Performance Comparison of Anomaly Detection Algorithms for Streaming Data

Zirije Hasani (zh12796@seeu.edu.mk)
UPZ https://orcid.org/0000-0001-6888-9465
Jakup Fondaj
South East European University Faculty of Contemporary Sciences and Technologies

Research

Keywords: Time series data, anomaly detection, big streaming data, Numenta, e-dnevnik

Posted Date: June 18th, 2020

DOI: https://doi.org/10.21203/rs.3.rs-28521/v1

Version of Record: A version of this preprint was published at Journal of Computer Science on July 1st, 2020. See the published version at https://doi.org/10.3844/jcssp.2020.950.955.

Abstract

Most of the today's world data are streaming, time-series data, where anomalies detection gives significant information of possible critical situations. Yet, detecting anomalies in big streaming data is a difficult task, requiring detectors to acquire and process data in a real-time, as they occur, even before they are stored and instantly alarm on potential threats. Suitable to the need for real-time alarm and unsupervised procedures for massive streaming data anomaly detection, algorithms have to be robust, with low processing time, eventually at the cost of the accuracy. In this work we compare the performance of our proposed anomaly detection algorithm HW-GA[1] with other existing methods as ARIMA [10], Moving Average [11] and Holt Winters [12]. The algorithms are tested and results are visualized in the system R, on the three Numenta datasets, with known anomalies and own e-dnevnik dataset with unknown anomalies. Evaluation is done by comparing achieved results (the algorithm execution time and CPU usage). Our interest is monitoring of the streaming log data that are generating in the national educational network (e-dnevnik) that acquires a massive number of online queries and to detect anomalies in order to scale up performance, prevent network downs, alarm on possible attacks and similar.

Introduction

Proposing an algorithm for detecting real-time anomalies in large amounts of data is not an easy task due to the fact that many researchers tend to show that their solution is better. In research done earlier [1] we have proposed an algorithm for detecting anomalies in large real-time data. We tested the accuracy of the algorithm there, comparing it to several other algorithms that we singled out from previous research such as ARIMA [10], Moving Average[11] and Holt Winters [12].

The purpose of this research is to test the performance of the proposed algorithm HW-GA and compare it with other algorithms which are used for finding anomalies in large amounts of data. Comparisons will be made between different algorithms such as HW-GA, ARIMA, MovingAverage, Holt Winter, etc.

The field of analyzing large amounts of data is current because the number of data is increasing every day. When dealing with large amounts of data, it should be in mind that they are characterized by three characteristics: volume, veracity and variety of data. Hence the need for performance testing in order to meet the speed characteristic of large amounts of data. It is a vast field of research because it involves algorithms from different disciplines. First, to make the selection of the algorithm important is to specify the data to be analyzed in order to know how we make the algorithm selection.

The study will use comparative methods in order to draw conclusions regarding comparative performance. Experiments, statistical analysis and visualization were managed in R, a free software environment for statistical computing and graphics.

There will be used benchmark and real time data to test the algorithm. The NUMENTA benchmark [3] database will be used and real time data from e-dnevnik application which is electronic education system in North Macedonia.

Related Work

In our previous research [2] we have compared many algorithms as MAD, RunMAD, Boxplot, Twitter ADVec, DBSCAN, Moving Range Technique, Statistical Control Chart Techniques, ARIMA and Moving Average, to find which one is faster. So the most important aspects that we considered, in order to find anomaly detection algorithm suitable for future implementation in the online environment was the execution time (complexity), the CPU usage and the satisfactory quality of algorithm (measured through TP- True Positive, FP-False Positive, FN-False Negative, TN-True Negative anomalies found).

Best algorithms selected from that research ARIMA and Moving Average are compared with or proposed algorithm [1] and Holt Winters where we have tested the correctness of our algorithm in our previous research [1] and now in this research we are going to test the performance/ speed and CPU usage of our algorithm.

The authors [3] propose a benchmark Numenta Anomaly Benchmark (NAB), this benchmark is used in our research. Numenta Anomaly Benchmark (NAB), attempts to provide a controlled and repeatable environment of open-source tools to test and measure anomaly detection algorithms on streaming data. The perfect detector would detect all anomalies as soon as possible, trigger no false alarms, work with real-world time-series data across a variety of domains, and automatically adapt to changing statistics.

The authors [4] propose an online and unsupervised anomaly detection algorithm for streaming data using an array of sliding windows and the probability density-based descriptors (PDDs) (based on these windows). This algorithm mainly consists of three steps: 1) we use a main sliding window over streaming data and segment this window into an array of nonoverlapping subwindows; 2) we propose the PDDs with dimension reduction, based on the kernel density estimation, to estimate the probability density of data in each subwindow; and 3) we design the distance-based anomaly detection rule to determine whether the current observation is anomalous. The experimental results and performances are presented based on the Numenta anomaly benchmark. Compared with the anomaly detection algorithm using the hierarchical temporal memory proposed by Numenta (which outperforms a wide range of other anomaly detection algorithms), our algorithm can perform better in many cases, that is, with higher detection rates and earlier detection for contextual anomalies and concept drifts.

Martin,.et.al.[5] investigate to what extent sequence-based Markov models can be used for anomaly detection by means of the endusers' control sequences in the video streams, i.e., event sequences such as play, pause, resume and stop. This anomaly detection approach is further investigated over three different temporal resolutions in the data, more specifically: 1 h, 1 day and 3 days. The proposed anomaly detection approach supports anomaly detection in ongoing streaming sessions as it recalculates the probability for a specific session to be anomalous for each new streaming control event that is received. Two experiments are used for measuring the potential of the approach, which gives promising results in terms of precision, recall, F1-score and Jaccard index when compared to k-means clustering of the sessions.

The authors Filipe Falcão, et.al., [6] they evaluate experimentally a pool of twelve unsupervised anomaly detection algorithms on five attacks datasets. Results allow elaborating on a wide range of arguments, from the behavior of the individual algorithm to the suitability of the datasets to anomaly detection. They identify the families of algorithms that are more effective for intrusion detection, and the families that are more robust

to the choice of configuration parameters. Further, they confirm experimentally that attacks with unstable and non-repeatable behavior are more difficult to detect, and that datasets where anomalies are rare events usually result in better detection scores.

Huihui Zhu, et.al., [7] propose a real-time anomaly detection framework with low computational complexity and high efficiency. They propose Histogram of Magnitude Optical Flow (HMOF) which capture the motion of video patches. They show that HMOF is more sensitive to motion magnitude and more efficient to distinguish anomaly information. Experimentally they show that the framework outperforms state-of-the-art methods, and can reliably detect anomalies in real-time.

Based on the existing research summarized above, we have identified a research gap related to the analysis of speed of anomaly detection algorithms HW-GA.

Methods

To test the performance of HW-GA we have used different datasets to make the experiments. The real data from e-dnevnik and NAB benchmark [3].

NAB contains datasets with a real world, labeled data files across multiple domains and the associated anomaly detectors applicable for the streaming data. We use three NAB datasets, HotGym (the energy consumption in one gym center in Australia), CPU utilization and NycTaxi (the number of ride for NYC taxi), as also our e-dnevnik. Parts of the datasets are in the table below.

1. Part of benchmark data used for experiments

The datasets contain a timestamp and single value based on the log. The first two have real and next two integer values. Known anomalies are detected by human inspection and confirmed by HTM algorithm in all three NAB datasets, while in the e-dnevnik dataset anomalies are not known.

In Fig. 1 below is presented a sequence of data analyzed from e-dnevnik. They are data with seasonality, is the working period from 07:00 to 19:00, and not the working period 19: 00–07: 00. If there is an increase in demand during the non-working period, it is calculated as an anomaly. Figure 1 shows the data for the two-week period.

In Fig. 2 below are shown with red circles what means one anomaly in our real data and the algorithms tend to pick them in the result.

Algorithms Used For Testing

In statistics and econometrics, and in particular in time series analysis, an autoregressive integrated moving average (ARIMA)[10] model is a generalization of an autoregressive moving average (ARMA) model. Both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting) Moving average. In time series analysis, the moving-average (MA)[11] model is a common approach for modeling univariate time series. Together with the autoregressive (AR) model, the moving-average

model is a special case and key component of the more general ARMA and ARIMA models of time series, which have a more complicated stochastic structure.

HW-GA algorithm: The Adaptive Algorithm for Anomaly Detection. In Fig. 3 the positive feedback optimization method for continuous adaptation of the anomaly detection parameters is shown. The method is composed of four different stages.

First is the annotation of the anomalies in the training dataset. The anomaly annotation is defined as a time interval where an anomaly is located. The annotation is done by a human or an oracle.

The second stage is the computation of anomaly detection parameters for our algorithm using GAs, i.e. computation of HW or TDHW parameters, together with δ, *k* and *n*. GAs have been successfully applied to solve optimization problems, both for continuous (whether differentiable or not) and discrete functions. This enables us to find near-optimal values of the anomaly detection parameters very successfully.

The third stage is the actual anomaly detection engine based on the computed optimal parameters from the second stage. This stage outputs the detected anomalies with our proposed algorithm.

The fourth stage is the human acknowledgment of the output data, and classify the output data into TP (true positive), FP (false positive) and FN (false negative). The result of the verification/ acknowledgment stage is then used again in the second stage for further optimization of the anomaly detection parameters.

Results

In real time analytic is very important the speed because it have to deal with real time data. Our proposed algorithm HW GA [1] is tested for correctness but not for performance. In this paper we test the performance of our algorithm and compare it with other selected algorithms from previous research.

Our proposed algorithm (HW-GA)[1] with GA optimized parameters (α, β, γ, δ, *k*, *n*) and with improved $MASE_{t_{(\alpha,\beta,\gamma,\delta,k,n)}}$ is copared with ARIMA, MA (implemented in our previous work [2]), HW where smoothing parameters are calculated by formula and default MASE (HW calc.MASE), HW by default smoothing parameters (optimized in R) and default MASE (HW def.MASE), HW by default smoothing parameters and improved $MASE_{k,n}$ (HW def.MASE(k,n)).

Algorithms evaluation focus on execution response time and CPU usage. These parameters are important for us because, in the future, the algorithm HW-GA have to work in the real-time environment. Other criteria should be robustness, flexibility, scalability and simplicity to implement in our online infrastructure [8, 9].

Table 2 below show the execution time and CPU usage for six algorithms which are compared between them HW-GA with others. The experiments are done in real data and benchmark data as a described above. From the result we can see that in real data e-dnevnik data the execution time is faster in HW-GA 3.36seconds compared to other which is larger also the CPU usage is smaller in real data. The Execution time depends also from the amount of data to be tested this affects also the CPU usage but when it is compared to HW-GA it shows better results.

1. Experimental results from CPU usage and execution time

	E-dnevnik > 40000		NYcTaxi-1441		HotGYM-169		CPU usage-3653	
Algorithms	Execution time(seconds)	CPU Usage	Execution time (seconds)	CPU Usage	Execution time (seconds)	CPU Usage	Execution time (seconds)	CPU Usage
HW GA	3.36	27%	1.20	9.8%	1.69	9.9%	0.32	12%
HW calc. MASE	12.17	43%	1.28	11.3%	1.47	14.6%	5.66	18%
HW def. MASE(k)	5.77	39%	1.44	10.4	1.21	10.4%	5.04	17.9%
HW def. MASE(k,n)	5.98	45%	1.22	10.6%	1.37	10.5	5.51	18%
ARIMA	3.52	32%	1.21	3%	0.73	2.3%	4.07	17.6%
MA	23.38	65%	1.27	7.6%	0.53	3.2%	14.38	22%

In real data the last algorithm MA show the larger CPU usage 65% but is not the same situation which benchmark data, but this happen because the amount of data in e-dnevnik real data is larger than 40000 where in others is much smaller (NYcTaxi-1441 records, HotGYM-169 records, CPU usage-3653 records).

Discussion

From table 2 we can see that CPU usage depends from the number of records in one dataset. In general, two smaller datasets of NYcTaxi and HotGYM the execution time is smaller compared to CPU usage which have more than three thousand records. This is for benchmark dataset but the real dataset from e-dnevnik have larger amount of data more than 40 thousand and here we can see that CPU usage is larger compared to others. The other thing that we can see is that ARIMA and MA in general in all datasets have smaller CPU usage this because they are not complicated algorithms compared to HW-GA but related to correctness they are not good.

If we compare HW-GA with Holt Winters and their modifications we made the CPU usage is smaller in all datasets.

Based on the results we get from the execution time we can say that when the dataset is larger our algorithm HW-GA show better results compared to smaller datasets. For example, the execution time in e-dnevnik dataset for HW-GA is 3.36 seconds all others have more then 5 seconds execution time. The worse result show MA with 23.38 seconds execution time on e-dnevnik dataset.

On two smaller datasets NYcTaxi and HotGYM our algorithm good execution time 1.2 and 1.69 seconds on these two datasets. Also, the others algorithm is these two small benchmark datasets outperform well.

On the other side the third benchmark dataset CPU-Usage which have more than 3 thousand records our algorithm has performed much better than others with execution time 0.32 seconds. When it is compared with

others the difference is very large more then 5 seconds. The worst case is with MA with 14.38 seconds execution time.

Based on these facts that we describe here we can say that our algorithm outperforms others algorithm in both measurements parameters (execution time and CPU usage) and better results are shown with large amount of data which is very important for Big Data.

Conclusion

As a conclusion from our experiment, we may say that HW-GA is efficient concerning execution time and CPU usage. Our algorithm have smaller CPU usage and less execution time compared to other algorithm. The results are shown in table 2.

Based on this research we can say that our algorithm outperforms others algorithm in both measurements parameters (execution time and CPU usage) and better results are shown with large amount of data which is very important for Big Data.

In our continuous work, we are building an infrastructure for anomaly detection in the big log files in real-time that contains computational, storage, scalability and real-time challenge. To make proper choice of infrastructure we have done extensive investigation reported in [8] and [9]. We have found infrastructure appropriate, because it is possible to modify it when needed by adding various other components or scale up or down by adding (duplicate, triplicate) some of its existing components. Ongoing experiments are motivated by need to use such an infrastructure for anomaly detection in big log data generated by load balancers of servers in Faculty of Computer Sciences and Engineering (FINKI).

The next phase of our research is to modify the proposed algorithm in order to add more parameters into the optimization procedure with Genetic Algorithms in order to see if it will give better performance. Also, we plane to implement this algorithm in our proposed infrastructure and to test it in real time environment.

Abbreviations

NAB-Numenta Anomaly Benchmark

HW- Holt Winters

FINKI- Faculty of Computer Sciences and Engineering

HW-GA-Our proposed method

MA- moving-average

Declarations

Availability of data and materials

Not applicable

Competing interests

Not applicable

Funding

Not applicable

Authors' contributions

Author 1 do the research and write the paper.

Author 2 do the experiments

Acknowledgements

Not applicable

Ethics

There are no ethics issues!

References

1. Margita Kon-Popovska.
   Zirije Hasani B, Jakimovski G, Velinov. Margita Kon-Popovska.: An Adaptive Anomaly Detection Algorithm for Periodic Real Time Data Streams. International Conference on Intelligent Data Engineering and Automated Learning (IDEAL). Springer, Spain (2018).
2. Zirije Hasani.: Robust anomaly detection algorithms for real-time big data: Comparison of algorithms. 6th Mediterranean Conference on Embedded Computing (MECO). pp. 1–6. IEEE, Montenegro. (2017).
3. Alexander Lavin and Subutai Ahmad. Evaluating Real-time Anomaly Detection Algorithms – the Numenta Anomaly Benchmark. 2015 IEEE 14th International Conference on Machine Learning and Applications.
4. Zhang L, Zhao J, Li W. Online and Unsupervised Anomaly Detection for Streaming Data Using an Array of Sliding Windows and PDDs. IEEE TRANSACTIONS ON CYBERNETICS 2019.
5. Boldt M, Borg A, Ickin S, Jörgen Gustafsson. Anomaly detection of event sequences using multiple temporal resolutions and Markov chains. Knowl Inf Syst. 2020;62:669–86.
6. Filipe Falcão T, Zoppi CBarbosaV, Silva A, Santos B, Fonseca A, Ceccarelli. Andrea Bondavalli. Quantitative Comparison of Unsupervised Anomaly Detection Algorithms for Intrusion Detection. SAC'19, April 8–12, 2019, Limassol, Cyprus.
7. Zhu H, Liu B, Lu Y, Li W. Nenghai Yu. Real-time Anomaly Detection with HMOF Feature. ICVIP 2018, December 29–31, 2018, Hong Kong, Hong Kong.
8. Zirije Hasani B, Jakimovski, Margita Kon-Popovska and Goran Velinov. *Real-time analytic of SQL queries based on log analytic.* ICT Innovations 2015.

9. Zirije Hasani. Implementation of infrastructure for streaming outlier detection in Big Data. WorldCist'17–5th World Conference on Information Systems and Technologies. To be published by Springer Advances in Intelligent Systems and Computing, Porto Santo Island, Madeira, Portugal, 11–13 April 2017.
10. Kasunic M, McCurley J. Dennis Goldenson and David Zubrow. An Investigation of Techniques for Detecting Data Anomalies in Earned Value Management Data. Carnegie Mellon University. 2012;3(2):2–103.
11. Moving Average. https://cran.r-project.org/web/packages/smooth/vignettes/sma.html. last accessed February 28, 2018.
12. Jarkko Ekberg J, Ylinen PLoula: Network Behaviour Anomaly Detection Using Holt-Winters Algorithm. 6th International Conference on Internet Technology and Secured Transactions, pp. 627–631. IEEE, Piscataway, NJ, (2011).

Tables

TABLE I. PART OF BENCHMARK DATA USED FOR EXPERIMENTS

HotGym	
timestamp	kw_energy_consumption
7/2/2010 0:00	21.2
7/2/2010 1:00	16.4
7/2/2010 2:00	4.7

CPU utilization	
timestamp	metric_value
4/10/2014 0:04	93.1456
4/10/2014 0:24	94.5935
4/10/2014 0:44	93.521

Nyctaxi	
timestamp	value
7/1/2014 0:00	10844
7/1/2014 0:30	8127
7/1/2014 1:00	6210

Rtime e-dnevnik	
timestamp	value
6/13/2016 0:00	6413
6/13/2016 0:00	345
6/13/2016 0:00	354

TABLE II. EXPERIMENTAL RESULTS FROM CPU USAGE AND EXECUTION TIME

	E-dnevnik>40000		**NYcTaxi-1441**		**HotGYM-169**		**CPU usage-3653**	
Algorithms	Execution time(seconds)	CPU Usage	Execution time (seconds)	CPU Usage	Execution time (seconds)	CPU Usage	Execution time (seconds)	CPU Usage
HW GA	3.36	27%	1.20	9.8%	1.69	9.9%	0.32	12%
HW calc. MASE	12.17	43%	1.28	11.3%	1.47	14.6%	5.66	18%
HW def. MASE(k)	5.77	39%	1.44	10.4	1.21	10.4%	5.04	17.9%
HW def. MASE(k,n)	5.98	45%	1.22	10.6%	1.37	10.5	5.51	18%
ARIMA	3.52	32%	1.21	3%	0.73	2.3%	4.07	17.6%
MA	23.38	65%	1.27	7.6%	0.53	3.2%	14.38	22%

Figures

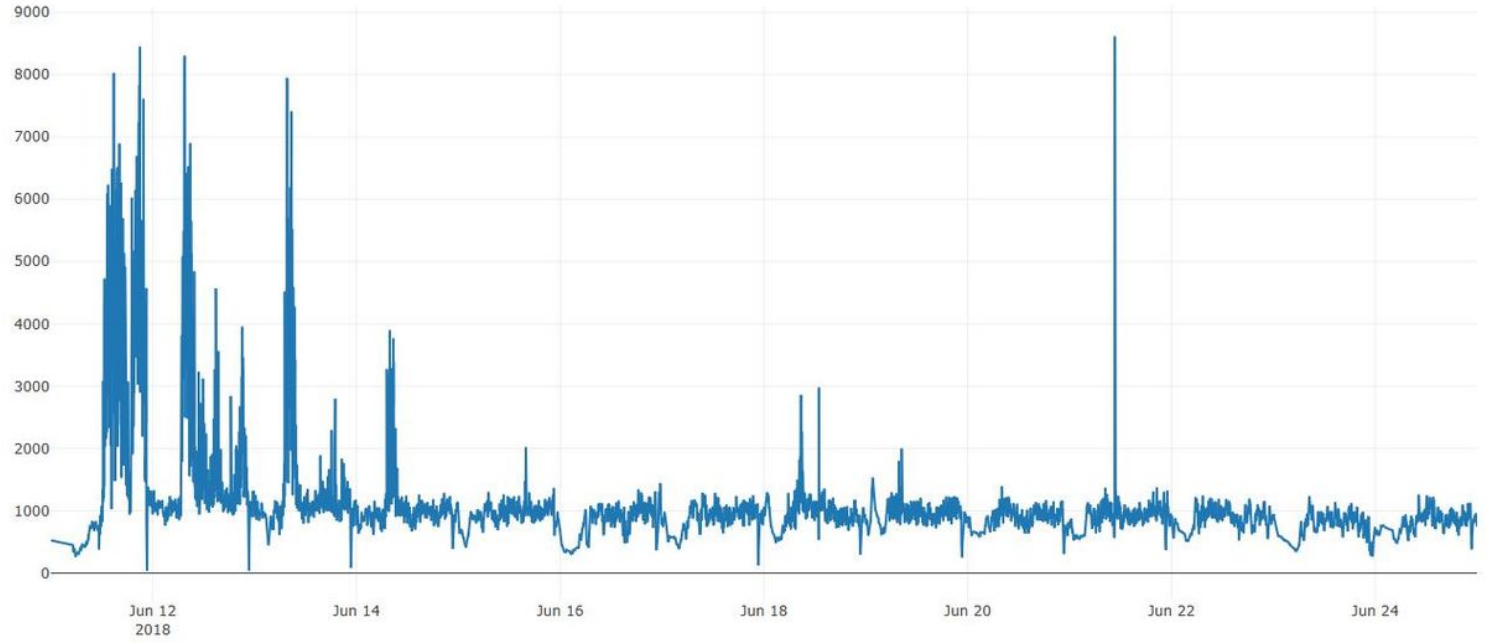

Figure 1

e-dnevnik two-week data.

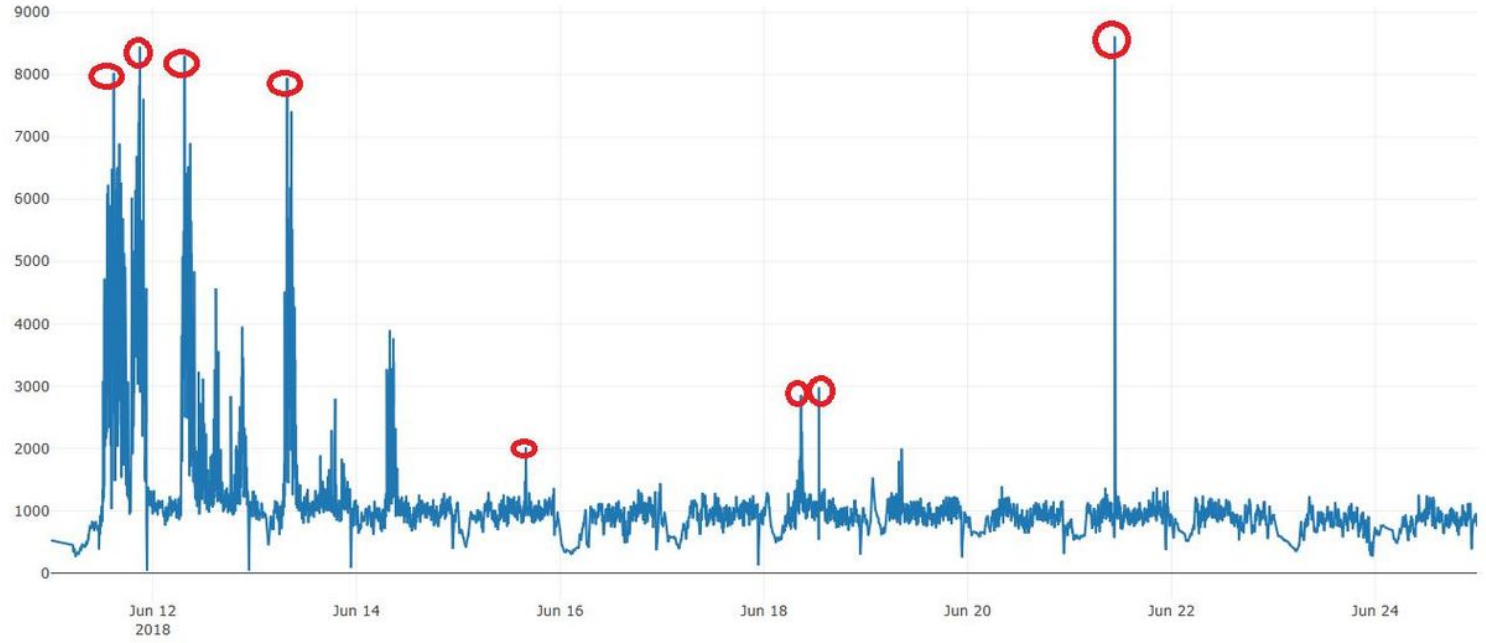

Figure 2

Anomalies in e-dnevnik data.

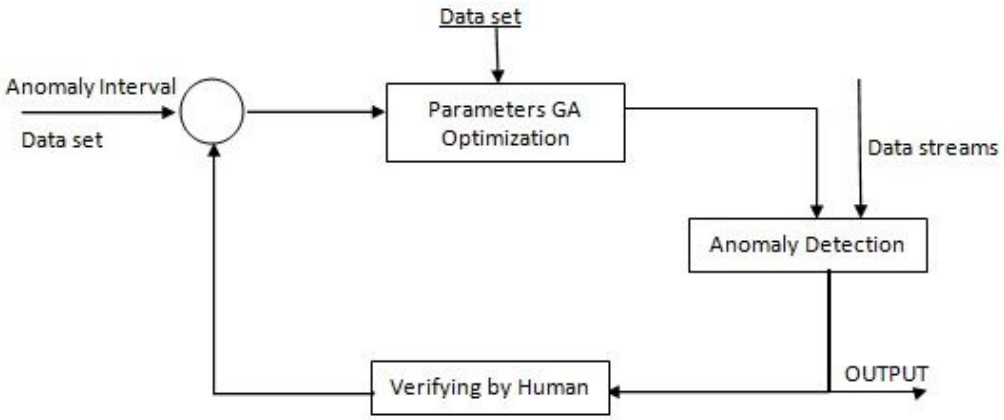

Figure 3

Model for HW-GA method for anomaly detection.