Learning Thresholds for PV Change Detection from Operators' Labels

Dapeng Liu†, Youjian Zhao†, Kaixin Sui†, Shiwen Cheng†, Dan Pei†* , Chengbin Quan†
Jiao Luo‡, Xiaowei Jing§, Mei Feng§
†Tsinghua University ‡Baidu §PetroChina
†Tsinghua National Laboratory for Information Science and Technology (TNList)

***Abstract*—Page Views (PVs) are very crucial for search engines due to their close relationship to the revenue. When PVs change significantly, operators must be informed so that they can diagnose and fix the problem quickly, and prevent further loss. In reality, PVs can be counted in many ways (*e.g.,* PVs originated from different ISPs), and different PVs are of different interest to operators (*e.g.,* the PVs of a larger ISP is more important). As a result, different PVs often require different detection standards, or thresholds. However, attempts to tune a number of thresholds have been hampered by the cost of the manual effort involved.**

To address the above problem, we propose a practical framework, called PTL (practical threshold learning). Operators only need to provide a few simple labels about the detection results, then PTL will automatically tune the thresholds for different PVs. Using 4-month PVs from a global top search engine, our evaluation demonstrates that PTL can improve the accuracy of detection dramatically. More importantly, it introduces very little labeling overhead for operators. For example, when detecting the PVs of 103 ISPs, PTL can reduce the overall false negative rate from 96% to 9% using only 29 labels per week on average.

I. Introduction

Search engine has become one of the most used Internet applications. According to the data from ComScore [1], each of the global top search engines, such as Google, Baidu, Yahoo, Yandex, and Bing, has billions of searches monthly. The display ad and click ad that immediately follow the search results are the major revenue sources of search engines. It is thus critical for search engines to detect in real time any unexpected significant change in the number of searches served, or called Page Views (PVs), because these symptoms are to be potentially caused by anomalies, such as DDoS attach, data center failure, and service upgrade bugs. Those significant changes need to be timely reported so that operators can troubleshoot the possible causes, and fix them to prevent unnecessary potential revenue loss.

Our interview with the operators from one of the largest search engines shows that, both short-time dramatic PV changes and long-time gradual PV changes should be detected. In other words, the operators have potential *thresholds* of two aspects, *i.e.,* change severity and duration, for deciding whether the PV change significantly or not. These thresholds are also necessary for different change detection approaches [2], [3], [4], [5], [6], [7], [8]. However, the operators cannot easily quantify the thresholds in advance, as [9] can also attest. On the other hand, the default thresholds, assumed in literatures, often require onerous manually tuning in practice to achieve a reasonable detection accuracy. We argue that the threshold tuning is neither convenient nor scalable for the PV change detection due to the following challenges:

First, for the purposes such as fine-grained monitoring and revenue debugging [10], PVs are always counted in many ways (*e.g.,* the PVs from different ISPs, data centers, devices or advertisers). This requirement leads to a lot of different PVs for detection. Second, operators often have various detection standards for different PVs, mostly due to their different importance. For instance, a large ISP is deemed more crucial as it contributes a majority of PVs, thus deserving more careful detection. On the contrary, operators do not want to receive frequent alarms from a small ISP, unless its PV change is very serious. Therefore, the operators have to configure a number of thresholds to obtain a detection system as they expected. Third, some change detection approaches adopt complex techniques to measure the change [2], [3], [4], [5], [7], [6], and it is not intuitive for the operators to translate their detection standards into the corresponding thresholds. Last, the concepts of the significant PV changes in operators' mind could drift over time [8], so that the thresholds are often not set once and for all.

As a consequence of the above unaddressed challenges, the PV change detection in practice is still ad hoc and needs a lot of manual efforts. The operators we interviewed suggest that this kind of manually tuning is both inefficient and time-consuming. In this paper, we propose a novel thresholds learning framework, called *PTL*, standing for *Practical Threshold Learning*, to address the above challenges. Our contributions are as follows:

- PTL provides a practical way of automatically tuning the thresholds of an anomaly or change detectors. Operators just need to label the detection results, *e.g.*, whether the reported anomalies are true or there are anomalies missed. Given these operators' simple labels, PTL can tune the detection thresholds of a detection approach. More importantly, to be more practical, PTL does not require strictly precise labels from operators. For example, PTL tolerates incomplete and inaccurate labels.
- We propose a sharing mechanism to learn a number of thresholds for different ISPs more effectively. First, we identify the similarities of different ISPs from operators' labels. Then, PTL shares the learning outcomes of a single label on a certain ISP among the similar ISPs. Through this way, the number of labels required can be reduced obviously, and also the learning can be more effective.
- We use 4-month real PV data collected from one of the global top search engines to evaluate PTL. The results demonstrate that PTL can improve the detection accuracy greatly regardless what initial thresholds are set. Moreover, PTL introduces very little labeling overhead for operators. For example, PTL can reduce the false negative rate from 96% to 9% using only 29 labels per week on average.

The remainder of the paper is organized as follows. Section II presents the background of detecting significant PV changes and summarizes the goals. Section III describes the design details of PTL. Section IV evaluates PTL with real PV data. Section V reviews the related work, and Section VI concludes the paper.

* Dan Pei is the corresponding author.

978-1-4799-7575-4/14/$31.00 ©2015 IEEE

II. Background and Problem

In this section, we first introduce some basic concepts of significant PV changes. Then we motivate our design by modeling the existing change detection approaches and characterizing the threshold problem. Last, we give our goals and observations for solution intuitions.

A. PV

A Page View (PV) is defined as a successful response to a user's search request [11]. PVs are one of the most biggest concerns of search engines as they are very closely related to the ad revenue. In fact, there are many ways to count PVs, such as the PVs of different ISPs, devices and advertisers. In this paper, we focus on the ISP PVs. Specifically, the PVs of a ISP refer to the PVs whose source IP addresses are originated from that ISP. In the search engine we studied, there are 103 different ISPs in total.

B. Significant PV Change

To figure out the significant PV changes concerned by operators, Fig. 1 shows several real examples collected from the search engine we studied. Three main observations are as follows:

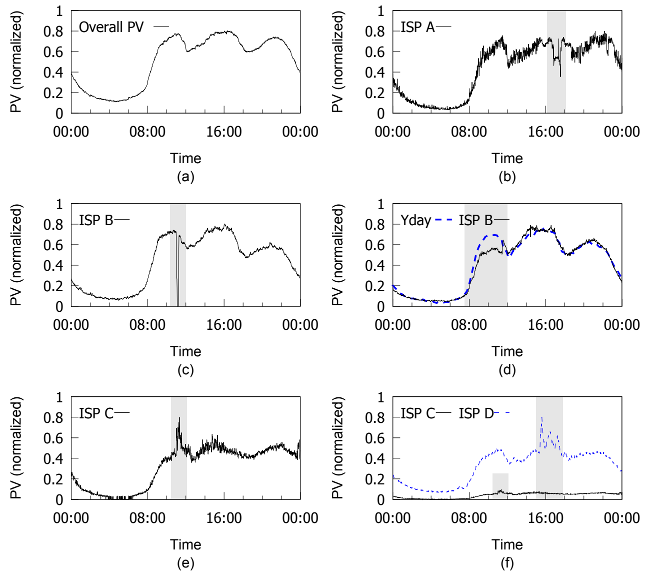


Fig. 1. Examples of significant PV changes concerned by operators. In compliance with confidentiality constraints, the PVs are all normalized in each plot. The shaded areas indicate the significant changes.

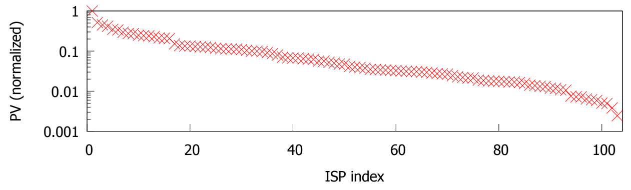


Fig. 2. Average PVs per day of the 103 ISPs. The PVs are normalized by the maximum value observed in the plot. The Y axis is in log scale.

- **ISP PVs more detailed than overall PVs**. This is because the overall PVs are too coarse to capture details. Also, they cannot help to diagnose the problem, *e.g.,* which ISP should be blamed for the PV change. As shown in Fig. 1 (a) and (b), while the overall PVs seem relatively smooth, the PVs of ISP A drop obviously in the same day due to a failure of the access network of ISP A. The problem lasts for 46 minutes, yet we see that it is completely invisible in the overall PVs.
- **Two metrics: the change severity and the duration**. Fig. 1 (c) shows a dramatic PV change of ISP B at about 11:00. The operators expect that this event can be detected at its very beginning. On the other hand, Fig. 1 (d) shows a gradual PV change of ISP B as well. The dashed line denotes its PVs of yesterday that is used as a baseline here. We observe that the PVs of ISP B start dropping slowly at 8:00 when compared to the baseline. However, the alarm should not be raised immediately. This is because the changes of such moderate severity are very common but rarely turn into real problems like Fig. 1 (d). From these two examples, we found that the operators take the change severity and the duration into account simultaneously to determine significant PV changes. Furthermore, these two metrics are interdependent. For example, in Fig. 1 (c), when the change severity is high, the threshold for the duration would be short.
- **Different detection standards for different ISPs**. Fig. 2 illustrates the average PVs per day of the 103 ISPs, which are called ISP (PV) scales in the rest of paper. We see that the ISP scales can differ greatly from each other. As such, those ISPs are considered differently by the operators. For example, in Fig. 1 (e) and (f), while there happens a PV change of similar shape for both ISP C and ISP D, the operators are only interested in the case of ISP D. This is because ISP D is of a large scale but ISP C is not (see Fig. 1 (f), where the PVs of ISP C in Fig. 1 (e) is plotted with the PVs of ISP D together under the same Y axis). Therefore, the PV change of ISP C would not affect enough users to trigger the operators' further investigation. Note that, besides the decreases, the abnormal increases of PVs can also indicate problems such as DDoS attacks and flash crowds, thus they should also be detected.

The above examples clearly demonstrate the complicated and flexible demands for PV change detection. To realize such detection, in addition to change detection approaches, a number of thresholds of the change severity and the duration for different ISPs are inevitable.

C. Thresholds Tuning: A Missing Puzzle of Existing Change Detection Approaches

Recently, many change detection approaches have been proposed and employed in real systems. Table I shows four examples of approaches and their high level ideas. Typically, the process of a change detection approach can be divided into several steps as shown in Fig. 3, which is similar to the detection models given by [9], [12].

- *Step 1:* When the data arrive, the change detection approach first quantify the change of each data point using a certain technique. For example, in Table I, [4] adopts a forecast based method called Holt-Winters and [5] uses a method of wavelet analysis. The change measures depend on both the technique used and its internal parameters, *e.g.,* the three weighted parameters α, β, and γ in Holt-Winters [4].
- *Step 2:* The change measures are further normalized, such as using the mean and the standard deviation of historical data [3], [2].
- *Step 3:* Single point change detection is then applied on each normalized change measure. Specifically, if the measure exceeds the **severity threshold**, it is identified as a significant change point.
- *Step 4:* To avoid triggering alarms more than necessary, those significant change points do not individually trigger alarms, but are filtered by the duration detection. That is, if the continuous significant change points exceed the **duration thresholds**, a

TABLE I

A SUMMARY OF FOUR CHANGE DETECTION APPROACHES. THE DEFAULT THRESHOLDS ARE ASSUMED IN THE RELATED LITERATURES.

Detection approaches	Change measures	Normalization	Thresholds
Historical average [3]	First, divide the data into hourly intervals, expecting that each interval avoids capturing the time-of-day effect. In this way, each data value V itself is directly used to measure the change of that point.	Based on the Gaussian distribution, use $\mathcal{C} = \|V - \mu\|/\sigma$ to normalize change measures, where μ is the mean and σ is the standard deviation of each interval.	$\mathcal{C} > 2$
Time series decomposition [2]	Decompose every data value V into three components: the trend, the season and the noise. Measure the changes using noise N.	Also use $\mathcal{C} = \|N - \mu\|/\sigma$ to normalize the change measure.	$\mathcal{C} > 1.96$
Holt-Winters [4]	Predict the data value of the t-th slot, P_t, using exponential smoothing processes on three components: the baseline, the linear trend, and the season. Measure the change with the residual $R_t = \|P_t - V_t\|$, where V_t is the real value of the t-th slot.	Maintain historical residuals HR_t via exponential smoothing and normalize the change measures with $\mathcal{C} = D_t/HR_{t-s}$, where s is the season length.	$\mathcal{C} > 2$
Wavelet analysis [5]	Deem the data as signals and decompose them into low, mid, and high frequency parts based on wavelet analysis.The variances of the high and the mid frequency, H and M, are used to measure the changes.	Use a weighted sum of H and M to get the normalized score $\mathcal{C}$.	$\mathcal{C} > 1.7$

significant change is identified and an alarm is triggered. Here, the severity thresholds and the duration thresholds are collectively called **detection thresholds**.

In the above process, the detection thresholds play a very important role in identifying or defining the significant changes, thus affecting the detection accuracy greatly. However, how to adjust those thresholds conveniently and effectively has not been explored very much. A very common solution is to simply assume some default detection thresholds as shown in Table I. But these empirical thresholds often cannot meet the detection requirement [2], [13], and thus requiring a lot of manual tuning. To the best of our knowledge, [14] provides an automatic way to tune the thresholds according to labeled data. Yet their method only focuses on single point detection. We will show later in Section III that when combined with the duration thresholds together, the problem of threshold tuning becomes more complicated. Moreover, we are not aware of any prior work that intends to reduce the labeling overhead when there are a number of thresholds (*e.g.,* for different ISPs) to learn.

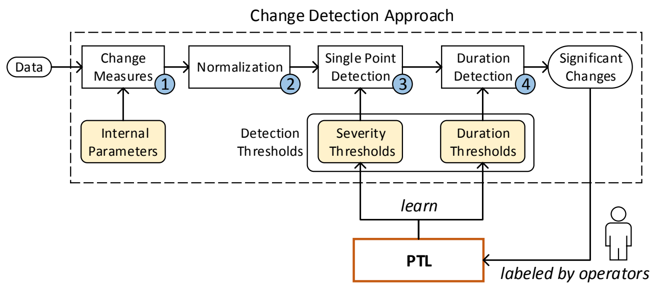


Fig. 3. The overview of change detection approaches and the position of PTL.

In addition to the detection thresholds, the internal parameters can also affect the detection. Some previous works [15], [16], [17] have proposed several automatic ways to choose the proper internal parameters, *e.g.,* multi-pass grid search [16]. In this paper, we focus on the detection thresholds rather than the internal parameters. Our focus has not been studied deeply in real systems. Those internal parameters search methods are complementary to our work.

D. Design Goals

To resolve the above problems, we propose a framework, called PTL. The goals of PTL are:

- Learning the detection thresholds from operators' labels on the detection results, rather than letting operators manually tune the thresholds directly. This is based on the fact that it is more straightforward for operators to visually inspect the PVs and label the significant changes they confirm [2], [6], [8], [5], [18], [19], [20].
- Being robust to incomplete and inaccurate labels, which always exist in practice.
- Reducing the labeling overhead, *i.e.,* the number of labels needed, when learning many thresholds of different ISPs.

The overview of PTL is shown in Fig. 3. It takes as input the operators' labels about the detection results, and then learns the severity thresholds and the duration thresholds accordingly.

E. Key Observations and Solution Intuitions

We have three key observations of significant PV change detection, and we take advantage of these observations to guide the design of PTL.

OBSERVATION 1: The change significance monotonically increases with the change severity and the duration. Give two intervals of PVs m and n, if the change of m is more severe than n (*e.g.,* m decreases by 50% and n decreases by 10%), and the change duration of m is longer than that of n, then m is considered more significant by operators. In other words, if n should be detected, then m should be detected too.

OBSERVATION 2: The labeling error exists, but can be relative small. Considering the 46-minute significant PV change in Fig. 1 (b), when let operators label that window, they can seldom label the boundaries exactly, but often provide a rough and wider window. However, this extending error can be relatively small. For example, it is quite easy for the operators to label that 46-minute window via a 60-minute window (about 30% extension). By taking advantage of this, we can learn operators' detection standards from each false negative label conservatively (Section III-D2).

OBSERVATION 3: Same detection standards for the ISPs of similar PV scales. Through months of field work with the operators, we found that despite the operators' bias on the ISP scales when they determine significant changes, they always have the same detection standard for the ISPs of similar PV scales. The only problem is that they cannot describe this intuition (*i.e.*, which scales are similar) precisely. This observation motivates our design of sharing learning outcomes in Section III-E, which can reduce the labeling overhead and improve the effectiveness of learning.

III. PTL Design

In this section, we first formalize several key concepts in the PV change detection, and give the assumptions and requirements of the operators' labels. Then we describe how PTL learns the detection thresholds from labels and shares the learning outcomes among similar ISPs.

A. PVs and Significant PV Changes

PVs can be represented by time series data $\{p_1, p_2, ..., p_t\}$, where t is the slot index and p_t is the sum of PVs in the t-th time slot[1]. Since the change severity and the duration are considered when determining significant PV changes, which means the changes should continue for a period of time before triggering alarms, we first give the definition of a window, then define significant changes based on it.

Definition 1: Window and window length. Let t_0 and t_1 be two time slot indexes, where $t_0 \le t_1$, then a window w refers to the time slots between t_0 and t_1 (including t_0 and t_1), denoted as $w = t_0 \sim t_1$. The length (or duration) of w is denoted as l_w and $l_w = t_1 - t_0 + 1$. We say a time slot index $i \in w$ if $i \ge t_0 \wedge i \le t_1$.

Definition 2: Change severity of a window. Given a window w, let p_i be the PVs of the i-th slot, where $i \in w$. A change detection approach would measure the change severity of p_i, which is denoted as s_i and $s_i \ge 0$. The larger s_i is, the more severe p_i changes. Then the change severity of the window w is denoted as s_w, and $s_w = min\{s_i \,|\, i \in w\}$. Such definition of s_w is to satisfy the requirement of continuous change detection. In particularly, let $\mathbf{sThld}$ be the severity threshold, then $s_w > \mathbf{sThld} \Leftrightarrow \forall s_i, i \in w : s_i > \mathbf{sThld}$, which indicates that all the slots in w change significantly.

Definition 3: Detection threshold and significant change. We define a *detection threshold* in either of a *closed* form or a *open* form. This is because the labels of false positives and false negatives have different implications of the threshold boundary (details will be discussed later in Section III-D). Let $\mathbf{lThld}$ be the duration threshold, then the closed form detection threshold is denoted as $\tau = [\mathbf{lThld}, \mathbf{sThld}]$. It means that a window w is called a significant change if $l_w \ge \mathbf{lThld} \wedge s_w \ge \mathbf{sThld}$; the open form detection threshold is denoted as $\tau = (\mathbf{lThld}, \mathbf{sThld})$. Similarly, it indicates that a window w is called a significant change if $l_w > \mathbf{lThld} \wedge s_w > \mathbf{sThld}$. For convenience of later discussions, we use $\mathcal{F}(w, \tau) = 1$ to represent that w is identified as *a significant change* under the detection threshold τ; on the other hand, $\mathcal{F}(w, \tau) = 0$ means that w is *a normal change*.

Definition 4: Detection threshold set. Since operators may have different duration thresholds for different change severities, for each ISP, we use a ***detection threshold set*** $\mathcal{T} = \{\tau\}$, which contains multiple individual detection thresholds τ, to capture the operators' detection demands. Then we define $\mathcal{F}(w, \mathcal{T}) = 1 \iff \exists \tau \in \mathcal{T}, \mathcal{F}(w, \tau) = 1$; otherwise, $\mathcal{F}(w, \mathcal{T}) = 0$. That is, w is a significant change under the detection threshold set $\mathcal{T}$ if w violates any detection threshold τ in $\mathcal{T}$.

B. Monotonic Increase Property

According to the Observation 1, for a certain ISP, $\mathcal{F}(w, \mathcal{T})$ monotonically increases with l_w and s_w. An illustrative example is shown in Fig. 4. Suppose two windows m and n, the duration and the change severity of n are both no less than those of m. Naturally we deem that $\mathcal{F}(n, \mathcal{T}) \ge \mathcal{F}(m, \mathcal{T})$. In particular, as shown in Fig. 4 (a), if m is identified as a significant change, so is n. Generally, the top-right shaded area of m is the ***significant change area*** according to m, and any window falls into this area should be identified as a significant change. On the other hand, as shown in Fig. 4 (b), if n is a normal change, then m should be a normal change as well. The bottom-left shaded area of n is the ***normal change area*** according to n, and any window fall into this area should be identified as a normal change. Formally, we have: $l_n \ge l_m \wedge s_n \ge s_m \Rightarrow \mathcal{F}(n, \mathcal{T}) \ge \mathcal{F}(m, \mathcal{T})$ and $l_n \le l_m \wedge s_n \le s_m \Rightarrow \mathcal{F}(n, \mathcal{T}) \le \mathcal{F}(m, \mathcal{T})$.

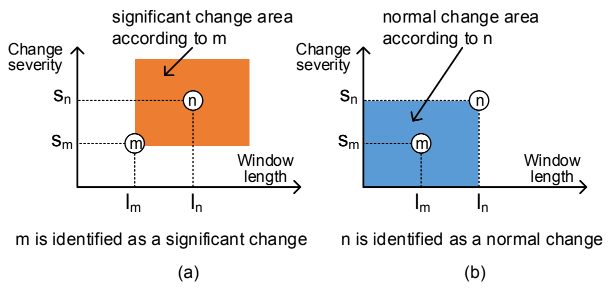

significant change area according to m
Change severity
s_n
s_m
n
m
Window length
l_m
l_n
m is identified as a significant change
(a)
normal change area according to n
Change severity
s_n
s_m
n
m
Window length
l_m
l_n
n is identified as a normal change
(b)

Fig. 4. $\mathcal{F}(w, \mathcal{T})$ monotonically increases with the change severity and the duration.

C. Definitions, Assumptions and Implications of Labels

By setting an initial threshold set $\mathcal{T}$ without any manual tuning, PTL begins to detect significant PV changes. PTL requires operators to label what they *think* about the alarms, so that PTL can learn the threshold set from the labels.

In actual use, operators have two opportunities to check the detection results of PTL, one is when they receive alarms raised by PTL, and they would verify whether the alarms are false positives; the other is when they conduct some visual inspections upon PV or are warned by a third party (*e.g.,* user complainants or other existing monitors), they would check whether PTL has missed those alarms, called false negatives. Therefore, we consider only two types of labels: false positive labels and false negative labels. We do not require, assume or use any true positive or true negative labels. Furthermore, we do not assume that *all* the false positives and false negatives would be labeled by operators. We now introduce a few definitions and realistic assumptions about operators' labels.

Definition 5: Label and label period. A label is denoted as $\mathcal{L}(P) = \{\text{FP}, \text{FN}\}$, where P is the period where operators disagree with the detection result of PTL, FP means false positive, *i.e.*, the alarms raised by PTL is false, and FN means false negative, *i.e.*, there are significant changes missed by PTL. The length of P is denoted as l_P.

For $\mathcal{L}(P) = \text{FN}$, the period P is specified by operators manually, which can contain multiple continuous significant changes. For example, if operators label a 20-minute period as FN, every 5-minute window in the period can still be considered significant if it appears individually. For $\mathcal{L}(P) = \text{FP}$, the period P is the alarm period given by PTL. The continuous significant changes detected by PTL will be merged as one alarm period. Such merging is apparently a design trade off. On one hand, merging can greatly reduce the overhead of labeling and increase the operators' willingness to label. On the other hand, labeling the merged windows instead of multiple individual ones gives us coarse information to learn from, which will become clear later. We choose to be conservative with the hope that we can continuously get useful labels from operators, although in a slower way.

Definition 6: Label requirements and implications. Suppose that $\mathcal{T}_\mathcal{E}$ is the underlying threshold set that most satisfies the detection

[1]We use 1-minute slot in this paper, which is a tradeoff between the temporal granularity and the computation overhead.

requirements of operators, *i.e.,* the significant changes detected by $\mathcal{T}_\mathcal{E}$ are as desired. Then A false positive label $\mathcal{L}(P) = \text{FP}$ indicates that $\forall w \subset P : \mathcal{F}(w, \mathcal{T}_\mathcal{E}) = 0$, and we call $\{w\}$ the ***false positive windows***. So operators should label an alarm as false positive only if they *think* there is no significant changes (true positives) in the alarm period P. A false negative label $\mathcal{L}(P) = \text{FN}$ indicates that $\exists w \subset P : \mathcal{F}(w, \mathcal{T}_\mathcal{E}) = 1$, and we call w the ***false negative window***. When operators label a period as false negative, they should make sure there exists at least one window of significant change (false negative) in P. Also, a labeled false negative period should not have intersections with any alarm period already detected by PTL.

ASSUMPTION 1: False negative labels. According to the OBSERVATION 2, we assume that when operators provide a false negative label $\mathcal{L}(P) = \text{FN}$, the false negative slots are more than the true negative slots within P, or the proportion of false negative slots is more than 50%. It is quite easy for operators to label in such precision. We do not assume 100% precision because it would unrealistically require operators to exactly label the beginning and ending slots each time, and this sensitivity would hurt operators' willingness to label.

ASSUMPTION 2: Maximum delay of alarms. Obviously, given a detection threshold $\tau = [\mathbf{lThld}, \mathbf{sThld}]$ or $\tau = (\mathbf{lThld}, \mathbf{sThld})$, an alarm must wait for at least $\mathbf{lThld}$ or $\mathbf{lThld} + 1$ slots before raised. In order to detect with less delay, we assume that $\mathbf{lThld}_{max}$ (the maximum of $\mathbf{lThld}$) should be no more than 30 minutes (slots).

D. Learning from Labels: Toy Examples

Now, we show how PTL learns the threshold set $\mathcal{T}$ via two toy examples. The basic idea is that, for each label $\mathcal{L}(P)$, PTL adjusts $\mathcal{T}$ so that $\mathcal{T}$ could detect P as suggested by $\mathcal{L}(P)$, *e.g.,* if $\mathcal{L}(P) = \text{FP}$, the updated $\mathcal{T}$ should also identify P as a normal change instead of a significant change. As more labels are learned in this way, $\mathcal{T}$ would converge to $\mathcal{T}_\mathcal{E}$ and detect as expected.

1) Learning from a false positive label: First, Fig. 5(a) shows the change severities of five slots. With the initial threshold set $\mathcal{T} = \{[1, 2]\}$, those slots are detected as significant changes and form a continuous alarm period $1 \sim 5$. The initial threshold set $\mathcal{T} = \{[1, 2]\}$ is represented by the double circle in Fig. 5(b), and it results in the significant change area represented by those closed boundaries. Notice that, since the window length is measured by discrete values, *i.e.,* the number of slots, the significant change area is also discrete along the dimension of the window length.

Then the alarm in Fig. 5(a) is labeled as a false positive $\mathcal{L}(P) = \text{FP}$, where $P = 1 \sim 5$. According to the implication of the false positive label, $\forall w \subset P : \mathcal{F}(w, \mathcal{T}_\mathcal{E}) = 0$. Thus we need to adjust $\mathcal{T}$ so that $\mathcal{F}(w, \mathcal{T}) = 0$. The adjustment of $\mathcal{T}$ is called the ***learning outcome***, which will be shared among similar ISPs later in Section III-E.

We obtain false positive windows $\{w\}$ like this: first, for $l_w = 1$, we can get five windows by sliding 1-slot window across P. The severities of these five windows have three distinct values *i.e.,* 2, 3, 4. These three kinds of windows are illustrated by the three squares at window length = 1 in Fig. 5(c). Similarly, for $l_w = 2, 3, 4, 5$, we can obtain another five distinct combinations of change severities and window lengths, illustrated by other five squares in Fig. 5(c).

The false positive label implies that all those false positive windows should be identified as normal changes. Then based on the monotonic increase property (Section III-B), all the bottom-left areas of those false positive windows are the normal change areas, and should be eliminated from the original significant change area. As a result, we can get a new significant change area as shown in Fig. 5(d), which can be described by a new threshold set $\mathcal{T}' = \{(1, 4), (2, 3), (3, 2), [6, 2]\}$. $\mathcal{T}'$ can thus detect as the false positive label indicates. Note that $\mathcal{T}'$ does not have to contain all the boundary points of the significant change area, *e.g.,* $\tau = (4, 2)$ and $\tau = (5, 2)$, since $\tau = (3, 2)$ can detect what can they detect.

Additionally, in order to satisfy the ASSUMPTION 2, *i.e.,* $\mathbf{lThld}_{max} \leq 30$ minutes, if $l_P > 30$ minutes, we will only obtain the positive windows w whose $l_w \leq 30$ minutes. This also avoids learning from an excessive number of positive windows in the case of very a long l_P.

2) Learning from a false negative label: The five slots in Fig. 5(e) are labeled as false negatives, denoted as $\mathcal{L}(P) = \text{FN}$, where $P = 1 \sim 5$. The initial threshold set $\mathcal{T} = \{[2, 4]\}$ and its corresponding significant change area is shown in Fig. 5(f). The false negative label indicates that $\exists w \subset P : \mathcal{F}(w, \mathcal{T}_\mathcal{E}) = 1$. A naive method to obtain the false negative window w is to deem the entire labeled period as w. However, as aforementioned, operators can also involve true negatives in the period. Therefore, learning from the labeled period directly can lead to an incorrect detection threshold set. For example, when operators label $1 \sim 5$, they could consider only the period of $2 \sim 4$ as false negatives, implying that $\mathcal{T}_\mathcal{E} = \{[3, 3]\}$. However, if we obtain the false negative window w from $1 \sim 5$, *i.e.,* $l_w = 5$ and $s_w = 2$, the learned threshold set would be $\mathcal{T} = \{[5, 2]\}$, which is wrong as as it can detect significant changes that $\mathcal{T}_\mathcal{E} = \{[3, 3]\}$ does not agree with.

To resolve the above problem, we conservatively learn from each false negative label based on the ASSUMPTION 1. Given a label $\mathcal{L}(P) = \text{FN}$, we first calculate the median of the change severities of all the slots in P, denoted as s_{median}. Then we estimate the false negative window w as $l_w = l_P$ and $s_w = s_{median}$. The reason of using the median is that we assume that at least 50% slots in P are false negatives. Since the change severities of false negative slots are larger than those of true negative ones, s_{median} could always be the change severity of a false negative slot. In the example of Fig. 5(e), we can obtain a false negative window w of $l_w = 5$ and $s_w = s_{median} = 3$, which is represented by the cross in Fig. 5(g). The false negative label indicates that w should be identified as a significant change. Again, according to the monotonic increase property, the top-right area of w is the significant change area, and should be added to the original one. The updated significant change area is shown in Fig. 5(h), and can be captured by the new threshold set $\mathcal{T}' = \{[2, 4], [5, 3]\}$. Considering the ASSUMPTION 2, if a false negative label is longer than 30 minutes, we estimate the length of the false negative window using 30 minutes to avoid producing a threshold with $\mathbf{lThld}_{max} > 30$ minutes.

E. Sharing Learning Outcomes

So far, we have introduced how PTL learns the detection threshold set for one ISP from its labels. Nevertheless, labeling many ISPs can cost a lot of time for operators. We resolve this problem through sharing the learning outcome of each label among the ISPs of similar PV scales. This solution is based on the OBSERVATION 3. Here, the learning outcome refers to the adjustment of the threshold set of the labeled ISP, and sharing means applying the adjustment also to the threshold sets used for other ISPs that are not labeled directly by operators, but have similar PV scales with the ISP labeled. However, except that the ISP scales matters, we do not know exactly which ISPs are treated as similar. We design PTL to learn the similarities of ISPs from the operators' labels as well.

The intuition is that every label on an ISP can reveal the operators' detection preferences about the ISP. For instance, if an ISP receives

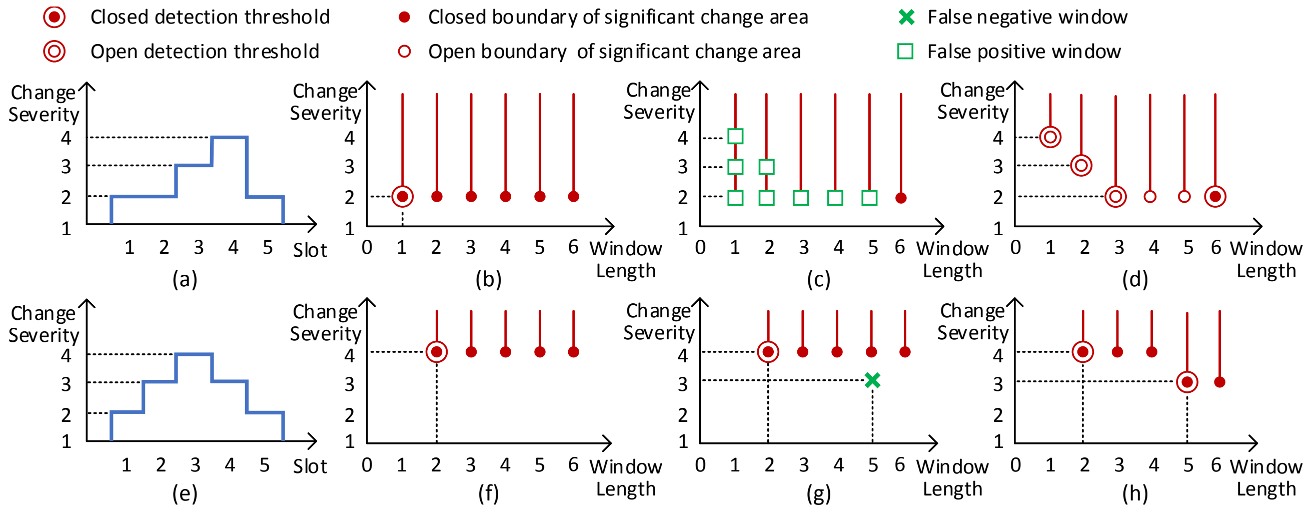
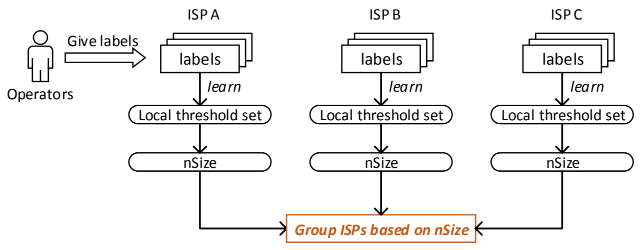
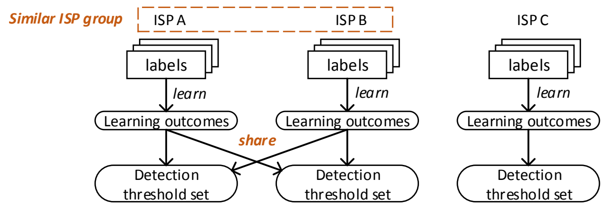


Fig. 5. Examples of learning from a false positive label (a)(b)(c)(d) and a false negative label (e)(f)(g)(h). (a) shows the change severity of five slots that have been labeled as false positives. (b) shows the initial detection threshold set $\mathcal{T} = \{[1, 2]\}$ and the corresponding significant change area. (c) shows the false positive windows obtained from the false positive label in (a). (d) shows the new detection threshold set $\mathcal{T}' = \{(1, 4), (2, 3), (3, 2), [6, 2]\}$ learned from the label. For the case of the false negative label, similarly, (e) gives the change severity of five slots that have been labeled as false negatives. (f) shows the initial detection threshold set $\mathcal{T} = \{[2, 4]\}$ and the significant change area derived from it. (g) shows the false negative window estimated from (e). (i) shows the learned detection threshold set $\mathcal{T}' = \{[2, 4], [5, 3]\}$.



(a) Grouping similar ISPs.



(b) Sharing learning outcomes in the similar ISP groups.

Fig. 6. Sharing learning outcomes among similar ISPs.

several false positive labels, learning from which, the threshold set of the ISP becomes larger (*i.e.,* the normal change area gets larger), then the ISP is supposed to be concerned less crucial by the operators as they would like to receive only very serious PV changes of the ISP. Based on this intuition, our key idea is to find similar ISPs according to the sizes of their normal change area.

Fig. 6 shows the overview of the method. The first step is to group similar ISPs. As shown in Fig. 6(a), the labels of an ISP are used to learn its ***local threshold set*** as described in Section III-D. Since the local threshold set is learned completely from the direct labels of the ISP, it can reflect the operators' detection preference regarding the ISP. We define the size of the normal change area derived from the local threshold set as $nSize$. In the calculation of $nSize$, there could be some window lengths corresponding to unlimited change severity. For example, the window length of 1 in Fig. 5(f). It means no matter how severe PVs change in one slot, it should not be reported. To calculate $nSize$ in such case, we use a relatively large change severity, *i.e.,* 10, instead of the unlimited one, since in our data, there are very few slots whose change severities can exceed 10.

Algorithm 1 GROUP $(ispList, \theta)$

```
1:  $\mu$ is the mean and $\sigma$ is the standard deviation.
2:  for $isp_i$ in $ispList$ do
3:      CALCULATE $nSize$ and $sRank$ of $isp_i$
4:  while $ispList$ is not empty do
5:      SORT $ispList$ by $nSize$ in ascending order
6:      for $isp_i$ in $ispList$ (i starts from 1) do
7:          CALCULATE the similarity between $isp_i$ and $isp_0$:
8:          $sim_i = |(isp_i.nSize - isp_0.nSize) \times (isp_i.sRank - isp_0.sRank)|$
9:          CALCULATE $\mu$ and $\sigma$ of $\{sim_1, sim_2, ..., sim_i\}$
10:         if $|sim_i - \mu| > \theta \cdot \sigma$ then
11:             // Splitting a group
12:             $RANK = max\{isp_k.sRank | k < i\}$
13:             OUTPUT $\{isp | isp.sRank \leq RANK\}$ as a group
14:             delete the group from $ispList$
15:             break
```

We group similar ISPs as described in Algorithm 1. $sRank$ denotes the PV scale ranking of an ISP. First, $sRank$ and $nSize$ of each ISP are calculated. We sort ISPs by their $nSize$ in ascending order. The ISP with minimum $nSize$ (*i.e.,* isp_0 in $ispList$) is under the most strict (smallest) detection threshold set, and is categorized into the first ISP group. The ISPs belonging to the same group should have similar $nSize$ and $sRank$, so we calculate the differences of $nSize$ and $sRank$ between isp_0 and isp_i. As $nSize$ and $sRank$ are not measured by the same unit, we use the absolute product of their differences to measure how similar isp_0 and isp_i are. When the similarity of isp_i deviates a lot, a group is split. The similarity deviation is quantified by the mean and the standard deviation, and if the similarity of isp_i deviates the mean by $\theta\times$ the standard deviation, we deem isp_i does not belong to the group. We assume the normal distribution and use $\theta = 2$ to achieve 95% confidence in statistics [21]. After grouping by $nSize$, the ISP, whose $sRank$ is higher than that of at least one ISP already in the group, is also categorized into the group. This is because though some ISPs are of similar scales with isp_0, they might not be labeled by operators, and their $nSize$ is not updated. As a result, they could be excluded from the group by mistake if only considering $nSize$. We group the ISPs once a week (a typical season of the PVs).

Fig. 6(b) shows that, in the same group, the learning outcomes are applied to every ISPs, generating the ***detection threshold set*** for the actual change detection. In this way, one label on a single ISP can have effect on many ISPs, thus improving the learning effectiveness and reducing the labels needed.

IV. Evaluation

In this section, we evaluate PTL through the simulation with 4-month PV data collected from a top search engine. The PVs are originated from 103 ISPs.

A. Methodology

1) Change Detection approach: We use a popular change detection approach to work with PTL, *i.e.*, *time series decomposition* [2], which could deal with both the long-term trend and the seasonality of PVs. The high level idea is described in Table I. It first breaks down the PVs of each slot into three components: the long-term trend, the seasonality and the noise. Since the noise does not contain the normal changes caused by the trend and the seasonality, it is used for the change detection. Besides, to avoid significant changes contaminating the mean and the standard deviation, those data are excluded when calculating the mean and the standard deviation. More details can be found in [2].

2) Results Validation: Evaluating the accuracy of PTL requires the ground truth, a complete set of significant changes that should be detected. The ground truth set is also used to simulate the operators' labels when the detection results are wrong. One way to obtain the ground truth is using the *real world tickets* that record the PV significant changes verified by operators manually. The advantage of using the tickets is that they are the real world events and interested by the operators. However, the tickets are often rare in reality. For example, the search engine we studied has only maintained the tickets for the overall PVs rather than the ISP PVs in their database. Also, the tickets can also introduce errors and disagreement [6], [8]. Thus, we could not draw strong conclusions about the performance of PTL based on the tickets.

A second way, commonly used in prior works [22], [6], [2] to bypass the above issue, is *pair-wise validation*. The core idea is to compare the detection results with another approach or different detection configurations. The advantage of this way is that we can obtain the ground truth automatically for all the ISPs. Obviously, the drawback is that the significant changes in such ground truth set have not been verified by the operators, which will take a huge amount of efforts to investigate.

In this paper, we adopt both of the above two ways, so that we can obtain the benefit of each method. There are 24 history tickets of the overall PV for the 4-month data we used. As for the pair-wise validation, because different detection approaches can differ in the change patterns they are designed to detect (*e.g.,* identifying the edge or the duration of changes), their results cannot be compared fairly. To avoid those influences and clearly show the threshold learning performance, we use the detection results of the same detection approach, *i.e.,* the time series decomposition, but under different threshold settings as the ground truth. Another advantage of this choice is that we can compare the thresholds directly since the two detection approaches are the same and their thresholds are of the same meaning. Finally, the thresholds used to obtain the ground truth are set as follows: we divide the 103 ISPs into three groups by their scales, which are 18 large ISPs, 59 medium ISPs, and 26 small ISPs. The threshold sets for each group are $\mathcal{T}_{large} = \{[2, 7.84], [5, 5.88], [10, 3.92]\}$, $\mathcal{T}_{medium} = \{[5, 7.84], [10, 5.88]\}$, and $\mathcal{T}_{small} = \{[10, 7.84]\}$. Here, the principle is that, in the same group, the larger the severity thresholds are, the shorter the duration thresholds should be. The severity thresholds are $4\times$, $3\times$, $2\times$ of 1.96 used in [2].

The above validation methods are probably not entirely true in reality, but give a good approximation and handles to evaluate our system. In the simulation, we let PTL start with an arbitrary threshold set for all the ISPs, which can be quite different from the ones used to generate the ground truth, and see how PTL can learn them from the "operators' labels".

3) Performance metrics: To quantify the performance of PTL, we adopt four metrics: (a) the false positive rate (FPR), referring to the percentage of false alarms in all alarms raised by PTL; (b) the false negative rate (FNR), that is the percentage of significant changes missed by PTL; (c) the number of the checks of the detection results and the number of labels; (d) the threshold convergence, which compares the threshold sets learned by PTL with those used to generate the ground truth. Since each threshold set contains multiple combinations of the severity threshold and the duration threshold, they cannot be quantitatively compared directly. We measure the difference of two threshold sets of one ISP using the difference between the $nSize$ derived from them, and averaging the differences of the 103 ISPs.

4) Labeling Conditions: To be more practical, we will evaluate PTL under different labeling conditions. There are four aspects of the simulation of labeling:

- *Checking possibility* ($C\%$) represents that (a) when PTL raises an alarm, operators have the chance of $C\%$ to check it; (b) when there are PV significant changes in the ground truth, operators have the chance of $C\%$ to verify whether PTL has raised alarms. In both of the cases, if the detection result are not as expected, operators will provide a label.
- *Maximum of checks per day* (C_{max}) indicates that operators would check PTL for C_{max} times at most per day.
- *Mislabeling possibility* ($M\%$) means that when operators check PTL, they have the chance of $M\%$ to label by mistake, *e.g.,* the alarm is correct but they label it as a false positive by misoperation. Since operators would not check PTL without any purposes, we do not consider that operators will label false negatives when they do not believe them.
- *Labeling extension* ($E\%$) means that when operators label a false negative, *i.e.,* a certain significant change period missed by PTL, they could label the truly period by at most $E\%$ extension.

We provide two types of representative labeling conditions as shown in Table II, namely lazy operators and careful operators.

TABLE II
The simulation of two types of labeling conditions.

Name	$C\%$	C_{max}	$M\%$	$E\%$
Lazy operators	20%	20	5%	100%
Careful operators	80%	40	1%	10%

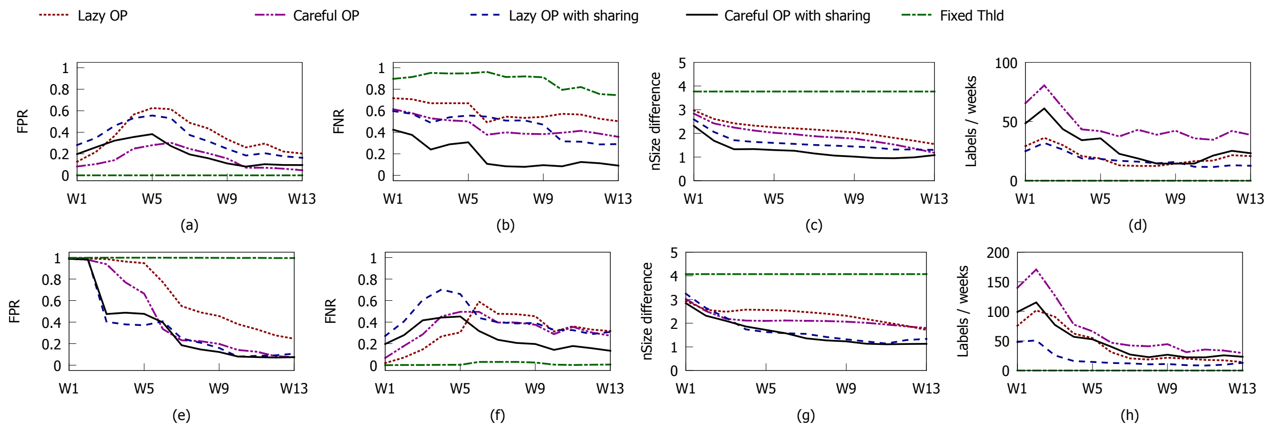


Fig. 7. Detection performance of the 103 ISPs in pair-wise validation. The X axis is the index of weeks in the 4 months. The Y axis represents different performance metrics, *i.e.,* FPR, FNR, the average $nSize$ difference (Section IV-A3), and the number of labels per week. (a)(b)(c)(d) shows the results when the initial threshold set is $\mathcal{T} = \{[10, 9.8]\}$ and (e)(f)(g)(h) shows that of $\mathcal{T} = \{[2, 1.96]\}$. Because the significant changes are infrequent in some weeks, which makes FPR and FNR jitters a lot, all the four metrics are obtained using 3-week moving average. That is also why the X axis ends at W13 rather than W16 (4 months).

B. Results

1) Real History Tickets: We run time series decomposition upon the overall PV in three threshold learning ways: no learning (fixed Thld), PTL with lazy operators (lazy OP), and PTL with careful operators (careful OP) in Table II. Also we start PTL with different initial threshold sets $\mathcal{T}$, from very small ones to very large ones. In the detection process, operators are supposed to check PTL according to the tickets only. Table III shows the results. Since the tickets can be incomplete, we can only determine how many tickets have been detected, but cannot provide the accurate FPR or FNR.

TABLE III
DETECTION RESULTS OF THE OVERALL PV (VALIDATED BY TICKETS).

Initial $\mathcal{T}$	Threshold Learning	#Alarms	Identified Tickets(%)	#Labels	#Checks
$\{[2, 1.96]\}$	Fixed Thld	3744	100%	-	-
	Lazy OP	84	79%	16	28
	Careful OP	54	87%	25	68
$\{[3, 3.92]\}$	Fixed Thld	108	83%	-	-
	Lazy OP	39	83%	4	10
	Careful OP	51	83%	25	56
$\{[5, 5.88]\}$	Fixed Thld	18	67%	-	-
	Lazy OP	18	79%	4	8
	Careful OP	30	79%	20	27
$\{[7, 7.84]\}$	Fixed Thld	13	50%	-	-
	Lazy OP	25	58%	4	9
	Careful OP	24	75%	9	41
$\{[10, 9.8]\}$	Fixed Thld	4	17%	-	-
	Lazy OP	13	46%	2	7
	Careful OP	34	67%	19	49

First, focusing on $\mathcal{T} = \{[2, 1.96]\}$, we see that though the fixed Thld identified all the tickets, it triggers 3744 alarms. They are much more than necessary in contrast with other cases, and thus most of them are supposed to be false alarms. This is because the initial $\mathcal{T}$ is set too aggressively. On the other hand, using the same initial $\mathcal{T}$, PTL can reduce the number of alarms greatly, and still many of the tickets can be identified (*e.g.,* 87% tickets for the careful OP). Second, the results also demonstrate an important fact that it is difficult to set the most suitable thresholds once and for all. See $\mathcal{T} = \{[3, 3.92]\}$ and $\mathcal{T} = \{[5, 5.88]\}$, the fixed Thld seemingly yields a relatively good results, but PTL can still improve them by lowering the number of alarms while still identifying the same number of tickets, or more tickets. Note that, in these two cases, the careful OP raises more alarms than the lazy OP. This is probably because the 24 tickets in the database missed some real significant PV changes, and through learning, they are identified by the careful OP. Third, as for $\mathcal{T} = \{[10, 9.8]\}$, PTL can also achieve better results but the improvement is less, compared with other cases. The reason is that the initial $\mathcal{T}$ is too large and PTL needs false negative labels to adjust $\mathcal{T}$. Nevertheless, we lack tickets (only 24 tickets) to produce such labels. Last, we observe that both the labels and the checks required by PTL are relative few no matter which initial $\mathcal{T}$ is used. This indicates the labeling overhead is low for operators when using PTL.

2) Pair-wise Validation: In Fig. 7, we detect the 103 ISP PVs with pair-wise validation as described in Section IV-A2. Two experiments are conducted with different initial $\mathcal{T}$, a large one $\mathcal{T} = \{[10, 9.8]\}$ and a small one $\mathcal{T} = \{[2, 1.96]\}$. In both experiments, we compare the performance of five threshold learning methods. In addition to the three methods introduced in Section IV-B1, we also compare PTL with sharing learning outcomes, namely "lazy OP with sharing" and "careful OP with sharing" in Fig. 7.

First, for $\mathcal{T} = \{[10, 9.8]\}$ (Fig. 7(a), (b), (c), (d)), the main deficiency of the accuracy at beginning is the FNR. This is because the initial threshold set $\mathcal{T}$ is too large and few significant changes can be identified. In Fig. 7(b) we see that, the FNR of the fixed Thld is always above 75%, and reaches 96% for the worst case. Although its FPR is 0 (Fig. 7(a)), such detection results are useless for operators. As for the results of PTL, the FNR is reduced strikingly as learning. In particular, the FNR of week 1 has already been decreased to 41% for the careful OP with sharing, from 90% of the fixed Thld. During PTL learns the thresholds, the FPR increases but then goes down as shown Fig. 7(a). This is because the significant changes actually detected are different from the ones in the ground truth, and this further affects normalizing the measures of the change severity since different data are eliminated from the calculation of the mean and the standard deviation (Section IV-A1). As a result, the change severity of the labeled period could be smaller from the perspective of PTL, leading to that the thresholds are over adjusted. This influence would be mitigated as learning. We see the FPR and the FNR at week 13 are 8% and 9% respectively for the careful OP with sharing. We also observe that sharing learning outcomes can further improve the detection accuracy (Fig. 7(a), (b)), compared with no sharing, and converge the thresholds more quickly (Fig. 7(c)), which is measured by the average $nSize$ difference as described in Section IV-A3.

In addition to the accuracy, another important performance metric is the number of labels required. As shown in Fig. 7(d), PTL requires only a few labels per week. Specifically, the careful operators (without sharing) need to provide 47 labels every week on average for the 4 months, while the careful OP with sharing only needs to provide 29 labels every week on average. We see that sharing learning outcomes can reduce the labeling overhead by 38% here. As for the lazy operators, as they provide less labels in nature (about 20 labels per week on average), sharing learning outcomes does not benefit them a lot, reducing 3 (15%) labels per week on average.

The conclusions of $\mathcal{T} = \{[2, 1.96]\}$ are much similar except two things. One is that the FPR (Fig. 7(e)) and the FNR (Fig. 7(f)) are exchanged when compared with $\mathcal{T} = \{[10, 9.8]\}$. This is because $\mathcal{T} = \{[2, 1.96]\}$ is very aggressive and triggers too many false alarms, so the FPR is the major problem instead of the FNR. Another is that the number of labels (Fig. 7(h)) is about twice as that in $\mathcal{T} = \{[10, 9.8]\}$ (Fig. 7(d)). This is also caused by the excessive false alarms, and operators would have more chances to label false positives. But the number of labels decreases quickly as learning, and is finally under 25 per week for careful OP with sharing.

V. Related Work

Many change detection approaches have been proposed in recent years. [2] employs change detection in a search engine as we do, but they focus on the overall search response time, not like the PVs that can be aggregated in many ways. There are some work attempting to apply change detection approaches in ISP networks, such as [4], [3], [7], [8], [22], [6], [13], [9]. Although several sophisticated change detection approaches are designed, we argue that it is still inconvenient for operators to use them in the real world, especially when there are a large number of metrics for detecting (*i.e.,* PVs from many ISPs). One of main problems is tuning thresholds, which is neither intuitive nor scalable for operators. To solve this, [14] provides an automatic way to adjust the severity thresholds based on the labeled data, but they ignore the duration thresholds which make the problem more complicated. Besides, we are not aware of any prior work trying to reducing the labeling overhead of learning many thresholds for detecting different metrics. Some work also devotes to automatically tune the internal parameters of change detection approaches, such as [15], [16], while our focus is on learning the detection thresholds from operators' labels. Those works are complementary to ours.

VI. Conclusion

PVs are crucial for search engines as they are closely related to the ad revenue. To monitor PVs in a fine granularity, PVs are always counted in many ways. For example, the PVs of different ISPs. When applying change detection approaches on those PVs, it requires a lot of thresholds, including both the change severity thresholds and the duration thresholds, to characterize the detection requirements of operators. However, these thresholds are typically tuned manually, and such attempt has been hampered by the cost of manual efforts. To adjust the thresholds more conveniently and effectively, we propose a framework, called PTL, intending to learn the thresholds from the operators' simple labels of the detection results. Through the simulation on 4-month real PV data from a global top search engine, we demonstrate the effectiveness of PTL. For example, in our experiment, PTL can decrease the FNR from 96% to 9% with only 29 labels per week on average.

We believe that PTL is a very useful framework to help tune the thresholds in significant PV change detection. We also believe that PTL can be extended to other detection scenarios as well.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61472214, the National Key Basic Research Program of China (973 program) under Grant No. 2013CB329105, the State Key Program of National Science of China under Grant No. 61233007, the National Natural Science Foundation of China (NSFC) under Grant No. 61472210, the National High Technology Development Program of China (863 program) under Grant No. 2013AA013302.

References

[1] Statistics of google. http://searchengineland.com/google -worlds-most-popular -search-engine-148089.

[2] Yingying Chen, Ratul Mahajan, Baskar Sridharan, and Zhi-Li Zhang. A provider-side view of web search response time. In *SIGCOMM 2013*.

[3] Suk-Bok Lee, Dan Pei, M Hajiaghayi, Ioannis Pefkianakis, Songwu Lu, He Yan, Zihui Ge, Jennifer Yates, and Mario Kosseifi. Threshold compression for 3g scalable monitoring. In *INFOCOM, 2012 Proceedings IEEE*, pages 1350–1358. IEEE, 2012.

[4] He Yan, Ashley Flavel, Zihui Ge, Alexandre Gerber, Daniel Massey, Christos Papadopoulos, Hiren Shah, and Jennifer Yates. Argus: End-to-end service anomaly detection and localization from an isp's point of view. In *INFOCOM 2012*. IEEE.

[5] Paul Barford, Jeffery Kline, David Plonka, and Amos Ron. A signal analysis of network traffic anomalies. In *SIGCOMM Workshop on Internet measurment*, pages 71–82. ACM, 2002.

[6] Augustin Soule, Kavé Salamatian, and Nina Taft. Combining filtering and statistical methods for anomaly detection. In *IMC 2005*.

[7] Ajay Mahimkar, Zihui Ge, Jia Wang, Jennifer Yates, Yin Zhang, Joanne Emmons, Brian Huntley, and Mark Stockert. Rapid detection of maintenance induced changes in service performance. In *CoNext 2011*.

[8] David R. Choffnes, Fabián E. Bustamante, and Zihui Ge. Crowdsourcing service-level network event monitoring. In *SIGCOMM 2010*. ACM.

[9] Shashank Shanbhag and Tilman Wolf. Accurate anomaly detection through parallelism. *Network, IEEE*, 23(1):22–28, 2009.

[10] Ranjita B., Rahul K., Ramachandran R., George V., Surjyakanta M., Hemanth M., and Piyush S. Adtributor: revenue debugging in advertising systems. In *NSDI 2014*.

[11] Jason Burby and Angie Brown. Web analytics definitions. *Washington DC: Web Analytics Association*, 2007.

[12] F. Silveira and C. Diot. Urca: Pulling out anomalies by their root causes. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, March 2010.

[13] Ajay Anil M., Han Hee S., Zihui G., Aman S., Jia W., Jennifer Y., Yin Z., and Joanne E. Detecting the performance impact of upgrades in large operational networks. In *SIGCOMM 2010*. ACM.

[14] Y. Himura, K. Fukuda, K. Cho, and H. Esaki. An automatic and dynamic parameter tuning of a statistics-based anomaly detection algorithm. In *ICC '09*.

[15] Jake D Brutlag. Aberrant behavior detection in time series for network monitoring. In *LISA*, pages 139–146, 2000.

[16] Balachander K., Subhabrata S., Yin Z., and Yan C. Sketch-based change detection: methods, evaluation, and applications. In *IMC 20013*.

[17] Cycles analysis methods. http://ray.tomes.biz/cy302.htm.

[18] Daniela B., Xenofontas D., Arno W., and Kavè S. Anomaly extraction in backbone networks using association rules. In *IMC 2009*.

[19] Xin Li, Fang Bian, Mark Crovella, Christophe Diot, Ramesh Govindan, Gianluca Iannaccone, and Anukool Lakhina. Detection and identification of network anomalies using sketch subspaces. In *IMC 2006*.

[20] Fernando Silveira, Christophe Diot, et al. Astute: Detecting a different class of traffic anomalies. In *SIGCOMM 2010*.

[21] 68-95-99.7 rule. http://en.wikipedia.org/wiki/68-95-99.7_rule.

[22] Yin Zhang, Zihui Ge, Albert Greenberg, and Matthew Roughan. Network anomography. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, IMC '05, pages 30–30, Berkeley, CA, USA, 2005. USENIX Association.