Proceedings of Machine Learning Research 71:56–65, 2017 KDD 2017: Workshop on Anomaly Detection in Finance

Real-time anomaly detection system for time series at scale

Meir Toledano, Ira Cohen, Yonatan Ben-Simhon, Inbal Tadeski
{MEIR, IRA, YONATAN, INBAL}@ANODOT.COM
Anodot Ltd., HaTidhar 16, Ra'anana, Israel

Abstract

This paper describes the design considerations and general outline of an anomaly detection system used by Anodot. We present results of the system on a large set of metrics collected from multiple companies.

Keywords: Anomaly detection, time series modeling, high scalability, seasonality detection

1. Introduction

A challenge, for both machines and humans, is identifying an anomaly. Very often the problem is ill-posed, making it hard to tell what an anomaly is. Fortunately, many metrics from online systems are expressed in time series signals. In time series signals, an anomaly is any unexpected change in a pattern in one or more of the signals.

There are obvious things that most people agree on of what it means to be the same, and that's true for time series signals as well, especially in the online and financial world where companies measure number of users, transaction volume and value, number of clicks, revenue numbers and other similar metrics. People do have a notion of what "normal" is, and they understand it because, in the end, it affects revenue. By understanding "normal", they also understand when something is "abnormal". Thus, knowing what an anomaly is isn't completely philosophical or abstract.

Anomaly detection has been an active research area in the fields of statistics and machine learning. Temporal methods, going all the way back to Holt-Winters Chatfield (1978), classical ARIMA and seasonal ARIMA models Chatfield (2016), clustering techniques for discovering outliers in time series and other types of data, and more, have been researched and suggested over the years Krishnamurthy et al. (2003)Chandola et al. (2009). But, when building a production ready anomaly detection system, what are the main design principles? It depends on the needs and use case.

Based on our experience, there are some main design considerations when building an automated anomaly detection system:

1. Timeliness: How quickly does the company need to an answer to determine if something is an anomaly or not? Does the determination need to be in real-time, or is it okay for the system to determine it was an anomaly after a day, week, month or year has already passed?
2. Scale: Does the system need to process hundreds of metrics, or millions? Will the datasets be on a large scale or a relatively small scale?
3. Rate of change - Does the data tend to change rapidly, or is the system being measured relatively static?

4. Conciseness: If there are a lot of different metrics being measured, must the system produce an answer that tells the whole picture, or does it suffice to detect anomalies at each metric level by itself?
5. Robustness without supervision: Can the system rely on human feedback in some of the modeling choices (e.g., choice of algorithms and/or parameters for data streams), or does it have to be produce robust results in a completely unsupervised settings?

For most use cases related to online businesses in general, and finance in particular, the requirements from an anomaly detection system is to produce near real time detection at very large scale, while producing concise anomalies and being able to adapt to normal changes in data behavior. Given the amount of signals such businesses produce, a system must work robustly with no human intervention.

In this paper we'll describe the key components of the Anodot anomaly detection system, a system which discovers anomalies today for about 50 different companies and over 120 million time series metrics reporting constantly. We'll explain the choices of algorithms and architecture required for such a system to meet the design principles outlined above. We'll show results on real data from companies in the financial areas, demonstrating detection of fraudulent bots, and other issues.

2. Time series modeling and anomaly detection

To meet all of the requirements stated above - robustly detecting anomalies in (near) real-time, at very large scale, while being adaptive to ever-changing data and producing concise anomalies, we built a learning system that follows the following five steps:

1. Universal Metrics collection: The ability to collect and analyze millions to billions of time series metrics in real time with zero manual configuration.
2. Normal behavior learning: The first step towards identifying anomalies is learning a model of what is considered normal and devising a statistical test to determine that data points are anomalous if they are not explained by the model
3. Abnormal behavior learning: Once an anomaly is discovered, it is important to understand both its significance and its type. A significant anomaly is one that is more "surprising" compared to past anomalies. Learning is required to learn the significance of each anomaly based on various attributes describing the anomaly.
4. Behavioral topology learning: To show concise anomalies, the relationships between the potentially millions of time series inputs needs to be learned. Learning the relationship can be achieved by observing various aspects of the metrics behavior and applying clustering algorithms to discover similarities of those behaviors.
5. Feedback-based learning: It is feasible to obtain small amounts of supervision in labeling anomalies to be true or false positives. These feedbacks are used to update the models learned in the previous steps.

3. Learning normal behavior

Learning the normal behavior of a time series X_t amounts to learning the temporal dynamics and distribution of the time series $P_{Normal}(X_t|X_{t-1}, ..., X_0)$ such that we can determine the likelihood of the data point X_{t+1} to have originated from $P_{Normal}(X_{t+1})$. There are several challenges in robustly learning the distribution for any time series (without any supervision).

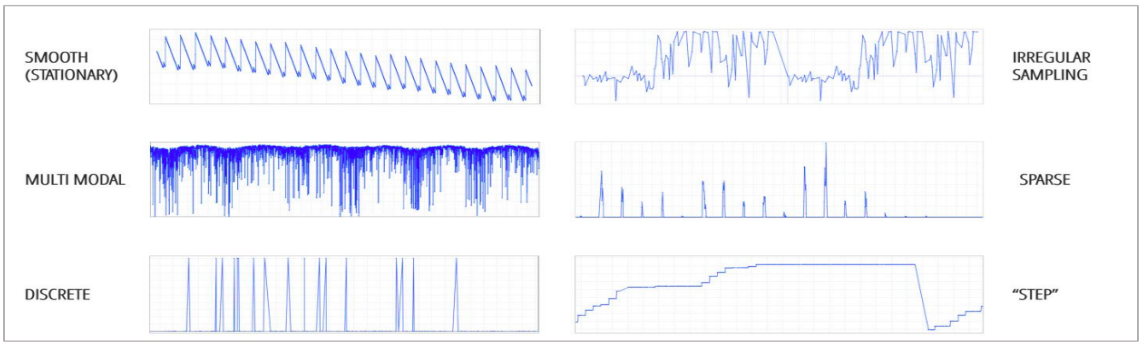


Figure 1: Example of six different types of time series signals

First one needs to choose the right parametric (or non-parametric) model before estimating P_{Normal}. Second, account for seasonal behavior that may or may not be present in the signal. Last, be able to adapt to changes in the behavior, even in the presence of anomalies.

3.1. Choosing the best model

There are a variety of models that capture and produce such a distribution, each fitting different types of time series signal behaviors. For example, time series models such as Holt-Winters, ARIMA models, and Hidden Markov Models, all capture temporal dynamics of a time series and produce a generative distribution for predicting the ranges of future values. But can these models be used for any type of signal? The answer is no - each model comes with its own assumptions, affecting what type of signal can be fitted well with it. For example, ARIMA and Holt-Winters models assume the the time series is ergodic, with a noise model that is Gaussian. A Markov model makes assumption on the memory of the process (current state depends only on previous state), and so on.

Based on about 120 million different time series signals, originating from a variety of businesses (from online financial services to manufacturing sensors), we discovered multiple types of behaviors that require separate types of models. Figure 1 demonstrates six main types of time series: Stationary "smooth", irregularly sampled, discrete valued, "sparse", "step", and multi-modal. While approximately 40% of metrics are stationary, many fall under the other types, requiring special models. For example, "sparse" metrics are signals that are mostly 0, but occasionally see a value different than 0. E.g., a signal counting the number of transactions related to a stock at each 5 minute window - may be 0 most of the time but non-zero values are only anomalies if larger than usual when the metric is not zero. Models such as Holt-Winters, would flag each non-zero value as an anomaly, producing many false positives. Similar examples demonstrate the need for a special model for each type.

The first step before learning a model of normality for a metric is therefore a classifier that decides for each metric, based on its history, what would be the best model for it.

Once we choose the model that would describe the metric the best, we must fit the best model parameters for it. Depending on the type of model, this step may take different forms - we use maximum likelihood estimation (MLE) for this step. While describing the maximum likelihood estimation method for each of the models is beyond the scope of this paper, there is a very important step that precedes the MLE, which is detecting whether the signal has seasonal patterns or not, and which seasonal patterns.

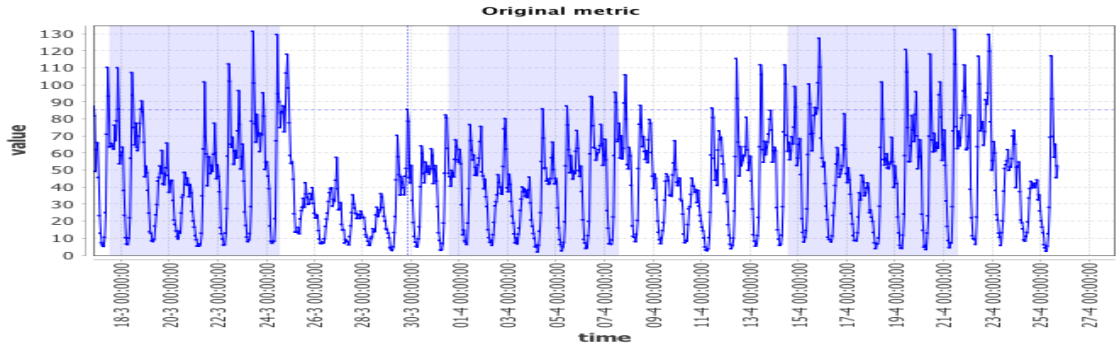


Figure 2: Example of periodic time series (with a weekly seasonal pattern)

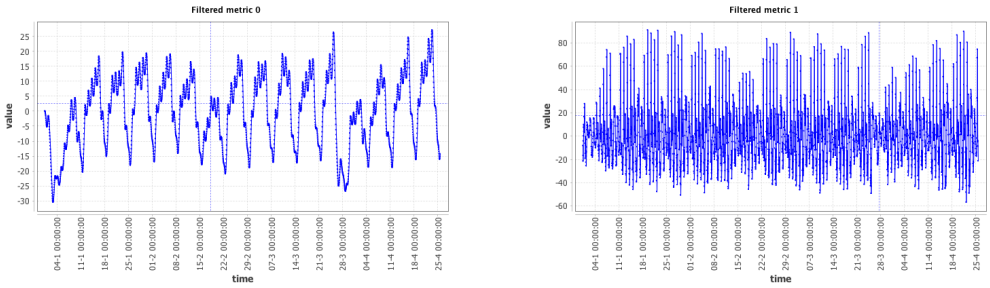
3.2. Seasonality detection

The purpose of seasonality detection is to automatically find the seasonal patterns present in the time series. In the context of large scale anomaly detection system, we are facing to several challenges:

1. In the context of business data, typical seasonal periods are linked to human activity : weekly, daily, hourly, etc. The range of possible seasons is large : from $10^3 s$ to $10^6 s$, which is three order of magnitude - this imposes a challenge to standard algorithms.
2. An algorithm must be robust to anomalies. One cannot assume that the anomalies are not present in the data used to detect seasonal patterns.
3. Multiple seasonal patterns in the time series, at low and high frequencies are challenging as high frequency seasonal patterns may prevent accurate detection of the lower frequency patterns (which we care about the most for time series modeling).
4. An algorithm must be efficient in CPU and memory, otherwise it cannot be applied to millions of metrics.
5. The algorithm must be robust to various types of unexpected conditions, such as missing data, pattern changes, etc.

3.2.1. Bank filter

To tackle the first challenge of having several order of magnitude of possible seasonal pattern periods (from minutes to weeks, and potentially more), we adopted a "Divide and Conquer" strategy on the spectrum. We split the series by applying three different band pass filters (three is for finding patterns from minutes to weeks). A Band pass filter operates on the Fourier space by selecting only a band of frequencies to be presents in the outputs, removing all other components. In our system we have three different filters conceived to extract the weekly band, the daily band and the hourly band. The number and the definition of bands are based on statistical analysis of our data-set. After applying the three filters on the times series, we have three different time series: Time series slow band, Time series medium band and Time series fast band. Our implementation uses the fifth order Butterworth filter. The effect of the filter bank can be seen in Fig. 3(a) for the medium band and Fig. 3(b).



(*a*) the slow band (*b*) the medium band

Figure 3: Effect of the filtering on the original series Fig. 2.

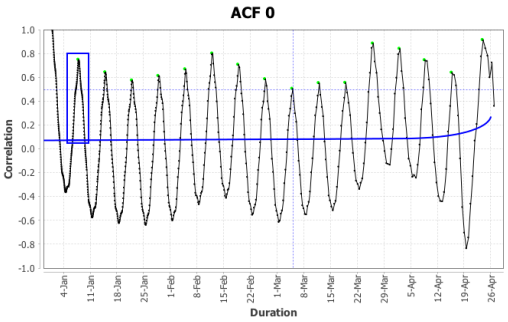


Figure 4: The ACF for the slow band, series Fig. 3(*a*).

3.2.2. Fast autocorrelation function

For each series produced by the filter bank we look for a prominent seasonal pattern, or determine that it does not contain one. For performing this task use the autocorrelation function (ACF) which is a function of the lag τ:

$$\rho(\tau) = \mathbb{C}or(X_t, X_{t+\tau}) = \frac{\mathbb{E}[(X_t - \mathbb{E}[X_t])(X_{t+\tau} - \mathbb{E}[X_{t+\tau}])]}{\sqrt{\mathbb{V}ar[X_t]\mathbb{V}ar[X_{t+\tau}]}} \tag{1}$$

The autocorrelation has several advantages for seasonality detection

1. The local maximums of the ACF are integer multiples of period of the series.
2. The ACF is a natural noise cancellation.

The effect of the ACF algorithm of the original series Fig 2 can be see Fig. 4 for the slow band. In the system, we are computing the ACF for each band.

The naive ACF algorithm is an $O(N^2)$ algorithm. Each computation of correlation coefficient $\rho(\tau)$ require N operations and there is N possible lags τ:

$$\tau_i = T_{sampling} i, i \in \mathbb{N}^*, \tau_i < T \tag{2}$$

This complexity prohibits computation at massive scale. To reduce complexity we use a sampling technique for faster computation of the ACF. The relative error of an ACF computation can be expressed as:

$$\epsilon_T = \frac{T_{sampling}}{T_{period}} \tag{3}$$

So, for a given a sampling interval of $T_{sampling}$, the larger the period, the more precise the estimate of the ACF coefficient will be. For example for a $T_{sampling} = 5$ min time series, the relative error of the estimate a one day period is $\pm 0.35\%$ but the same error became $\pm 8\%$ if the actual period is one hour. This means that the complexity of computation becomes higher in order to reach an extremely precise estimation as the period increases! In order to reduce the complexity, we set an additional constrain $\epsilon_T = C^{te} = c$. Solving this constrain mathematically gives us the explicit expression of τ:

$$\tau_i = \lfloor T_{sampling}(1+c)^i \rfloor, i \in \mathbb{N}, \tau_i < T \tag{4}$$

The sampling become now an exponential sampling.

By constraining in this way the computation, we only need $log(N)$ computation of the correlation coefficient - which is still an $O(N)$ algorithm - which means that the calculation of the overall ACF became an $O(Nlog(N))$ algorithm. This significant improvement make possible a massive scale use without compromising the quality of the period estimate.

3.2.3. Robust period extraction

For each computed ACF, the position of the local maximums are extracted $\{\tau_1, \tau_2, .., \tau_n\}$. A clustering of the inter-duration series, $\{\tau_2 - \tau_1, \tau_3 - \tau_3, .., \tau_n - \tau_{n-1}\}$, is performed. The dominant cluster give a very robust estimation of the period present in the series.

3.2.4. Relation to previous work and advantages of our method

Various methods have been suggested for the problem of *frequency detection* also known as *pitch detection* problem, they can be roughly split into two families: Fourier spectrum based methods and variance matrix eigen analysis.

The idea behind Fourier based methods is to compute the Fourier spectrum - which is operation with the FFT algorithm Welch (1967) and then extract local maximum of the spectrum. Some variants of the methods are computing the spectrum of the log-transformed data known as the cepstrum. Others are using parametric methods for estimating the spectrum by estimating an ARIMA model before. The methods is computationally fast, the FFT is need $O(N \log(N))$ floating point operation. But since the Fourier transform is a linear operator it propagate the noise and then the spectrum is noisy :

$$\mathcal{F}[signal + noise] = \mathcal{F}[signal] + \mathcal{F}[noise] \tag{5}$$

This make the identification of local maximum harder.

The second family is based on the analysis of the eigen values and eigen vectors of the variance matrix of the process. For stationary process the variance matrix is a Toepliz matrix build with the ACF vector. Historically the first method of this kind the Pisarenko's algorithm Pisarenko (1973). A notable variant is the MUSIC algorithm Schmidt (1986) which is often presented as an improvement and a generalization of the Pissarenko's method. Since the method is based on the ACF the noise - which is supposed to be independent of the signal - is automatically rejected :

$$\begin{aligned} ACF(signal + noise) &=^{1} ACF(signal) + ACF(noise) \\ &= ACF(signal) + 0 \\ &= ACF(signal) \end{aligned} \tag{6}$$

This is a huge advantage in practice. On the other hand both the computation of the ACF, which is a $O(N^2)$ algorithm and the diagonalization process - cubic complexity for SVD for example - are computationally expensive processes.

Our methods try to take advantages of the two strategies. It is an ACF based method so it is naturally robust to noise and it is a local maximum based method so it is fast. Besides there is an other advantage of using the ACF over Fourier spectrum in the context of frequency detection : The local maximums of Fourier spectrum are the fundamental f and the harmonics $2f, 4f, 8f...$ only the fundamental is an acceptable periods

$$x(t + 1/f) = x(t) \text{ BUT } x(t + 1/(2f)) \neq x(t)$$

On the other hand for the ACF all local maximums $T, 2T, 3T, ...$ are acceptable periods

$$x(t + T) = x(t) \text{ AND } x(t + 2T) = x(t)$$

This is a consequence of the group structure of periods. This means it is very costly to make a mistake in local maximums with FFT but it is almost free to make the same mistake with the ACF. In practical algorithms this is a huge advantage.

3.3. Adaptation

Typical time series signals collected from online services in general, and financial companies in particular, experience constant (normal) changes to their behavior. While some changes may be initially detected as anomalies, if they persist, the model describing the normal behavior should adapt to it. For example, a company stock price may be halved after a stock split, or its market cap may increase after an acquisition of another company - changes that must be adapted to by the normal model.

The need to constantly adapt, coupled with the need to scale the algorithms, require that online adaptive algorithms be used to change the normal behavior model. Most adaptive online algorithms have some notion of learning rate. For example, in the Holt-Winters model there are three coefficients accounting for level, trend and seasonal dynamics. The values of these coefficients determine how slow or fast the model adapts to every new data point.

Updating the parameters can be done using maximum likelihood estimation, but care should be taken in the face of anomalies. Anomalies consist of consecutive data points that are not explained by the normal model - however, if used as usual to update the model, the model may adapt too quickly to anomalies and miss future anomalies. However, if not used at all, long term changes to the behavior may never be learned and adapted to (e.g., the stock split example). Therefore, a different update policy needs to be used for data points that are anomalous. In the Holt-Winters model for example, reducing the value of model parameters acts as a reduction in speed of adaptation. A simple policy can be:

- Given that x_t is the first anomalous data point
- Reduce α, β and γ, the Holt-Winters parameters by a factor F.
- Repeat for every t' in $t + 1, ..., t + T$, where T is a parameter indicating the longest expected anomaly,
- If x'_t is an anomaly and $t' < t + T$ increase the HW parameters by a small factor such that at $t' = t + T$ they return to their original values.
- otherwise return the original α, β and γ values.

The policy above ensures that anomalies initially have little impact on the model of normality, but lasting anomalies eventually will change the normal model and adapt to the new state of the signal.

4. Abnormal behavior learning

Not all anomalies are equal. Some are more significant than others, and the reaction an anomaly causes might depend upon how significant it is. how do we understand which anomaly is more significant than another?

For every anomaly found in a metric, there is a notion of how far it deviates from normal as well as how long the anomaly lasts. There are additional attributes of each anomaly, such as its volatility and absolute peak (compared to the signal range) and relative peak (compared to the anomalies). To learn how to score the anomalies we fit a multivariate positive distribution (such as the exponential distribution) based on all past anomalies. Given a new anomaly, we compute its attributes and compute the significance using the estimated distribution that was trained using all past anomalies. This approach mimics that way humans rank anomalies. The stranger the current anomaly is compared to past anomalies, the higher its significance.

5. Behavior Topology learning

The next step in the overall process of learning and identifying anomalies in a system is behavioral topology learning.

Behavioral topology learning provides the means to learn actual relationships among different metrics. We employ several methods for discovering the relationships: abnormal and normal based similarity and similarity based on the meta data attached to each signal (i.e. a textual description of the signal). With millions to billions of time series, computing the similarities is a challenging task computationally.

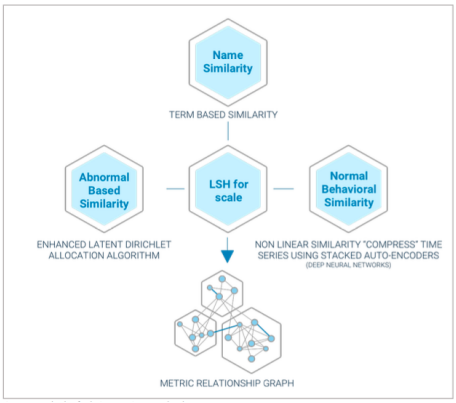


Figure 5: A framework for discovering topological relationships between time series

Describing all three methods is beyond the scope of this paper - but let's describe abnormal based similarity. Intuitively, human beings know that when something is anomalous, it will typically affect more than one key performance indicator (KPI). For example, when someone has the flu, the illness will affect his or her temperature, and possibly also heart rate, skin pH, and so on. Many parts of this system called a body will be affected in a related way. When an automatic anomaly detection system takes in these measurements, it does not know that the temperature, heart rate and skin pH are from the same person (unless someone tells the system that fact). However, if the person gets the flu several times, several of his or her vital signs will become anomalous at the same time, thus there is a high likelihood that some of the anomalies on their measurements will overlap. The chance of two metrics having a single concurrent anomaly is high if you are measuring many things. If we were to simply rely on anomalies happening together to determine that they are related, it would cause many mistakes. But the probability of them being anomalous twice at the same time is much lower. Three times, even lower. The more often the metrics are anomalous at similar times, the more likely it is that they are related.

The metrics do not always have to be anomalous together. A person temperature could increase but his or her heart rate might not increase at the same time, depending on

the illness. But we know that many illnesses do cause changes to the vital signs together. Based on these intuitions, one can design algorithms that find the abnormal based similarity between metrics. One way to find abnormal based similarity is to apply clustering. One possible input to the clustering algorithm would be the representation of each metric as anomalous or not over time (vectors of $0s$ and $1s$); the output is groups of metrics that are found to belong to the same cluster. While there are a variety of clustering algorithms, "soft" clustering is required in this case, as a metric may belong to multiple groups. Latent Dirichlet Allocation is an example algorithm that allows each item (metric) to belong to multiple groups.

Scaling such methods is also important. Using Locality Sensitive Hashing on the vector representing each metric provides initial rough grouping of the entire set of metrics, and in each the full algorithm can be applied.

Figure 5 illustrates the three types of similarities learned to create the behavioral topology. Once learned, metrics can be grouped to discover complex multivariate anomalies.

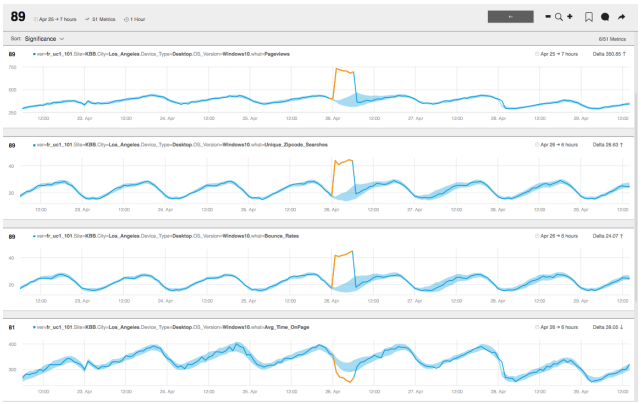
6. Results

Example of important anomalies are abundant in our system. Figure 6(a) illustrates an anomaly discovered by our system that is fraud related. A bot begins attempting to search for products in multiple zipcodes, causes increase in pageviews, unique zipcode related searches, and bounce rates, while decreasing the average time on page, among other metrics. The anomaly group highlights that it happens on a particular site, with traffic originating from a certain geography and client side operating system. Quickly blocking the bot's activity using his signature stops the attack before any damage is done.

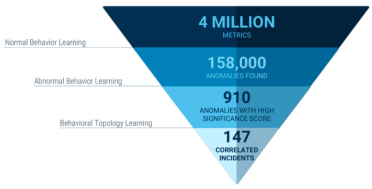
Figure 6(b) illustrates the importance of all the steps in an anomaly detection system: normal behavior learning, abnormal behavior learning, and behavioral topology learning. It is based on 4 million metrics for one company over a period of a week. Out of this, we found 158,000 single metric anomalies in a given week, meaning any anomaly on any metric. This is the result of using our system to do anomaly detection only at the single metric level, without anomaly scoring and without metric grouping. Without the means to filter things, the system gives us all the anomalies, and that is typically a very large number. Even though we started with 4 million metrics, 158,000 is still a very big number, too big to effectively investigate; thus, we need the additional techniques to whittle down that number. If we look at only the anomalies that have a high significance score, in this case a score of probability of 0.7 (score 70) or above, the number of anomalies drops off dramatically by an order of magnitude to just over 910. This is the number of significant anomalies we had for single metrics out of 4 million metrics for one week, 910 of them. Better, but still too many to investigate thoroughly. The bottom of the funnel shows how many grouped anomalies with high significance we end up after applying behavioral topology learning techniques. This is another order of magnitude reduction, from 910 to 147. This number of anomalies is far more manageable to investigate. Any organization with 4 million metrics is large enough to have numerous people assigned to dig into the distilled number of anomalies, typically looking at those anomalies that are relevant to their areas of responsibility.

7. Summary

In this paper we outlined our large scale anomaly detection system, which is used commercially by dozens of companies. We described the requirements from such a system and how we addressed them: namely, advanced algorithms to learn normal behavior, an abnormal



(a)



(b)

Figure 6: (a) An anomaly which includes multiple metrics (51) showing a bot related fraud on a commercial website. (b) Illustration of the importance of all steps

behavior probabilistic model, and scalable methods for discovering relationships between time series for the purpose of anomaly grouping.

To date, our system handled over 120 million time series, discovering around 20 million with seasonal patterns, over a 500 million discovered relationships between metrics and processing over 6 billion data points per day. We have not discussed the method by which feedback from users on the quality of anomalies affect the learning methods - pushing the learning problem towards a semi-supervised approach, whenever such labels are available.

References

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. volume 41, page 15. ACM, 2009.

Chris Chatfield. The holt-winters forecasting procedure. pages 264–279. JSTOR, 1978.

Chris Chatfield. *The analysis of time series: an introduction*. CRC press, 2016.

Balachander Krishnamurthy, Subhabrata Sen, Yin Zhang, and Yan Chen. Sketch-based change detection: methods, evaluation, and applications. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 234–247. ACM, 2003.

Vladilen F. Pisarenko. The retrieval of harmonics from a covariance function. volume 33, pages 347–366, 1973. URL http://gji.oxfordjournals.org/content/33/3/347.short.

Ralph Schmidt. Multiple emitter location and signal parameter estimation. volume 34, pages 276–280. IEEE, 1986.

Peter Welch. The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. volume 15, pages 70–73. IEEE, 1967.