Label-Less: A Semi-Automatic Labelling Tool for KPI Anomalies

Nengwen Zhao[†‖], Jing Zhu[†‖], Rong Liu[‡], Dapeng Liu[§], Ming Zhang[¶], Dan Pei[*†‖]
[†]Tsinghua University [‡]Stevens Institute of Technology [§]BizSeer [¶]China Construction Bank
[‖]Beijing National Research Center for Information Science and Technology (BNRist)

***Abstract*—KPI (Key Performance Indicator) anomaly detection is critical for Internet-based services to ensure the quality and reliability. However, existing algorithms' performance in reality is far from satisfying due to the lack of sufficient KPI anomaly data to help train and evaluate these algorithms. In this paper, we argue that labeling overhead is the main hurdle to obtain such datasets.**

Thus we novelly propose a semi-automatic labelling tool called *Label-Less*, which minimizes the labeling overhead in order to enable an ImageNet-like large-scale KPI anomaly dataset with high-quality ground truth. One novel technique in *Label-Less* is *robust and rapid anomaly similarity search*, which saves operators from scanning and checking the long KPIs back and forth for abnormal patterns or label consistency. In our evaluations using 30 real KPIs from a large Internet company, our anomaly similarity search achieves the best F-score of 0.95 on average, and a real-time per-KPI response time (less than 0.5 second). Overall, the feedback from deployment in practice shows that *Label-Less* can reduce operators' labeling overhead by more than 90%.

I. Introduction

In recent years, Internet-based services, such as search engines and online shopping, have become an indispensable part of our daily life. To ensure undisrupted business, operators of these companies need to closely monitor various time series data called *Key Performance Indicators* (KPIs, *e.g.*, search response time, CPU usage), to accurately detect KPI anomalies and trigger timely troubleshooting/mitigation [1].

Despite the importance of KPI anomaly detection and also a large body of algorithms proposed in the literature [2]–[10], existing algorithms' performance in reality is far from satisfying. One key reason is the lack of generality in most of these algorithms. KPIs in practice have various types of patterns, yet the published algorithms are typically trained by private datasets, and thus cannot be easily generalized to other contexts. The majority of existing public time series datasets mainly aim at classification, clustering or regression [11] [12], not for anomaly detection. Although there exist a few public anomaly detection datasets like Yahoo Benchmark [3] and Numenta Anomaly Benchmark [4], they only contain KPIs with limited data points and synthetic anomalies. Consequently, the community of KPI anomaly detection is in an urgent need for a large-scale and diverse KPI anomaly dataset. Such a dataset would greatly benefit the anomaly detection research and practice just like what ImageNet [13] did to the fields of computer vision and deep learning.

* Dan Pei is the corresponding author.

Unfortunately, obtaining a large-scale KPI anomaly dataset with high-quality ground truth has been a great challenge due to following reasons. First, labeling KPI anomalies takes domain knowledge of IT operations. Thus the number of people who can reliably label in our domain is much fewer than in image and speech areas. Besides, even the experienced operators may not have enough confidence to perform high-quality labeling since it is difficult to give an explicit definition of "what is an anomaly" in various KPIs and applications. Second, it is labor intensive to carefully examine a several-month-long KPI back and forth and try to label anomalies in a consistent manner. For example, according to our observation, it takes an experienced operator a few hours, with the help of a labeling tool such as *Curve* [14] which has a dedicated user interface, to label a 6-month-long KPI with 300,000 data points spaced at 1-minute interval. Third, we fundamentally need to label a large number of KPIs to train anomaly detection algorithms, because the anomalies in each KPI are relatively rare, KPI patterns are diverse, and the number of KPIs in practice is huge in large companies. In a real and very recent example, when preparing for the KPI anomaly detection algorithm competition [15], 5 large Internet companies (Alibaba, Tencent, Baidu, eBay, and Sogou) sponsored us with thousands of KPIs (2-month to 6-month long), yet we ended up being able to publish just 30 labeled KPIs due to the high overhead to obtain high-quality ground-truth.

From the above discussion, it is apparent that *labeling overhead has become the main hurdle to large-scale KPI anomaly dataset, which in turn is the main hurdle to effective and practical KPI anomaly detection*. In this paper, we aim to tackle the KPI anomaly labeling overhead problem and propose a semi-automatic labeling framework called *Label-Less*. Our idea is inspired by one key observation: the majority of operators' labeling efforts are spent on visually scanning through the KPIs to gauge their "normal" patterns and variations *and* on examining KPIs back and forth to check whether "similar" KPI segments (see Fig. 1 for example) are labeled consistently. We argue that above "scanning and checking" should be facilitated by automatic tools driven by algorithms. In our idea, the visual scanning is replaced by **unsupervised anomaly detection** which emphasizes on high recall and acceptable precision, and the results are candidate potential anomalies. The manual consistency checking is replaced with our novel technique **anomaly similarity search**, which given an *anomaly template* by the operator, *i.e.*, a KPI segment

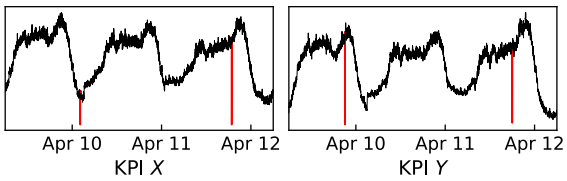


Fig. 1: Similar anomalies (labeled in red) in KPIs X and Y.

containing known anomalous data points, discovers anomalies similar to the template among the candidate anomalies.

Our major contributions are summarized as follows:

- To the best of our knowledge, this is the first work to focus on reducing the labeling workload for KPI anomaly detection by algorithms. Our proposed framework *Label-Less* including an unsupervised anomaly detection component and an anomaly similarity search component, is a general one to tackle labeling overhead reduction problem. Besides, the feedback from deployment in practice shows that *Label-Less* reduces more than 90% labeling time for operators.
- This paper is the first one that applies Isolation Forest [16] to unsupervised KPI anomaly detection, with features given by time series prediction models. Experiments have showed the unsupervised anomaly detection approach adopted in *Label-Less* has better performance and lower computation complexity compared with other alternatives, *e.g.*, One-Class SVM [17] and Local outlier factor [18].
- This is the first attempt to define and study the new problem: search anomalies similar to a given anomaly template from a set of candidate anomalies. Through experiments on various KPIs, the average best F-score can reach about 0.95. Besides, given that labeling's interactive nature requires fast response time, we propose to reduce the response time significantly through an *accelerated Dynamic Time Warping (DTW)* approach, which systematically adopts various techniques including constrained DTW, lower bound, and early stopping. The response time for one KPI is under 0.5 second. In addition to reducing labeling workload, the problem has other practical applications, as discussed in §VII.

II. Definitions and Overview

In this section, we first define a few key terms and formulate the problem of *anomaly similarity search*, then present an overview of *Label-Less* framework.

A. Definitions.

KPI: KPI is a special type of time series data. A KPI can be denoted as $X = \{x_1, x_2, \cdots, x_n\}$, where x_i is the value corresponding to time index i for $i \in 1, 2, \cdots, n$, and n is the length of the KPI. Compared with general time series, KPIs are based on the domain knowledge of IT operations. Besides, the length of one KPI is long and the number of KPIs is huge about millions to tens of millions in large companies [9], [19].

KPI Anomaly: Anomalies refer to the data points in a KPI that do not conform to the expected behavior and significantly differ from the normal data [2] [5], *e.g.*, jitters, spikes and dips. Since an anomalous case always has its context instead of an individual point, it is a collection of continuous points called *anomaly segment* (**anomaly** for short hereinafter), which can be formalized as a subsequence of KPI X, denoted as $segment(i) = \{x_i, x_{i+1}, \cdots, x_{i+m-1}\}$, where m is the length of the anomaly.

Anomaly template: An anomaly template is an anomaly segment chosen by an operator according to his interest, denoted as q with length m, which is the input to the *anomaly similarity search* problem. For example, the fisrt segment in red color in Fig. 1 can be an anomaly template.

Similar anomaly: According to some distance measure, $segment(i)$ with length m which has a small distance with the anomaly template q is considered as a similar anomaly to the template q. We require that two similar anomalies have the same length (m) for the following two reasons. First, there are no known techniques to support similarity search of arbitrary lengths [20]. Second, the techniques of lower bound used to speed up DTW in §IV-C requires that two segments must have same length. Otherwise, the search would be too slow for the interactive labeling.

B. Anomaly Similarity Search

Operators provide an anomaly template q, which is an anomaly case that operators are interested in, such as the dip of the number of requests, and the goal of "anomaly similarity search" is to discover top-k similar anomalies from candidate anomalies (given by unsupervised anomaly detection) with the given template q. We opted to use k-NN search since it is intuitive and practical. In contrast, range search which aims to find all similar anomalies where their distances are less than the tolerance value is very cumbersome to use in practice, since the tolerance value is not intuitive for operations and it varies under different KPIs. Therefore, it is hard to determine a suitable tolerance value in advance [21].

Anomaly similarity search is a little different from general time series similarity search [20] [22]. First of all, anomaly information is indispensable in our scenario. In other words, what we are searching for is not only a segment similar to the template but also an anomaly. Thus the search space here is candidate anomalies given by unsupervised anomaly detection instead of the entire KPI. Besides, different from traditional time series, *e.g.*, Electrocardiogram (ECG) and Electroencephalogram (EEG), a large number of long KPIs bring a great challenge in search responsiveness. The difference is demonstrated by experiments in §V-D.

C. Overview of Label-Less

The overall framework of *Label-Less* shown in Fig. 2 has two major components. The first component, *Unsupervised Anomaly Detection*, takes a set of related unlabeled KPIs (*e.g.*, the number of requests per server in a well load-balanced server cluster) as its input, and identifies a set of candidate anomalies as its output. In detail, we apply Isolation Forest [16] as an outlier detector and use the features extracted from time series models such as Holt-Winters [6]. Then an appropriate detection threshold θ is selected with bias towards

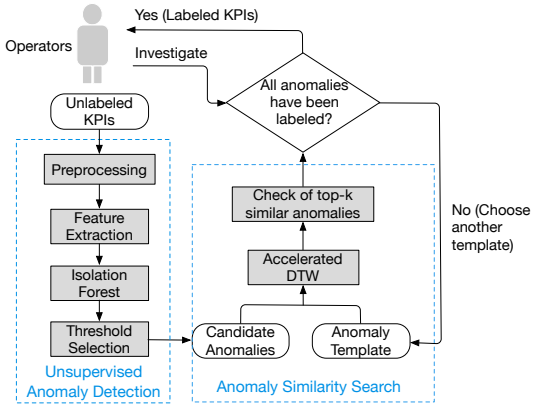


Fig. 2: The overall framework of *Label-Less*.

high recall and acceptable precision. As we will demonstrate in our experiments later (§V-C), with a high recall rate, most anomalies can be included as candidate anomalies. Moreover, unsupervised anomaly detection not only provides candidate anomalies which can save the visual scanning time for operators, but also plays a key role in anomaly similarity search. It filters out normal points so as to reduce false positives, *i.e.*, similar normal segment (§V-D). Besides, it can significantly reduce the search space and improve search efficiency (§V-F).

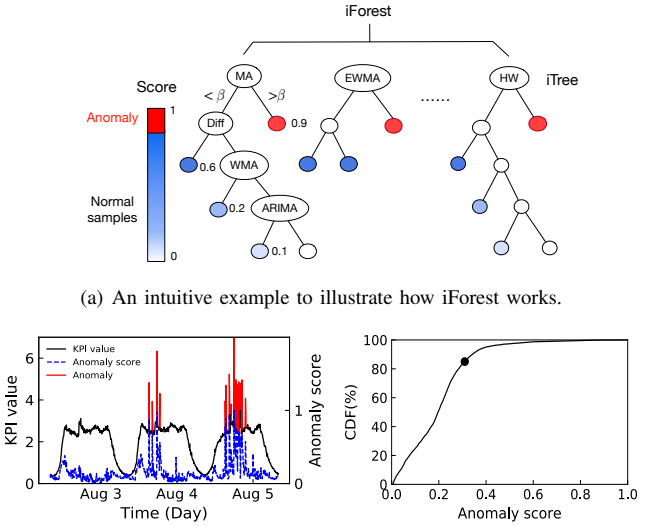
To label anomalies consistently for the given KPIs, an operator first labels a true candidate anomaly as template. Then the second component, *Anomaly Similarity Search*, uses the template to automatically search for the top-k similar anomalies from the candidate anomalies. We propose an *accelerated DTW* method to reduce the search response time. The operator then confirms the returned similar anomalies and labels them accordingly. This process continues until there are no more unlabeled anomalies. More details on how operators interact with *Label-Less* are introduced in §VI.

III. Unsupervised Anomaly Detection

A. Review of Isolation Forest

A rich body of literature exist on unsupervised anomaly detection, *e.g.*, One-Class SVM [17], deep generative model like Variational Auto-encoder [5]. Their philosophy is to focus on normal patterns instead of detecting anomalies directly. Since the majority of data points in KPIs are normal, these models all first recognize normal regions in the original or some latent feature space, and then compute the anomaly score by measuring "how far" an observation is from the normal regions. However, these approaches suffer from either high computational complexity [5] or poor performance [17]. Isolation Forest (iForest) is a popular outlier detector and has shown good performance with a linear time complexity in outlier detection [16], but it has not been applied in KPI anomaly detection. Thus, we novelly apply iForest to unsupervised KPI anomaly detection.

Different from most existing approaches, iForest explicitly isolates anomalies rather than profiling normal instances. It takes advantage of two properties of anomalies: they are the



(a) An intuitive example to illustrate how iForest works.

(b) Comparison of anomaly scores between normal points and anomalies.

(c) The CDF of anomaly scores. The black point is 85^{th} percentile.

Fig. 3: Detailed explanation of iForest and anomaly score.

minority consisting of fewer samples and they have some features which are very different from those of normal samples. In other words, anomalies are "few and different". Therefore, they are more susceptible to be isolated.

IForest isolates observations by randomly selecting a feature and a split value between the minimum and maximum values of the selected feature. Usually, only a few conditions are needed to isolate anomalies from normal ones, whereas separating normal observations requires more conditions. Thus an anomaly score can be calculated as the number of conditions required to separate a given observation. Recursive partitioning of iForest can be represented by a tree structure (iTree), and the number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating node. Therefore, iForest builds an ensemble of iTrees for the input data, and anomalies are those instances with short average path lengths on the iTrees. Fig. 3(a) presents an intuitive illustration and the features on every node will be introduced later.

B. Preprocessing

Before unsupervised anomaly detection, we first need to preprocess the input unlabeled KPIs. KPIs are monitored with certain interval, *e.g.*, every minute. Occasionally, a monitoring system does not receive data, leading to missing values. We simply use linear interpolation to fill them based on their adjacent data points. In addition, since KPIs are sourced from different servers, we normalize them to eliminate scale variances to prepare for similarity search. Zero-mean normalization is applied to transform a KPI to have zero mean and one standard deviation, *i.e.*, $\hat{x_i} = \frac{x_i - \mu_x}{\sigma_x}$, where x_i is the raw KPI value, μ_x and σ_x are its mean and standard deviation.

C. Feature Extraction

To apply iForest algorithm, we first need to extract anomaly features. Inspired by the idea of ensemble learning [2] [3],

we adopt several classical time series prediction models as feature extractors, *i.e.*, Difference [2], Moving Average (MA), Weighed MA (WMA), Exponentially Weighted MA (EWMA), Autoregressive Integrated MA (ARIMA) [7] and Holt-Winters [6]. The parameters are chosen based on our experience and domain knowledge. These six models have low computation complexity and good performance which have been proved in anomaly detection literature [2] [3]. Certainly, if needed, other suitable prediction models can be added or replace these six models.

In general, normal points can be well predicted with small prediction errors, since they conform to the expected behavior, while anomalies with unexpected patterns are hard to be predicted, creating large prediction errors. Let p_i be the predicted value of data point x_i, then we can calculate the absolute difference between the actual value and the forecast value of each data point, *i.e.*, $|p_i - x_i|$, as a feature of data point x_i.

D. Unsupervised Anomaly Detector

In previous step, the raw KPI vector X ($n \times 1$) was transformed into feature matrix X' ($n \times 6$). Then we recursively divide X' by randomly selecting a feature, *e.g.*, MA, and a split value ϵ, as shown in Fig. 3(a), to construct an iForest. Each terminal node in iForest is associated with a score between 0 and 1, calculated based on its path length [16]. The larger the score is, the more likely the node is an anomaly. Fig. 3(b) presents an example to illustrate the anomaly score (right Y-axis) from a real KPI (left Y-axis). Obviously, the anomaly scores of anomalies are much higher than normal points. The effectiveness of iForest will be studied in §V-C.

E. Threshold Selection

To detect potential anomalies based on anomaly scores, we need to choose an appropriate threshold θ. If the anomaly score of a point x_i is larger than θ, we regard this point as a potential anomalous point, and then add $segment(i)$ into the candidate anomalies. Usually, threshold selection for anomaly detection needs to trade off between high recall and high precision, and often uses F-score as the metric. However, since here we try to generate candidate anomalies for operators and as our search space for *anomaly similarity search*, we choose to bias towards high recall and acceptable precision, so that false negative rate can be minimized. In principle, the percentage of anomalous points in a KPI is very rare, usually about 1% [3], [5]. Fig. 3(c) shows the CDF of anomaly scores of a real KPI. We observe that the majority of points are likely to be normal with low anomaly scores and only few points have high anomaly scores. Thus we choose anomaly score at 85^{th} percentile (marked as the black dot) as the threshold θ such that the chance of false negatives is really small. The robustness and generality of our strategy of threshold selection will be discussed in §V-C.

IV. Anomaly Similarity Search

A. Overview

Through unsupervised anomaly detection, we can get a set of candidate anomalies as our search space. Next we search for the top-k anomalies most similar to a user-provided anomaly template and report to operators. The key issue here is how to find similar anomalies with high accuracy and low response latency. The design of anomaly similarity search is shown in Algorithm 1 and details will be introduced below.

Algorithm 1: Anomaly Similarity Search

Input: Anomaly template q, candidate anomalies C
Output: Top-k similar anomalies

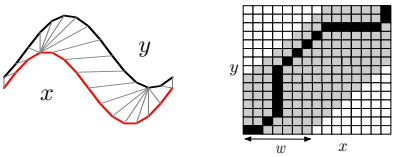
```
Create a max-heap h with size k to keep the top-k similar anomalies;
for each candidate anomaly c in candidate C do
    best-so-far=the root node of h;
    if LB_Kim(q, c) > best-so-far then
        prune off c;
    else if LB_Keogh(q, c) > best-so-far then
        prune off c;
    else if LB_Keogh_Reverse(q, c) > best-so-far then
        prune off c;
    else
        for j = 1; j ≤ m; j++ do
            dist = cDTW(1 : j) + LB_Keogh(j : m);
            if dist > best-so-far then
                early stopping and prune off c;
        h.push([c,dist]);
return Top-k similar anomalies;
```

B. DTW as Similarity Measure

Choosing a suitable distance measure is the first step for anomaly similarity search. In spite of dozens of alternative measures, recent empirical evidence strongly suggests that none of these alternatives routinely beats Dynamic Time Warping (DTW) [23] [20]. Thus we adopt DTW and the effectiveness of DTW will be discussed in §V-E. By optimally aligning two sequences in temporal domain, DTW calculates the minimized accumulated aligning cost as distance with dynamic programming. Hence, DTW can handle shift invariance as shown in Fig. 4(a). Given two time series x and y, with length of s and t, respectively, it first estimates the distance map d between two points x_i, y_j given by $d(i,j) = (x_i - y_j)^2$. A cumulative distance D with elements $D_{i,j}$ is computed from the distance map d as: $D_{i,j} = d(i,j) + \min\{D_{i,j-1}, D_{i-1,j}, D_{i-1,j-1}\}$. The final DTW corresponds to the total accumulated cost, *i.e.*, $d_{DTW}(x,y) = D_{s,t}$. The time complexity of DTW is $O(st)$. Fig. 4(b) shows the computation of DTW with a matrix, and the warping path representing the optimal alignment is marked in black color.

C. Accelerated DTW

Suppose there are r candidate anomalies, the complexity of similarity search reaches $O(rm^2)$, where m is the length of anomaly template. When m and r are large, DTW computation can be very costly. Thus speeding up DTW is a key challenge



(a) Elastic alignments. (b) Warping path.

Fig. 4: Illustration of DTW.

in our scenario. A few techniques have been proposed to speed up DTW in the literature, including early stopping [20], lower bound [24], local constraints [22], Piecewise Aggregate Approximation (PAA) [22], *etc*. Note that not all techniques work effectively in our scenario, for example, PAA only performs well on long time series about thousands of points. While the length of anomaly template is relative short with only dozens of points, PAA will greatly destroy raw information. After careful investigation and extensive experiments, we design an *accelerated DTW* approach leveraging constrained DTW, lower bound, and early stopping, as shown in Algorithm 1.

Constrained DTW. The core idea of constrained DTW (cDTW) is to limit the permissible warping paths by providing local restrictions on the set of alternative steps considered [20]. It works under the assumption that it is unlikely for q_i and c_j to be matched if i and j are too far apart. The threshold is determined by a warping window size w. In this way, we align q_i and c_j only if $|i-j| \leq w$. When $w = 0$, cDTW becomes the Euclidean distance, whereas when $w \geq m$, cDTW has no local constraint. The computational complexity of cDTW is $O(wm)$. As shown in Fig. 4(b), the gray area is the permissible scope of warping path.

Lower bound. The basic idea of lower bound is to use a cheap-to-compute lower bound of DTW to prune segments that cannot possibly be the top-k similar anomalies [20]. A defined lower bound of DTW should satisfy $LB(q,c) < DTW(q,c)$. The process of pruning by lower bound has been shown in Algorithm 1, where *best-so-far* is the furthest distance in top-k segments. In order to maintain *best-so-far* efficiently, we create a max-heap to keep the top-k similar anomalies. The root node of the max-heap is the value of *best-so-far*. When computing similarity, once the lower bound exceeds *best-so-far*, this segment is discarded since it cannot be in the top-k list. We adopt the following three lower bounds in *Label-Less*.

LB_{Kim} [24] extracts a four-tuple feature vector from each segment: the first, last, minimum, and maximum values. LB_{Kim} is the maximum squared difference of these features.

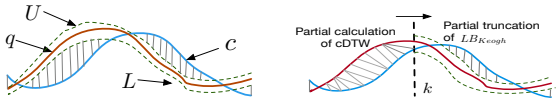
LB_{Keogh} [22]. We will use the warping window size w in cDTW to define two new sequences, U and L:

$$U_i = \max_j q_j \quad L_i = \min_j q_j, \quad \text{where } j \in [i-w, i+w]$$

U and L are the upper bound and lower bound of the template q, *i.e.*, $\forall_i L_i \leq q_i \leq U_i$. Then LB_{Keogh} can be computed as:

$$LB_{Keogh}(q,c) = \sum_{i=1}^{m} (c_i - U_i)^2 I(c_i > U_i) + (c_i - L_i)^2 I(c_i < L_i)$$

$I(\cdot)$ is the indicator function. As shown in Fig. 5(a), the squared length of gray lines represents the LB_{Keogh}.



(a) LB_{Keogh}. (b) Early stopping.

Fig. 5: Illustration of LB_{Keogh} and early stopping of DTW.

Reversing the q/c role in LB_{Keogh} [20]. Usually, LB_{Keogh} described above creates an envelope around the anomaly template q. We can also have the upper bound U and lower bound L around each candidate anomaly c so as to obtain another lower bound of DTW.

Early stopping. Fig. 5(b) explains the idea of early stopping. In the process of computing cDTW with dynamic programming, the black/dashed line moves from left to right. If we add the partial (incrementally calculated) cDTW contribution from the left side of the dashed line and the LB_{Keogh} contribution from the right of dashed line, this can be a new lower bound to DTW(q,c). Once the sum of partial DTW and partial LB_{Keogh} exceeds *best-so-far*, we can admissibly stop the calculation and prune off this candidate anomaly c.

V. Evaluation

In this part, we focus on evaluating two components in *Label-Less*. We first introduce the datasets and metric. Then, we present the performance of unsupervised anomaly detection and anomaly similarity search. In addition, the discussion about search response time is also provided.

A. Datasets

We obtained four datasets named $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$ (shown in Table I), , with 30 KPIs in total collected from a large Internet company through the real-time monitoring systems. All of KPIs have a time span of about six months with 1-minute monitoring interval. KPIs in each dataset monitor a service on different servers, so they have similar shapes and many similar anomalies. To illustrate our datasets intuitively, we select one KPI from each dataset and plot a 2-day-long fragment as shown in Fig. 6.

For the purpose of evaluating unsupervised anomaly detection, the experienced operators first label all anomaly segments as the ground truth (N'). Then in order to evaluate anomaly similarity search, operators pick the most frequent anomaly pattern in each dataset as the template, which is marked in blue dashed rectangle in Fig. 6 (other anomaly patterns can be seen in Fig. 11), and identify all anomalies in this dataset that are similar to the template as the ground truth of anomaly similarity search (N). For example, dataset $\mathcal{A}$ has 1030 labeled anomalies in total and 874 similar anomalies for the most frequent anomaly pattern A_1.

B. Performance Metric

Different from regular anomaly detection, the input and output are segments, instead of single observation. A returned segment may partially overlap with a true anomaly segment.

TABLE I: Details of the datasets in our experiments.

Datasets	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$
# KPIs	8	12	5	5
Template in Fig. 11	A_1	B_1	C_1	D_1
Length of template (m)	10	8	13	15
#Similar anomalies (N)	874	986	1308	634
#Total anomalies (N')	1030	1180	1436	752

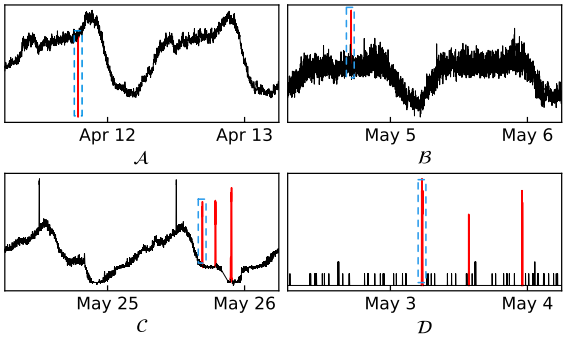


Fig. 6: 2-day-long fragments of the KPIs in our datasets. Similar anomalies are labeled in red color and the anomaly templates are marked in blue dashed rectangles.

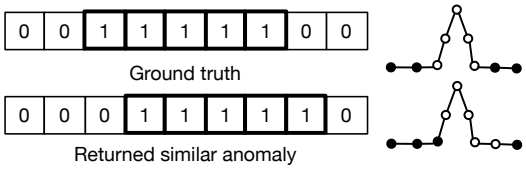


Fig. 7: Illustration of true positive. Anomaly is denoted by 1 and marked in hollow point on the right. The first row is the ground truth with an anomaly segment highlighted in shaded squares. The returned similar anomaly shown in the second row has a more than 50% overlap with labeled anomaly in truth. Hence, we regard it is a true positive.

TABLE II: Comparison of recall and per-KPI running time with different outlier detectors.

Datasets		$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$
Recall(%)	**IForest**	**97.4**	**98.9**	**95.9**	**99.4**
	LOF	66.7	49.7	62.8	68.0
	OCSVM	75.9	73.3	82.9	76.3
Per-KPI running time	**IForest**	**13.6s**	**9.3s**	**12.4s**	**12.4s**
	LOF	14.1s	11.5s	11.5s	11.9s
	OCSVM	5.7h	6.4h	5.9h	6.3h

As Fig. 7 shows, we define that if a returned anomaly segment overlaps with more than half of a labeled anomaly segment in ground truth, then we regard it is a true positive.

As for the metric of unsupervised anomaly detection, since it provides potential anomalies for operators, we mainly emphasize on high recall, which is computed as $\frac{\#TP}{N'}$, where N' is the number of total labeled anomaly segments and $\#TP$ is the number of true positives found by candidate anomalies.

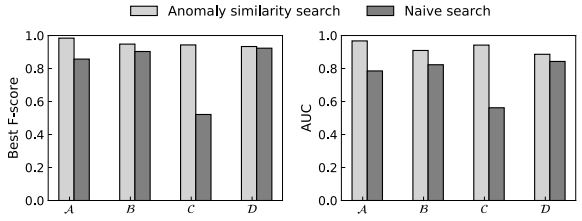
Top-k precision and top-k recall are two popular metrics for k-NN search, which can be computed as: $precision = \frac{\#TP}{k}$ and $recall = \frac{\#TP}{N}$, where N is the number of true similar anomalies, and $\#TP$ is the number of true positives found. Besides, F-score as the harmonic mean of precision and recall can be computed as: $F\text{-}score = \frac{2 \times precision \times recall}{precision + recall}$. However, the selection of k has a great impact on precision and recall. When k is too small, many similar anomalies cannot be reported, resulting in low recall and high precision. When k is too large, segments with low similarity may will be reported, leading to high recall and low precision. In practice, k can be determined by operators based on their needs and domain knowledge. Considering the best k for each dataset may be different, a more general approach for evaluation is to focus on precision-recall curve and compute the AUC (area under curve), which is the average precision over recall given all possible k, or compute best F-score indicating the best possible performance given an optimal k. Therefore, we adopt best F-score and AUC for evaluating anomaly similarity search.

C. Performance of IForest

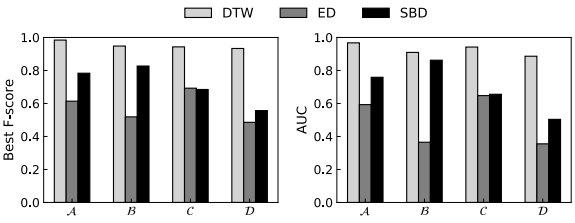
In order to show the effectiveness of unsupervised anomaly detection in *Label-Less*, we compare iForest with two popular outlier detectors, *i.e.*, One-Class SVM (OCSVM) [17] and Local outlier factor (LOF) [18]. OCSVM obtains a spherical boundary in feature space around the data. The volume of this hypersphere is minimized so as to isolate outliers outside the boundary. LOF aims to find anomalous data points by measuring the local deviation of a given data point with respect to its neighbors. These three outlier detectors adopt same features (§III-C) and threshold selection strategy (§III-E). The results about recall and per-KPI running time are shown in Table II. It is obvious that both LOF and OCSVM show poor recall, and OCSVM has extremely high computation complexity. The high recall of iForest also demonstrates that the features extracted by time series models is effective and threshold selection strategy is robust. Note that it is always possible to have false negatives, operators can adjust θ to control false negatives within an acceptable range.

D. Comparison with Naive Search

The search space of anomaly similarity search in *Label-Less* is candidate anomalies given by unsupervised anomaly detection. In order to demonstrate the importance of unsupervised anomaly detection on anomaly similarity search, we compare it with the *naive search* whose search space is all segments with length m in KPIs. The result is displayed in Fig. 8(a). We observe that the best F-score of anomaly similarity search reaches about 0.95 on average, higher than naive search in all datasets. This is because that naive search simply reports all similar segments without considering whether they are normal or anomalous. However, some similar segments are normal patterns. For example, for dataset $\mathcal{C}$ as shown in Fig. 6, there is a spike (black one) at about 12:00 every day. Although similar to the template, these spikes are normal periodical behaviors, probably due to regular business promotions at that time. Naive search will output them simply based on similarity, leading to



(a) Comparison between anomaly similarity search and naive search.



(b) Comparison with different distance measures.

Fig. 8: Performance of anomaly similarity search.

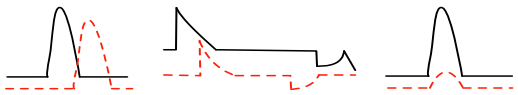
poor performance (false positive), while unsupervised anomaly detection can filter out these normal points.

On the one hand, this comparison clearly demonstrates that it is essential to adopt unsupervised anomaly detection before anomaly similarity search, since it not only filters out normal points to reduce false positives, but also reduces the search space and improves search efficiency significantly (§V-F). On the other hand, the result also highlights the difference between anomaly similarity search and general time series similarity search (§II-B).

E. Comparison with Other Similarity Measures

To demonstrate the effectiveness of accelerated DTW (DTW for short hereinafter) in *Label-Less*, we use Euclidean Distance (ED), and Shape-Based Distance (SBD) [25] as similarity measure to compare with DTW, and other modules remain unchanged. ED is a simple and well-accepted similarity measure, computed as $ED(q,c) = \sqrt{\sum_{i=1}^{m}(q_i - c_i)^2}$. SBD is a normalized version of cross-correlation, which can handle distortions in amplitude and phase.

The performance comparison is shown in Fig. 8(b). Clearly, DTW consistently outperforms other distance measures on all datasets. We can observe that ED perform poorly, since it cannot handle shift or local scaling problem as shown in Fig. 9(a) and Fig. 9(b). When calculating the distance between two segments, ED assumes the i-th point in one segment is aligned with the i-th point in the other, and it is very sensitive to small distortions in the time axis. The performance of SBD is unstable across the datasets. Although SBD can recognize time shift, it fails to deal with local scaling like in Fig. 9(b). In addition, it focuses on shape regardless of variance in amplitude [25]. For example, a sharp spike and a gradual increase in Fig. 9(c) are considered to have a small distance. However, amplitude variance cannot be ignored in anomaly detection, since a sharp spike is likely an anomaly while a gradual increase is probably not, but SBD misjudges



(a) Shift (b) Local scaling (c) Amplitude

Fig. 9: Analysis of distance measures. Euclidean distance fails to tackle (a) and (b); SBD fails to tackle (b) and (c).

TABLE III: Comparison of per-KPI running time (seconds).

Datasets	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$
Anomaly similarity search	**0.38**	**0.35**	**0.43**	**0.39**
Naive search	2.18	2.22	2.95	2.60
Original DTW	3.84	3.99	5.29	4.95

this case. This experiment concludes that DTW is the best choice for anomaly similarity search among these alternatives.

F. Efficiency of Anomaly Similarity Search

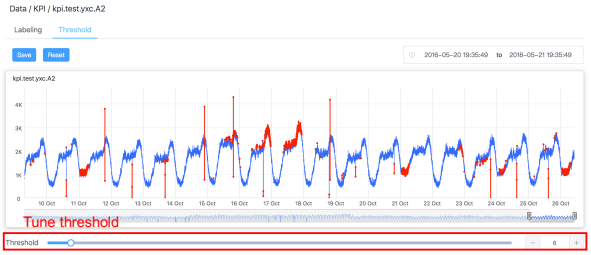
We have stated that response time is a big concern due to a large number of long KPIs in §I. By comparing with *naive search* (without unsupervised anomaly detection), and *original DTW* (with $O(m^2)$ time complexity and without any techniques in §IV-C), we demonstrate the efficiency of anomaly similarity search in *Label-Less*. The results are shown in Table III and the response time of per-KPI anomaly similarity search in *Label-Less* is under 0.5 second. In addition, the results show that both unsupervised anomaly detection and accelerated DTW play key roles in reducing response time. The unsupervised anomaly detection can reduce search space significantly, and the accelerated techniques in §IV-C are extremely effective in reducing DTW computation complexity.

VI. Overall Labeling Workload

In this section, we mainly focus on evaluating the performance of *Label-Less* on reducing labeling workload. We first introduce the traditional labeling method. Then we describe the new labeling method with *Label-Less* and demonstrate the good performance of *Label-Less* by comparing the total labeling time with traditional labeling.

A. Traditional Labeling Method

In traditional labeling, given an unlabeled KPI, an operator first needs to manually scan the entire KPI to understand its overall normal behaviors and get a preliminary judgment of what can be considered as anomalies. After that, the operator is ready to label from the beginning of the KPI. When finding an anomaly case, instead of hurrying to make a decision, he will go back to see the historical information (last day/week/month) and make a comprehensive comparison to draw a reasonable conclusion whether the segment needs to be labeled. If the segment needs to be labeled as an anomaly, the operator will zoom in to the region where the anomaly is located and label the segment point by point, and then zoom out to the original time scale. The operator will repeat above process until all anomalies have been labeled. Besides, above labeling process is based on using a labeling tool with graphical interface.



(a) Interface of candidate potential anomalies (labeled in red) given by unsupervised anomaly detection.

(b) Interface of anomaly similarity search. On the left is the anomaly template labeled in pink band; on the right is the similar anomalies given by *Label-Less* sorted by similarity.

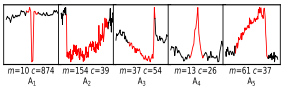
Fig. 10: Interface of labeling tool with *Label-Less*.

In summary, the majority of traditional labeling efforts are spent on visually scanning through KPIs to gauge "what is an anomaly" and examining KPIs back and forth to check whether "similar" segments are labeled consistently. When faced with a large number of long KPIs, traditional labeling is indeed a prohibitive task for operators.

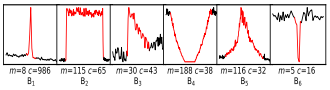
B. Labeling with Label-Less

Label-Less has been integrated into a labeling tool developed by us. Fig. 10(a) shows the interface of candidate potential anomalies given by unsupervised anomaly detection (labeled in red), and Fig. 10(b) shows the interface of anomaly similarity search. The improved labeling process with *Label-Less* is semi-automated, as shown in Fig. 2. More specifically, given a set of unlabeled KPIs, candidate anomalies on each KPI are identified through the unsupervised anomaly detection. Thanks to the high recall achieved by this component, an operator does not need to examine the entire KPI, but only the candidate anomalies. Besides, the operator can drag the slider to tune a suitable threshold. Later, the operator first labels one of the candidate segments as an anomaly (pink band on the left of Fig. 10(b)), then *Label-Less* uses it as a template to automatically search for similar anomalies (displayed on the right of Fig. 10(b) sorted by similarity). For the output similar segments, the operator can look though them one by one quickly. If top-k segments are all true positives, the operator can click the "submit" button, so that top-k segments can be labeled automatically. If there is a false positive, the operator can click the "wrong" button to ignore the segment. In this way, the operator saves time significantly in finding where the anomalies are and manually labeling. The only manual work for operators is choosing a template and checking the results reported by *Label-Less*.

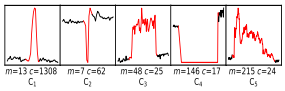
In order to show the effectiveness of *Label-Less* intuitively, we invite eight voluntary experienced operators from an Internet company which are divided into two groups to participate



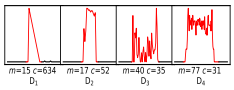
(a) Dataset $\mathcal{A}$.



(b) Dataset $\mathcal{B}$.



(c) Dataset $\mathcal{C}$.



(d) Dataset $\mathcal{D}$.

Fig. 11: All templates in four datasets, where m is the length of template and c is the number of similar anomalies.

TABLE IV: Comparison of labeling time (minutes) between traditional labeling and *Label-Less*.

Datasets		$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$
Total labeling time	***Label-Less***	**91**	**114**	**56**	**52**
	Traditional	980	1243	573	602
Per-KPI labeling time	***Label-Less***	**9.1**	**9.5**	**11.2**	**10.4**
	Traditional	98.0	103.6	114.6	120.4

in our experiments. Every operator in each group chooses one dataset to label. The operators in the first group use our developed labeling tool without *Label-Less* to measure the workload of traditional labeling and record the total time of labeling these four datasets. The other four operators in the second group use the labeling tool with *Label-Less* plugged in to measure the total labeling time. Fig. 11 shows the templates used by operators in the second group in four datasets, *e.g.*, dataset $\mathcal{A}$ has five different anomaly templates and 1030 anomaly segments in total. In traditional labeling, operators need to find and label all the 1030 segments by hand and examine carefully. With *Label-Less*, operators choose the first template A_1 in Fig. 11(a), then confirm all results provided by *Label-Less* until all anomalies similar to A_1 have been labeled. Next, the operators will probably choose A_2 iteratively until all anomalies in dataset $\mathcal{A}$ have been labeled. It is obvious that confirming thousands of segments is much faster than finding and labeling thousands of segments manually.

Table IV shows the workload comparison between *Label-Less* and traditional labeling in detail. It is obvious that it takes one or two hours to label a 6-month-long KPI for traditional manually labeling. However, it only needs about ten minutes with *Label-Less*. Thus we have confidence to make a conclusion that *Label-Less* can reduce more than 90% labeling time for operators.

VII. Related Work and Discussion

A. Related Work

Labeling overhead: Despite many efforts have been put into anomaly detection [2]–[10], effective ways to reduce the labeling overhead of labeling KPI anomalies have remained elusive. In the past, Haakon Ringberg *et al.* have argued the challenges of manual labeling traces for network traffic anomaly detection and presented WebClass [26], a web-based infrastructure that adds rigor to the manual labeling process,

which allows researchers to share, inspect, and label traffic time series through a graphical user interface, but it still requires operators to manually label and inspect. In this paper, we aim to facilitate labeling by automatic tools driven by algorithms. To the best of our knowledge, *Label-Less* is the first semi-automatic labeling tool that focuses on reducing the labeling overhead of KPI anomaly dataset by algorithm.

Anomaly similarity search: Although this paper is also the first one that studies anomaly similarity search in anomaly detection fields, the problem can be regarded as incorporating the domain knowledge about anomaly detection into general time series similarity search [20] [21]. Besides, the techniques adopted in accelerated DTW are also inspired from general time series similarity search. The difference between this problem and naive similarity search has been introduced in §II-B and demonstrated in §V-D.

B. Discussion

Beyond reducing the labeling overhead, we name a couple of applications of the anomaly similarity search. First, it can assist operators in discovering some specific types of anomalies which they are concerned about. For example, the template for dataset $\mathcal{A}$ in Fig. 6 indicates a sharp dip in the number of requests. Finding all similar dips can help operators timely troubleshoot whether and why such dips happen frequently, so that they can respond quickly to avoid the decline in the number of requests and reduce potential economic loses. Second, suppose the template for dataset $\mathcal{B}$ in Fig. 6 is of previously unknown type, operators can use anomaly similarity search to quickly find similar unknown anomalies in the all KPIs in dataset $\mathcal{B}$ to help pinpoint the potential root cause as soon as possible.

VIII. Conclusion

In this paper, we argue that labeling overhead has become the main hurdle to researching effective and practical KPI anomaly detection. Equipped with our novel technique *robust and rapid anomaly similarity search*, our proposed semi-automatic labeling tool *Label-Less* saves the operators from scanning and checking the long KPIs back and forth for abnormal patterns or label consistency. Our evaluations using 30 real KPIs show that anomaly similarity search achieves the best F-score of 0.95 on average, and a real-time response time (less than 0.5 second). Overall, *Label-Less* developed in practice can reduce operators' labeling overhead by more than 90%.

We believe that, *Label-Less* is important first step to enable an ImageNet-like large-scale KPI anomaly dataset with high-quality ground truth. This is greatly beneficial to KPI anomaly detection in academia and industry just like what ImageNet did to computer vision and deep learning.

IX. Acknowledgement

We thank the anonymous reviewers for their valuable feedbacks. We thank Juexing Liao, Minghua Ma and Ya Su for their helpful suggestions and proofreading. This work has been supported by the National Natural Science Foundation of China (NSFC) under grant 61472214, 61472210, the Beijing National Research Center for Information Science and Technology (BNRist) key projects, the Global Talent Recruitment (Youth) Program and Okawa Research Grant.

References

[1] Y. Chen, R. Mahajan, B. Sridharan, and Z.-L. Zhang, "A provider-side view of web search response time," in *SIGCOMM*, ACM, 2013.
[2] D. Liu, Y. Zhao, H. Xu, Y. Sun, D. Pei, J. Luo, and et.al, "Opprentice: towards practical and automatic anomaly detection through machine learning," in *IMC*, ACM, 2015.
[3] N. Laptev, S. Amizadeh, and I. Flint, "Generic and scalable framework for automated time-series anomaly detection," in *SIGKDD*, ACM, 2015.
[4] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark," in *ICMLA*, IEEE, 2015.
[5] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, and et.al, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *WWW*, 2018.
[6] H. Yan, A. Flavel, Z. Ge, and A. Gerber, "Argus: End-to-end service anomaly detection and localization from an isp's point of view," in *INFOCOM*, IEEE, 2012.
[7] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: Methods, evaluation, and applications," in *IMC*, ACM, 2003.
[8] S. Zhang, Y. Liu, D. Pei, Y. Chen, X. Qu, S. Tao, and Z. Zang, "Rapid and robust impact assessment of software changes in large internet-based services," in *CoNEXT*, IEEE, 2015.
[9] M. Ma, S. Zhang, D. Pei, X. Huang, and H. Dai, "Robust and rapid adaption for concept drift in software system anomaly detection," in *ISSRE*, IEEE, 2018.
[10] S. Zhang, Y. Liu, D. Pei, Y. Chen, X. Qu, S. Tao, Z. Zang, X. Jing, and M. Feng, "Funnel: Assessing software changes in web-based services," *IEEE Transactions on Service Computing*, 2016.
[11] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The ucr time series classification archive," July 2015. www.cs.ucr.edu/~eamonn/time_series_data/.
[12] M. Lichman, "UCI machine learning repository," 2013.
[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, IEEE, 2009.
[14] "Curve: Time Series Labeling Tool." https://github.com/baidu/Curve.
[15] "AIOps Challenge." http://iops.ai/.
[16] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *ICDM*, IEEE, 2008.
[17] M. Amer, M. Goldstein, and et.al, "Enhancing one-class support vector machines for unsupervised anomaly detection," in *SIGKDD*, ACM, 2013.
[18] M. M. Breunig, H.-P. Kriegel, and et.al, "Lof: Identifying density-based local outliers," in *SIGMOD*, ACM, 2000.
[19] Y. Sun, Y. Zhao, Y. Su, D. Liu, X. Nie, Y. Meng, S. Cheng, D. Pei, S. Zhang, *et al.*, "Hotspot: Anomaly localization for additive kpis with multi-dimensional attributes," *IEEE Access*, 2018.
[20] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, and et al., "Searching and mining trillions of time series subsequences under dynamic time warping," in *SIGKDD*, ACM, 2012.
[21] W.-S. Han, J. Lee, Y.-S. Moon, and H. Jiang, "Ranked subsequence matching in time-series databases," in *VLDB*, VLDB Endowment, 2007.
[22] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and Information Systems*, 2005.
[23] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series.," in *KDD workshop*, ACM, 1994.
[24] S.-W. Kim, S. Park, and W. W. Chu, "An index-based approach for similarity search supporting time warping in large sequence databases," in *ICDE*, IEEE, 2001.
[25] J. Paparrizos and L. Gravano, "k-shape: Efficient and accurate clustering of time series," in *SIGMOD*, ACM, 2015.
[26] H. Ringberg, A. Soule, and J. Rexford, "Webclass: adding rigor to manual labeling of traffic anomalies," in *SIGCOMM*, ACM, 2008.