec.

	$\lvert\mathscr{S}\rvert$	$\lvert V\rvert$	Del	$\lvert E\rvert$	E types	rate ($\underline{u}$/sec)
Eu Email	332,334	986	N	24,929	1	0.04
Columbus Bike	534,998	74	Y	2,951	1	0.02
Reddit	858,488	67,180	N	339,643	1	0.02
Darpa IP	4,554,344	25,525	N	68,910	1	78.5
Boston Bike	17,421,182	476	Y	81,508	1	0.05
Chicago Bike	33,331,104	712	N	171,651	1	0.25
Stackoverflow	63,497,050	2,601,977	N	36,233,450	3	0.27
NYC Taxi	3,077,990,404	265	Y	60,750	1	9.29

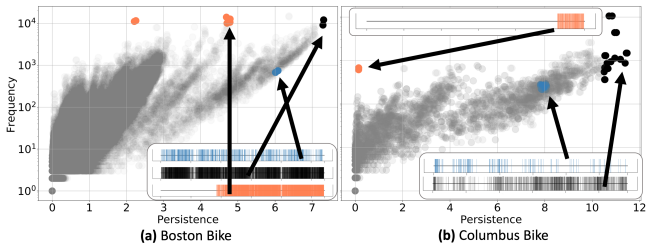

Figure 4: Boston (a) and Columbus (b) bike networks. Representative timelines from various parts of the PvF plots demonstrate how persistent, bursty, and subtle activity can be identified.

5.2 RQ1: Persistent vs. Frequent

The relationship between frequency and persistence allows for a more complete view of activity in networks. We show this via a Persistence vs. Frequency, or **PvF**, plot of activity snippets, with snippet frequency, $|O_x|$, in log-scale on the y-axis, and persistence $P(x; [\textsc{start}(\mathscr{S}), \textsc{end}(\mathscr{S})])$ on the x-axis. Activity snippets with unusually high persistence relative to their frequency fall towards the lower-right of the plot. Those with unusually high frequency relative to their persistence fall towards the upper-left. We call the former type of activity snippet **subtly persistent**, and the latter **bursty** (whether a single burst or periodic bursts). We analyze persistent, subtly persistent, and bursty activity snippets in several networks via PvF plots, and show what they correspond to in each context. To see what the baseline of frequency would give, the variation of points along the y-axis only can be studied.

5.2.1 Transportation Networks. **Setup.** We generate PvF plots for bike trips in Boston, MA, Columbus, OH, and Taxi trips in New York City (NYC) for the two weeks surrounding Hurricane Sandy (10/22/12-11/05/12). We use view ϕ = ID. Since once a trip is complete the rider leaves the bike/taxi, sequences of edges are not linked. Thus, we set $k_{\max} = 1$ (and $\delta_{\max} = 1$), so that each activity snippet corresponds to a single bike/taxi trip. Next, in social networks, we use $k_{\max} > 1$. We set α, β and γ for visual clarity, and discuss how in the supplement on reproducibility (§ A.3).

Results. Results for Boston Bike and Columbus Bike are in Fig. 4. The most persistent bike trips (black) are both frequent and persistent. A representative timeline from each, where each tick is an occurrence of the snippet, is shown in the lower-right of each plot. In Boston, the most persistent trip is from Massachusetts Ave. in front of MIT, up the street to outside the Central subway station. This is a reasonable route for commuters to bike from MIT to the subway. In both networks, bursts (orange) reveal new bike stations opening. The new station in Boston opened right in front of MIT and became immediately popular, which led to a large burst of activity. With the station now established, its persistence should grow over time. For comparison, we show a representative point from the middle of the Columbus PvF plot, and a trip in Boston that is more persistent than expected given its frequency (blue).

In NYC Taxi, the bursty and subtly persistent snippets (Taxi trips) reveal which neighborhoods in NYC were most affected by Hurricane Sandy (Fig. 5). The hurricane made landfall in NYC around 8pm EST on 10/29 (shown with a dashed line). The bursty anomalies (orange) all correspond to neighborhoods that were brought to a

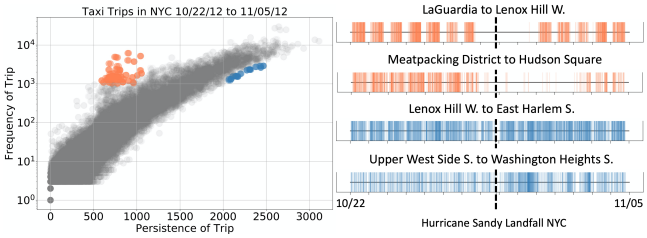

Figure 5: Taxi trips in NYC with high frequency but low persistence reveal neighborhoods that were brought to a standstill by Hurricane Sandy, while more persistent trips reveal those that were not.

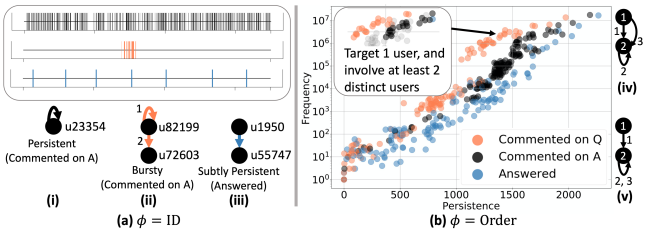

Figure 6: Stackoverflow Analysis. (a): ϕ = ID. Persistent snippets usually correspond to users commenting on their own answers. The bursty snippet reveals 36 back-and-forths between two users in 1hr. The subtly persistent snippet reveals user `u1950` regularly answering user `u55747`'s questions. (b): ϕ = Order. Discussions targeting one user's content and involving at least two users, occur with similar frequency for comments on Qs and As, but comments on Qs are burstier—a distinction not captured by frequency alone.

stand-still during the storm. For example, the first trip shown is from LaGuardia airport to the Lenox Hill neighborhood. Flights were canceled *en masse* at LaGuardia prior to the hurricane, interrupting trips even before the storm's landfall. The second timeline shows a trip from the Meatpacking District to Hudson Square. Both neighborhoods are on the coast of Manhattan Island, and experienced severe flooding. The decrease in trips prior to the storm is likely due to businesses closing in preparation. Many of the subtly persistent trips (blue) correspond to neighborhoods that were resilient despite the storm. The first example is Lenox Hill and East Harlem. While these neighborhoods border the East River, they extend inland, allowing plenty of roads for taxis to continue service on. Lenox Hill hospital was also the subject of a study of the clinical response to Hurricane Sandy [29]. Possibly, the continuity in taxi trips was due to visits to the hospital. Similarly, the Upper West Side extends into inner-Manhattan, and Washington Heights is a significant distance north of where the hurricane made landfall.

Takeaways on Transportation. We find that subtly persistent activity often reveals commutes, because of their high regularity. This has applications in city planning, since commute routes can be good options for introducing ride-share programs. We also find that bursty activity often reveals major changes in the real world, such as severe weather disturbances or new routes becoming possible.

5.2.2 Social Networks. **Setup.** We use the social network Stackoverflow, where users interact by answering technical questions, commenting on questions, and commenting on answers. We first analyze a 3-month interval. For this analysis, we use the view ϕ = ID, set $k_{\max} = 3$, and $\delta_{\max} = 1$hr. Secondly, we analyze the entire dataset using ϕ = Order, with $k_{\max} = 3$ and $\delta_{\max} = 15$min. In the latter scenario, the snippets we analyze are a superset of the 3-node, 3-edge temporal motifs studied in [24]. We plot, in orange, snippets involving users commenting on questions, in black, those involving users commenting on answers, and in blue, answering questions.

Results. Results are shown in Fig. 6. The timelines in (a) are for activity snippets among *specific* users. The bursty anomaly (orange) reveals users `u72603` and `u82199` (anonymized ids) commenting on `u82199`'s answer 36 times over the course of 1hr in the 3-month interval, then never interacting again. The timeline shown is zoomed-in on for clarity. The most persistent snippets (one shown in black) are users commenting on their own answers, suggesting that it is unusual for the same two distinct users to interact persistently over time. On the other hand, the subtly persistent anomaly (blue) reveals `u1950` regularly answering `u55747`'s questions. We show the persistent, bursty, and subtly anomalous snippets in (i)-(iii). We give the PvF plot used to identify these snippets in § A.4.

The right plot uses the view ϕ = Order, where any users can form an occurrence of the activity snippet. Remarkably, we find that activity involving the three interaction types (commenting on questions, commenting on answers, and answering questions) fall in distinct places on the plot. Activity involving comments on questions tends to fall in the bursty region of the plot, while answers are the most subtly persistent. To better understand the phenomenon, we zoom in on the region shown in the upper-left. We observe that most activity snippets in this box target a single user (i.e., all discussion is directed towards that user), and involve at least two distinct users. We give examples in (iv)-(v) of Fig. 6. In (iv), a user comments on another's post, the second user responds, and the original user comments again. In (v), the second user responds twice to the original comment. These snippets capture natural, technical discussions among users. Remarkably, these discussions center around questions (orange) and answers (black) at roughly equal frequencies, but they occur with higher persistence for answers. We conjecture that this is because, when a question is asked, it triggers a flurry of comments on it; but once the question is answered, these tend to die out. On the other hand, comments on *answers* persist, because new users may have the same question, and have follow-up questions, even months after the question has been answered. Since both have indistinguishable frequencies (10^5-10^7), this subtle difference in behavior is not revealed by frequency alone.

Takeaways on Social Networks. We find that subtly persistence activity can reveal users who do not interact often, but do *regularly*—i.e., *similar* users who are missed when only looking at how many times they interact. We also find that some types of posts promote continual activity, while others trigger bursts.

5.3 RQ2: Anomalies in Real Time

We next demonstrate that bursty and subtly persistent snippets can be identified in real time, right when they occur.

5.3.1 Generating Anomaly Scores. To identify anomalies automatically in real-time (without visually inspecting PvF plots as in § 5.2), we use the following process. When an activity snippet x appears in the stream at time t, we generate a 2D point <frequency,

persistence>, or $[F(x; [\textsc{start}(\mathscr{S}), t])$, $P(x; [\textsc{start}(\mathscr{S}), t])]$, corresponding to the dimensions of a PvF plot. Then, any streaming anomaly detection method can be applied. We use the Random Cut Forest (RCF) method [13], which gives a real-valued anomaly score for each point in a stream. For implementation details, see § A.5.

5.3.2 Real-time Qualitative Anomaly Detection (PvF). **Setup.** We analyze two networks: Reddit and NYC Taxi trips from the first two months of 2019. For each, we choose ground-truth snippets by choosing a subtly persistent anomaly, a bursty anomaly, and a snippet corresponding to a point from the middle of the PvF plot. We use sPENminer with $k_{\max} = 1$, and set (α, β, γ) values, which we give in § A.8. At each time t, we compute the anomaly scores of each ground-truth snippet (§ 5.3.1), and compare the scores to the median and standard deviation of the anomaly scores of all points seen to that point. We label an activity snippet at time t as a level 1, 2, or 3 anomaly if it is 1 to 2, 2 to 3, or 3+ standard deviations above the median score respectively.

Results. The results in Fig. 7 for Reddit (left) and NYC Taxi (right) show level 1, 2, and 3 anomalies colored green, orange, and maroon.

The subtly persistent snippet in Reddit corresponds to the subreddit `r/electronic_cigarette` referencing `r/ecrpoker`. This is the 3rd most persistent snippet overall, but only the 252nd most frequent. References between these subreddits occur remarkably regularly. Upon investigation, we found that `r/ecrpoker` was formed by a popular user in `r/electronic_cigarette`. We conjecture that the snippet corresponds to this user, or their followers, promoting the content of the other subreddit. The bursty anomaly corresponds to `r/nightly_pick` referencing `r/hockey`, presumably picking winners for each night's hockey game. The bursts align with hockey seasons. sPENminer consistently identifies this activity snippet as anomalous. Furthermore, the anomaly score decreases as expected over time, since as the bursts return yearly during hockey season, the snippet becomes more persistent. The third snippet is neither bursty nor regular, and it is correctly not flagged as an anomaly.

In NYC, the subtly persistent anomaly reveals a taxi trip from Kew Gardens Hills in Queens, to Manhattan, near the United Nations building. The taxi trip is repeated every day shortly after midnight, and is almost never taken at any other time. The nature of the trip is unknown, but surprisingly regular. The score, as expected, grows over time, as the continued regularity increases anomalousness. The bursty anomaly captures taxi trips departing from and arriving at the zone containing the NYC Taxi & Limousine Commission. The Taxi Commission inspects taxis and is open 5 days a week, which suggests that the bursts correspond to test-drives of taxis for inspection during business hours. sPENminer consistently identifies these bursts. Again, the third snippet is neither bursty nor regular, and is not often flagged as an anomaly.

5.3.3 Real-time Quantitative Anomaly Detection. **Setup.** We quantitatively analyze sPENminer's performance at identifying both subtle and bursty anomalies. For subtle anomalies, we use three months of Chicago Bike, from 01/2014 to 03/2014, and inject 50 synthetic bike trips that simulate infrequent, but lasting and surprisingly regular traffic, analogous to the first taxi trip in Fig. 7. We describe the exact injection procedure in § A.6. Each of the 50 anomalous trips occurs repeatedly, and the task is to identify

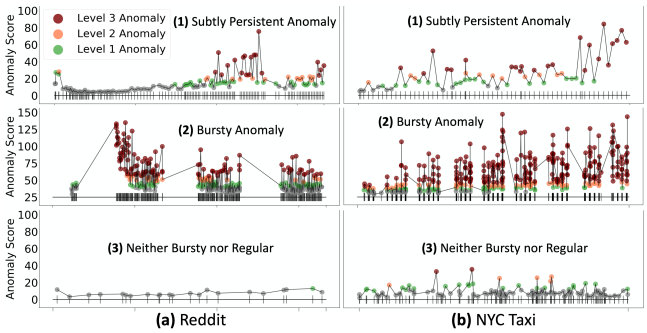

Figure 7: With PENminer, we are able to identify anomalies in *real-time*. Not only can we find bursty anomalies, but also subtly *persistent* anomalies: those that occur regularly and continually, but with frequency too low to be discovered by aggregate count alone. Anomaly levels capture how anomalous an occurrence is § 5.3.2.

Table 3: Results for identifying subtly persistent and bursty anomalies. Statistically significant results are marked with an "∗". sPENminer outperforms all baselines at identifying subtly persistent anomalies, which is not a well-studied problem. sPENminer also performs competitively with baselines on bursty anomaly detection, leading to the best overall performance (avg AUC).

	Metric	Freq	SedanSpot [11]	Midas-R [7]	DS [9]	sPENminer
Subtle	AUC	0.8325±0.02	0.4519±0.01	0.4520±0.02	0.7435±0.03	**0.9309**±0.00∗
	F1@100	0.0505±0.01	0.0001±0.00	0.0000±0.00	0.0076±0.00	**0.0508**±0.01
	F1@1K	0.1812±0.00	0.0035±0.00	0.0003±0.00	0.0378±0.01	**0.2580**±0.03∗
	F1@2K	0.1572±0.01	0.0098±0.00	0.0002±0.00	0.0561±0.01	**0.3292**±0.03∗
Bursty	AUC	0.8450±0.00	0.6390±0.00	**0.9434**±0.00∗	0.8632±0.00	0.8359±0.01
	F1@500K	**0.3089**±0.00∗	0.2745±0.00	0.3019±0.00	0.3063±0.00	0.2978±0.00
	F1@1M	**0.5351**±0.00∗	0.4527±0.00	0.5274±0.00	0.5295±0.00	0.5169±0.00
	F1@2M	0.7184±0.00	0.6309±0.00	**0.8378**±0.00∗	0.8066±0.00	0.7770±0.01
	Avg AUC	0.8388 (2)	0.5455 (5)	0.6977 (4)	0.8034 (3)	**0.8834** (1)

the occurrences of all 50 bike trips. We average results over ten random injection sets. For bursty anomalies, we use the DARPA IP network commonly used for the task [7, 11]. In this dataset, 2.7 million edges correspond to various bursty network attacks (e.g., denial of service). The goal is to identify edges that are part of these attacks. We use the same α, β, and γ as § 5.3.2, since we found them useful for visually identifying bursty and subtly persistent snippets.

Baselines. (1) Freq scores snippets as their number of occurrences, divided by the total number of all snippet occurrences. It can be thought of as [24] with motifs extended to activity snippets. (2) SedanSpot [11] and (3) Midas-R [7], state-of-the-art methods for *bursty* anomalies in edge-streams, output an anomaly score for each edge update. We set parameters as in the respective papers, and use the authors' code. (4) DS adapts the heuristic persistence of an item in a data-stream [9]. We apply DS exactly as PENminer, but replace the persistence in the 2D point with their heurisitc (described in § 3.4). For consistency, we follow the authors' suggestion of dividing the stream into 60 measurement periods, even though this unrealistically assumes that the stream length is known *a priori*.

Results. We give results in Tab. 3. sPENminer outperforms baselines at finding subtly persistent anomalies. Simultaneously, for bursty anomalies, it performs competitively with Midas-R, which is designed specifically for the task of finding bursty anomalies. On the other hand, the methods targeting bursts do not perform competitively at finding subtle anomalies. All but one result are statistically significant at a 0.01 p-value in a paired t-test.

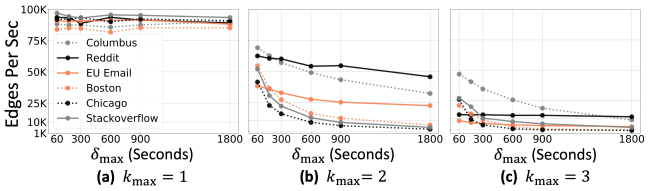

Figure 8: sPENminer's performance when varying $\delta_{\max}$, $k_{\max}$ on several datasets. Across all parameters, sPENminer processes edges in each stream 10K-360K times faster than the rate of the stream.

5.4 RQ3: Efficiency and Scalability

We evaluate whether sPENminer can process edges at least as quickly as they arrive in a stream, while allowing snippets to be reasonably sized ($k_{\max}$) and take a reasonable amount of time to form ($\delta_{\max}$). We then analyze how oPENminer scales with the number of edge-updates. We discuss hardware in § A.7.

5.4.1 sPENminer Efficiency. **Setup.** To evaluate sPENminer's efficiency over different $\delta_{\max}$ and $k_{\max}$ settings, we create plots for $k_{\max} \in \{1, 2, 3\}$. In each, we fix $k_{\max}$ and vary $\delta_{\max} \in \{60, 120, 180, 300, 600, 900, 1800\}$. We show edges processed per second for each parameter, for all datasets with rates less than 1 update/sec (since streams with significantly different rates are not comparable). Each point is averaged over 5 random intervals of 100K edge-updates (the same intervals across parameters).

Results. We show the results in Fig. 8. For $k_{\max} = 1$, activity snippets have duration $\delta = 0$, which is why in the first plot, the edges processed per second is consistent over all $\delta_{\max}$. Across all streams, and parameters, sPENminer processes edge-updates 10K to 360K times faster than the rate of the corresponding stream.

5.4.2 oPENminer Scalability. **Setup.** We evaluate how oPENminer scales with increasingly more edge-updates in Stackoverflow, our network with the most edges and nodes. We process the first 100K, 500K, 1M, 10M, 25M, and 50M edge-updates 5 times, and report the average runtime in seconds (Fig. 9). We fix $\delta_{\max} = 600$ sec (10 min) and evaluate $k_{\max} \in \{1, 2, 3\}$.

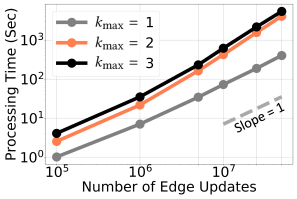

Figure 9: oPENminer scales linearly as the network grows.

Result. oPENminer scales linearly as the network grows.

6 CONCLUSION

In this paper, we propose mining persistent activity in continually evolving networks. Our precise, theoretical definition of persistence captures, beyond the aggregate number of occurrences, for how long and how regularly the activity has occurred. We propose PENminer (both offline and streaming variants) to measure the persistence of activity in evolving networks, and use it to gain a better understanding of networks by revealing activity that frequency alone could not, from infrequent but surprisingly regular trips in traffic networks to heated conversations in social networks. Future work includes further developing persistence-based anomaly detection, and techniques for automatic parameter tuning (α, β, γ).

ACKNOWLEDGEMENTS

This work is supported by an NSF GRFP Fellowship, the NSF under Grant No. IIS 1845491, Army Young Investigator Award No. W911NF1810397, and Adobe, Amazon, and Google faculty awards.

REFERENCES

[1] Motivate International Inc. https://www.motivateco.com/where-we-do-it/.
[2] NYC Taxi & Limousine Commission. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page.
[3] Ehab Abdelhamid, Mustafa Canim, Mohammad Sadoghi, Bishwaranjan Bhattacharjee, Yuan-Chi Chang, and Panos Kalnis. Incremental frequent subgraph mining on large evolving graphs. *IEEE TKDE*, 29(12):2710–2723, 2017.
[4] Charu C Aggarwal, Yao Li, Philip S Yu, and Ruoming Jin. On dense pattern mining in graph streams. *PVLDB*, 3(1-2):975–984, 2010.
[5] Rezwan Ahmed and George Karypis. Algorithms for mining the coevolving relational motifs in dynamic networks. *ACM TKDD*, 10(1):1–31, 2015.
[6] Cigdem Aslay, Muhammad Anis Uddin Nasir, Gianmarco De Francisci Morales, and Aristides Gionis. Mining frequent patterns in evolving graphs. In *CIKM*, pages 923–932. ACM, 2018.
[7] Siddharth Bhatia, Bryan Hooi, Minji Yoon, Kijung Shin, and Christos Faloutsos. MIDAS: Microcluster-Based Detector of Anomalies in Edge Streams. In *AAAI*, 2020.
[8] Thomas M Cover and Joy A Thomas. *Elements of information theory*. Wiley, 2012.
[9] Haipeng Dai, Muhammad Shahzad, Alex X Liu, and Yuankun Zhong. Finding persistent items in data streams. *PVLDB*, 10(4):289–300, 2016.
[10] Mohammed Elseidy, Ehab Abdelhamid, Spiros Skiadopoulos, and Panos Kalnis. Grami: Frequent subgraph and pattern mining in a single large graph. *PVLDB*, 7(7):517–528, 2014.
[11] Dhivya Eswaran and Christos Faloutsos. Sedanspot: Detecting anomalies in edge streams. In *ICDM*, pages 953–958. IEEE, 2018.
[12] Wenjie Feng, Shenghua Liu, Danai Koutra, Huawei Shen, and Xueqi Cheng. Unified dense subgraph detection. In *ECML/PKDD*, 2020.
[13] Sudipto Guha, Nina Mishra, Gourav Roy, and Okke Schrijvers. Robust random cut forest based anomaly detection on streams. In *ICML*, pages 2712–2721, 2016.
[14] Saket Gurukar, Sayan Ranu, and Balaraman Ravindran. Commit: A scalable approach to mining communication motifs from dynamic networks. In *SIGMOD*, pages 475–489. ACM, 2015.
[15] Chuntao Jiang, Frans Coenen, and Michele Zito. A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, 28(1):75–105, 2013.
[16] Chrysanthi Kosyfaki, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. Flow motifs in interaction networks. In *EDBT*, 2018.
[17] Lauri Kovanen, Márton Karsai, Kimmo Kaski, János Kertész, and Jari Saramäki. Temporal motifs in time-dependent networks. *JSTAT*, 2011(11):P11005, 2011.
[18] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2020.
[19] Rong-Hua Li, Jiao Su, Lu Qin, Jeffrey Xu Yu, and Qiangqiang Dai. Persistent community search in temporal networks. In *ICDE*, pages 797–808. IEEE, 2018.
[20] Richard Lippmann, Robert K Cunningham, David J Fried, Isaac Graf, Kris R Kendall, Seth E Webster, and Marc A Zissman. Results of the darpa 1998 offline intrusion detection evaluation. In *RAID*, volume 99, pages 829–835, 1999.
[21] Paul Liu, Austin R. Benson, and Moses Charikar. Sampling methods for counting temporal motifs. In *WSDM*, 2019.
[22] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. Graph summarization methods and applications: A survey. *ACM Comput. Surv.*, 51(3), 2018.
[23] Ron Milo, Shai S Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, D. Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298 5594:824–7, 2002.
[24] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. Motifs in temporal networks. In *WSDM*, pages 601–610. ACM, 2017.
[25] Abhik Ray, Larry Holder, and Sutanay Choudhury. Frequent subgraph discovery in large attributed streaming graphs. In *BigMine*, pages 166–181, 2014.
[26] Konstantinos Semertzidis, Evaggelia Pitoura, Evimaria Terzi, and Panayiotis Tsaparas. Finding lasting dense subgraphs. *DAMI*, 33(5):1417–1445, 2019.
[27] Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. Timecrunch: Interpretable dynamic graph summarization. In *KDD*, pages 1055–1064. ACM, 2015.
[28] Lorenzo De Stefani, Alessandro Epasto, Matteo Riondato, and Eli Upfal. Triest: Counting local and global triangles in fully dynamic streams with fixed memory size. *ACM TKDD*, 11(4):1–50, 2017.
[29] Maciej Walczyszyn, Shalin Patel, Maly Oron, and Bushra Mina. Battling superstorm sandy at lenox hill hospital: When the hospital is ground zero. *Critical care clinics*, 35(4):711–715, 2019.
[30] Qiankun Zhao, Yuan Tian, Qi He, Nuria Oliver, Ruoming Jin, and Wang-Chien Lee. Communication motifs: a tool to characterize social communications. In *CIKM*. ACM, 2010.

A SUPPLEMENT ON REPRODUCIBILITY

A.1 Complexity Analysis

In the special case of $k_{\max} = 1$, the window $\mathscr{W}$ need not be maintained (only singleton snippets are extracted). Thus, the per-update and total complexity (of oPENminer) are $O(1)$ and $O(|\mathscr{S}|)$ respectively. We now discuss $k_{\max} > 1$. To process a new update $\underline{u}_{\text{new}}$, the only non-constant cost comes from extracting new snippets in line 4 (lines 5-8 are $O(1)$). Let μ be the average rate of the stream $\mathscr{S}$ in updates per second. Then the average number of updates in a window $\mathscr{W}$ of width $w = \delta_{\max}$ seconds is equal to $\mu \cdot \delta_{\max}$. Lines 14-16 are $O(\mu \cdot \delta_{\max})$. While new snippets must be connected (cf. § 4), in the worst case all previous $\mu \cdot \delta_{\max}$ updates are connected with $\underline{u}_{\text{new}}$. Thus, there are $O\left(\sum_{k=1}^{k_{\max}} \binom{\mu \cdot \delta_{\max}}{k-1}\right) = O\left(\binom{\mu \cdot \delta_{\max}}{k_{\max}-1}\right)$ new snippets to extract (lines 18-20), which dominates $O(\mu \cdot \delta_{\max})$. Consequently, the per-update time can be controlled by choosing $\delta_{\max}$ to be reasonable based on the stream's rate. For oPENminer, the total time complexity to process $|\mathscr{S}|$ updates is $O(|\mathscr{S}|\binom{\mu \cdot \delta_{\max}}{k_{\max}-1})$.

A.2 Detailed Pseudocode

Algorithm 2 PENminer ($\mathscr{S}$, $\delta_{\max}$, $k_{\max}$, ϕ, α, β, γ, VARIANT)

Input: Stream $\mathscr{S}$, max snippet duration $\delta_{\max}$ and size $k_{\max}$, view ϕ, persistence exponents α, β, γ, the variant (oPENminer/sPENminer).

1: $w \leftarrow \delta_{\max}$ ▷ The window size enforces the maximum duration
2: $\mathscr{W} \leftarrow \emptyset$ ▷ Window is empty initially
3: **while** $\underline{u}_{\text{new}} \in \mathscr{S}$ **do** ▷ While there is a new update in the stream
4: **for** $x \in$ ExtractNewSnippets($\mathscr{W}$, $\underline{u}_{\text{new}}$, t, ϕ) **do**
5: **if** VARIANT is oPENminer **then**
6: $O_x \leftarrow O_x \cup \{t\}$ ▷ Add the new occurrence's timestamp
7: **else**
8: Update $P(x; [\text{START}(\mathscr{S}), t])$ via Thm. 3
9: **if** VARIANT is oPENminer **then**
10: **for** each x **do**
11: Compute $P(x; [\text{START}(\mathscr{S}), \text{END}(\mathscr{S})])$
12: **procedure** ExtractNewSnippets($\mathscr{W}$, $\underline{u}_{\text{new}}$, t, ϕ)
13: $\mathcal{X}_{\text{extracted}} \leftarrow \{\phi(\underline{u}_{\text{new}})\}$ ▷ Add the singleton snippet for $\underline{u}_{\text{new}}$
14: **for** $\underline{u} \in \mathscr{W}$ **do**
15: **if** $t - \text{TIMESTAMP}(\underline{u}) > w$ **then**
16: $\mathscr{W} \leftarrow \mathscr{W} \setminus \{\underline{u}\}$ ▷ Remove stale updates
17: $\mathscr{W} \leftarrow \mathscr{W} \cup \{\underline{u}_{\text{new}}\}$ ▷ Add the new update
18: **for** $k = 2, \ldots, k_{\max}$ **do**
19: **for** each new size k snippet x **do**
20: $\mathcal{X}_{\text{extracted}} \leftarrow \mathcal{X}_{\text{extracted}} \cup \{x\}$
21: **return** $\mathcal{X}_{\text{extracted}}$

A.3 Choosing α, β, γ

This section provides analysis of the exponents α, β, and γ. At the end of the section, we give suggestions for practitioners.

A.3.1 Rank Correlation Between Persistence and Components. We first show how varying the exponents affects how much each component contributes to persistence. To do so, we compare the ranking of snippets in descending order on $W(\cdot)$, $F(\cdot)$, and $S(\cdot)$, with the ranking in descending order based on persistence $P(\cdot)$. We use Kendall-Tau rank-correlation to compare rankings. For each component of persistence $W(\cdot)$, $F(\cdot)$, and $S(\cdot)$, we vary its corresponding exponent α, β, or γ over the values $\{0.2, 0.5, 1, 2.0, 5.0\}$, while fixing the other two exponents at 1. Each exponent has a plot in Fig. 10 showing the rank-correlation of that component with persistence. In general, as the exponent corresponding to a component is increased, that component becomes more correlated with persistence, while the others become less correlated.

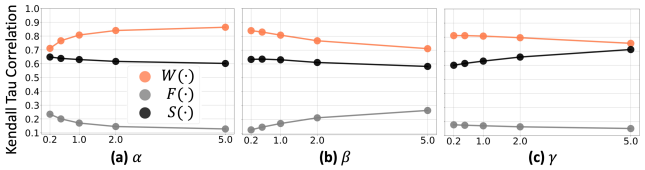

Figure 10: Kendall Tau rank correlation between snippets ranked by components of persistence and persistence itself, over various values of exponents. In general, as one exponent is increased, and the others fixed, the corresponding component becomes more correlated with persistence.

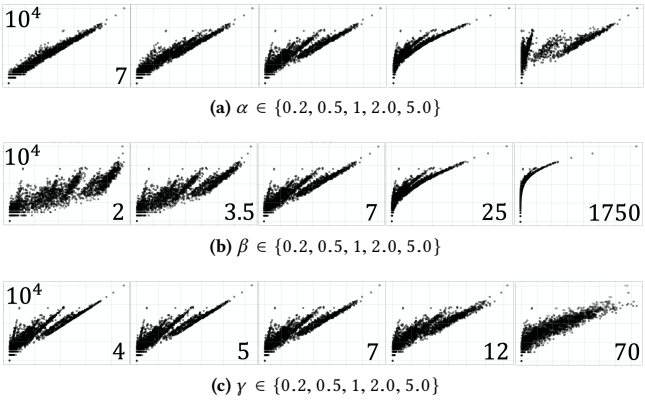

Figure 11: PvF plots for Columbus Bike varying each exponent in $\{0.2, 0.5, 1, 2.0, 5.0\}$, while fixing the other two at 1. The main takeaways are that small values of β (0.2 or 0.5) increase the spread of points, while increasing γ (2.0 or 5.0) emphasizes points in the lower left (i.e., very regular snippets). Thus, it is generally effective to set $\alpha = 1$, $\beta \in (0, 1)$, and $\gamma \in (1, \infty)$.

A.3.2 Effect on PvF Plots. We next show how varying the exponents changes PvF plots visually. Again, for each component of persistence $W(\cdot)$, $F(\cdot)$, and $S(\cdot)$, we vary its corresponding exponent α, β, or γ over $\{0.2, 0.5, 1, 2.0, 5.0\}$, while fixing the others at 1. We show plots for these exponents in Fig. 11. The value in the upper-left corner is the maximum frequency, and lower-right the maximum persistence. Since $W(\cdot) \in [0, 1]$, varying α does not change the range of persistence values. In contrast, $F(\cdot)$ is unbounded, and increasing it can cause the range of persistence values to grow significantly. The main takeaways are that small values of β (0.2 or 0.5) increase the spread of points, while increasing γ (2.0 or 5.0) emphasizes points in the lower left (very regular snippets). We chose exponents to emphasize the snippets of interest in our experiments.

A.3.3 Sensitivity on Anomaly Detection. We give anomaly detection results in Tab. 4, showing the effect of downweighting each exponent to 0.2. The results are mostly stable across exponents. The main exception, (0.2, 1, 1), leads to considerably better results on subtle anomalies. Since $W(\cdot) \in [0, 1]$, $\alpha = 0.2$ *up*-weights $W(\cdot)$.

Table 4: Additional results at identifying subtly persistent and bursty anomalies, showing the effect of parameters (α, β, γ).

	Metric	(1, 1, 1)	(0.2, 1, 1)	(1, 0.2, 1)	(1, 1, 0.2)
Subtle	AUC	0.709 ± 0.02	0.801 ± 0.02	0.712 ± 0.02	0.686 ± 0.03
	F1@100	0.006 ± 0.00	0.009 ± 0.00	0.005 ± 0.00	0.007 ± 0.00
	F1@1K	0.028 ± 0.00	0.051 ± 0.01	0.029 ± 0.00	0.028 ± 0.01
	F1@2K	0.042 ± 0.01	0.076 ± 0.01	0.043 ± 0.01	0.041 ± 0.01
Bursty	AUC	0.856 ± 0.01	0.867 ± 0.01	0.831 ± 0.01	0.853 ± 0.01
	F1@500K	0.307 ± 0.00	0.307 ± 0.00	0.307 ± 0.00	0.307 ± 0.00
	F1@1M	0.525 ± 0.00	0.527 ± 0.00	0.516 ± 0.01	0.524 ± 0.00
	F1@2M	0.763 ± 0.01	0.783 ± 0.01	0.742 ± 0.02	0.756 ± 0.01

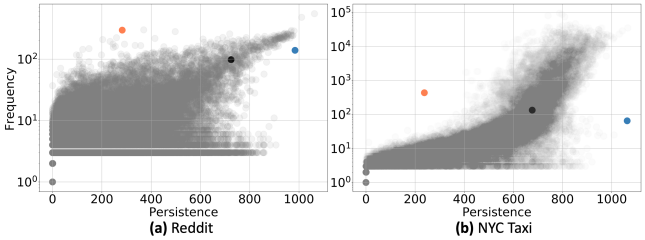

Figure 12: The plots used to extract ground-truth anomalies in § 5.3.2. Orange is the bursty anomaly, blue the subtly persistent anomaly, black the neither bursty nor subtly persistent snippet.

Since the subtly persistent anomalies (§ A.6) occur throughout the stream, we conjecture that $\alpha = 0.2$ increases their anomalousness.

A.3.4 Advice for Practitioners. For visual clarity, we found it generally effective to set $\alpha = 1$, $\beta \in (0, 1)$, and $\gamma \in (1, \infty)$—cf. A.3.2. If practitioners wish to analyze PvF plots visually, we recommend these values, especially $\gamma > 2$, to help discover subtly regular snippets. If other tasks are of interest, the exponents should be tuned for that task. Indeed, setting α, β, γ automatically for tasks like anomaly detection is an important direction for future work.

A.4 Choosing Activity Snippets

We discuss our choices of activity snippets for analysis in § 5.2.2-§ 5.3.2.

A.4.1 Stackoverflow. For Stackoverflow, we used the ground-truth points shown in Fig. 13. The snippets corresponding to the orange, blue, and black points are visualized in Fig. 6(a), (i)-(iii).

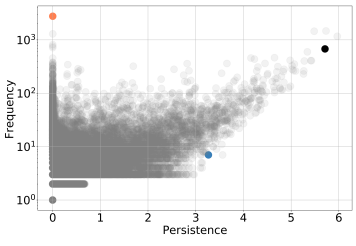

Figure 13: The plot used to extract ground-truth for Stackoverflow in § 5.2.2. Orange is the bursty anomaly, blue the subtly persistent anomaly, and black a persistent snippet.

A.4.2 Reddit and NYC Taxi. For Reddit and NYC Taxi, we used the ground-truth snippets in Fig. 12. The orange snippet for Reddit is `r/nightly_pick` referencing `r/hockey`, the blue is for `r/electronic_cigarette` referencing `r/ecrpoker`, and the black is for `r/bestof` referencing `r/personalfinance`. For NYC, orange is a trip from `zone207` to `zone207`, blue is a trip from `zone135` to `zone170`, and black a trip from `zone234` to `zone198`.

A.5 Using Random Cut Forests

We discuss details of using Random Cut Forests for anomaly detection in § 5.3. Throughout the experiments, we use 10 trees in the random forest, with each having a maximum depth of 256. To enforce the maximum size of trees, when the maximum size is reached, before adding a new point, we chose a leaf at random to remove. Since activity snippets can reoccur, when scoring a reoccurrence we make one minor adaption. When we score the point $[F(x; [\textsc{start}(\mathscr{S}), t]), P(x; [\textsc{start}(\mathscr{S}), t])]$, if we have already scored snippet x at some prior time $t' < t$, then we first remove the point $[F(x;[\textsc{start}(\mathscr{S}), t']), P(x;[\textsc{start}(\mathscr{S}), t'])]$ corresponding to the prior occurrence, to avoid scores decaying artificially due to prior occurrences of the same snippet.

A.6 Injecting Subtly Persistent Anomalies

We use the following procedure to inject subtly persistent anomalies for § 5.3.3. We inject bike trips into the first three months of Chicago Bike. For each anomalous bike trip, we select a start and end position at random within 10 minutes of the start and end of the stream, so that the trip covers most of the three months. We then select a number of occurrences $|O|_x$ from 5 to 100, weighted inversely proportional to the chosen number, to favor lower frequencies. We inject that many anomalies at roughly uniform intervals into the stream, but perturb the gaps from uniform by ±20 minutes to simulate realistic variance. The anomalous bike trips are chosen from among those not currently present in the stream so that they do not conflict with existing trips. The number of anomalous edges is the sum of the randomly chosen number of occurrences over all 50 bike trips, and these edges are labeled as 1 while the rest are 0. We generated 10 injection sets using different random seeds, and the exact number of resulting edges injected in each set was 3322, 3354, 2714, 2474, 3764, 3366, 3606, 2760, 3360, and 2560.

A.7 Hardware and Software

We perform all experiments on an Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz with 1TB RAM. Our code is implemented in Python.

A.8 Reference of Parameters Used

Table 5 gives the parameters used in each of the experiments.

Table 5: Reference of parameters used in our code for figures and tables reported in § 5. For maximum size of $k_{\max} = 1$, the duration of a snippet is always 0, in which case the maximum duration can be set arbitrarily without affecting results.

Tab./Fig.	$k_{\max}$	$\delta_{\max}$	Variant	View	ϕ	α	β	γ
Fig. 4 (Boston)	1	N/A	oPENminer	ID	1	0.5	2	
Fig. 4 (Columbus)	1	N/A	oPENminer	ID	2	0.5	3	
Fig. 5	1	N/A	oPENminer	ID	1	1	10	
Fig. 6 (left)	3	3600	oPENminer	ID	1	0.5	2	
Fig. 6 (right)	3	900	oPENminer	Order	2	0.5	10	
Fig. 7	1	N/A	sPENminer	ID	1	0.2	10	