TimeEval: A Benchmarking Toolkit for Time Series Anomaly Detection Algorithms

Phillip Wenig*
Hasso Plattner Institute,
University of Potsdam
Potsdam, Germany
phillip.wenig@hpi.de

Sebastian Schmidl*
Hasso Plattner Institute,
University of Potsdam
Potsdam, Germany
sebastian.schmidl@hpi.de

Thorsten Papenbrock
Philipps University of Marburg
Marburg, Germany
papenbrock@informatik.uni-marburg.de

ABSTRACT

Detecting anomalous subsequences in time series is an important task in time series analytics because it serves the identification of special events, such as production faults, delivery bottlenecks, system defects, or heart flicker. Consequently, many algorithms have been developed for the automatic detection of such anomalous patterns. The enormous number of approaches (i. e., more than 158 as of today), the lack of properly labeled test data, and the complexity of time series anomaly benchmarking have, though, led to a situation where choosing the best detection technique for a given anomaly detection task is a difficult challenge.

In this demonstration, we present TimeEval, an extensible, scalable and automatic benchmarking toolkit for time series anomaly detection algorithms. TimeEval includes an extensive data generator and supports both interactive and batch evaluation scenarios. With our novel toolkit, we aim to ease the evaluation effort and help the community to provide more meaningful evaluations.

PVLDB Reference Format:
Phillip Wenig, Sebastian Schmidl, and Thorsten Papenbrock. TimeEval: A Benchmarking Toolkit for Time Series Anomaly Detection Algorithms. PVLDB, 15(12): 3678 - 3681, 2022.
doi:10.14778/3554821.3554873

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/hpi-information-systems/timeeval-gui.

1 TIME SERIES ANOMALY DETECTION

Time series anomaly detection (TSAD) is the process of detecting unusual and, hence, anomalous subsequences in data series. An *anomaly* can describe special events, such as heart failures in cardiology [1], structural defects in jet turbine engineering [13], or ecosystem disturbances in earth sciences [3]. For this reason, anomaly detection is a central activity in various domains ranging from car manufacturing over finance applications to health monitoring. The relevance of anomalies has led to the development of a plethora of different anomaly detection algorithms in different domains. In a recent study [10], we collected 158 publications on TSAD algorithms and benchmarked a representative set of 71 algorithms on more than 900 datasets. In this demonstration paper, we introduce the TimeEval benchmarking toolkit that enabled this research project. We believe that the toolkit will help our large anomaly detection community to ensure the long-term quality assurance of future TSAD algorithm evaluations.

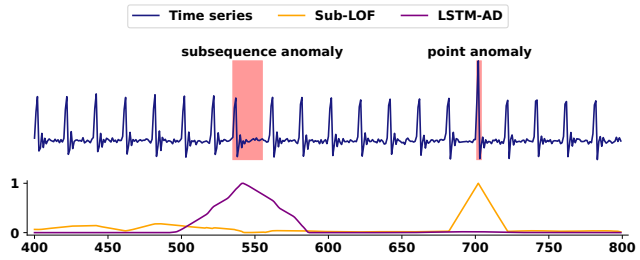

Figure 1: A synthetic ECG time series generated with TimeEval and the scorings of two algorithms. The time series has a subsequence anomaly (pattern shift) and a point anomaly (extremum).

A *data series* as shown in Figure 1 is an ordered sequence of data points recorded in equidistant intervals based on some continuous measure, such as temperature, angle, position, or speed. If the data series is ordered by time, it is called a *time series*. Because most anomaly detection approaches are agnostic of the reference measure, we can use the terms data series and time series interchangeably. A time series *anomaly* is either a point or a point sequence that deviates significantly from the regular patterns observed in the time series. Anomalies might have different lengths, shapes, and magnitudes, and they could re-appear multiple times in the same time series. As the anomaly detection algorithms sometimes differ in their formulation of an anomaly, we proposed to translate all detection results into point-wise anomaly scores, such as those also shown in Figure 1, to make them comparable [10].

The need for a systematic TSAD benchmarking toolkit arises from the observation that the time series research community struggles to offer a consistent and complete overview of their research achievements: Many algorithms are remarkably heterogeneous in their detection approaches, the current number of proposed algorithms is large, and the research on novel algorithms is still very active. What makes the necessary evaluations even harder is a lack of well-labeled benchmarking datasets, a careful consideration of runtime and quality aspects, and the need to evaluate a multitude of specific properties including anomalies that are uni-/multivariate,

*Both authors contributed equally to this work.

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 15, No. 12 ISSN 2150-8097.
doi:10.14778/3554821.3554873

point/sequence, unique/repeating, shape/magnitude etc. anomalous. For this reason, most TSAD publications focus on specific, isolated domains, which makes their evaluations severely limited in scope. Wu and Keogh even showed that most evaluations use trivial, cherry-picked, biased, mislabeled, or only few datasets [14].

As an answer to the evaluation deficit, we developed TimeEval, a Python toolkit that evaluates TSAD algorithms on real-world and synthetically generated datasets in either automatic or interactive mode. It consists of four major components: a dataset generator called GutenTAG, the core evaluation engine called Eval, a Python API to programmatically configure systematic batch experiments, and a visual frontend for interactive experiments. TimeEval is highly customizable through various configuration options and user-defined code/plugins to meet different existing and upcoming evaluation settings. The toolkit can execute custom TSAD algorithms and work with external datasets, which ensures its applicability to future evaluation needs.

In our demonstration, we show a TimeEval setup with 71 state-of-the-art TSAD algorithms and 1,354 benchmark datasets from related work. Attendees will be able to inspect the results of extensive TimeEval runs that took multiple weeks to execute (cf. [10]). These results provide detailed insights about the performance of specific TSAD algorithms. The demo will also offer the opportunity to interact with the toolkit, create own benchmarking datasets, and run challenging experiments with algorithms of choice. In this way, the demo is supposed to spark a lively discussion in our community and improve collaboration between research areas.

In this paper, we (i) discuss the use cases of TimeEval (Section 2), (ii) introduce TimeEval's architecture (Section 3), and (iii) present the contents of our live system demonstration (Section 4).

2 TIMEEVAL USE CASES

Systematic evaluation. TimeEval's primary use case is the systematic evaluation of numerous TSAD algorithms on large sets of time series. We already demonstrated this use case in a recent extensive evaluation [10]. Data scientists can use the toolkit to specify all necessary evaluations as a batch job. TimeEval then automatically generates the according executions, distributes them on a compute cluster, and collects and aggregates the results; intermediate results are written to disk to prevent data loss on failures or early stopping.
Algorithm selection. TimeEval helps data scientists to find an appropriate algorithm for a specific anomaly detection task. This use case is the main focus of our live demonstration. If a dataset with ground truth is available, existing algorithms can be benchmarked directly on this dataset. In the more likely case where ground truth data is not available, TimeEval can generate labeled time series with the required characteristics and expected anomaly properties.
Algorithm evaluation. TimeEval supports developers of new TSAD algorithms in evaluating the performance of their novel approaches w. r. t. various dataset characteristics and w. r. t. existing approaches. As we provide the TimeEval toolkit with a broad selection of state-of-the-art TSAD algorithms and datasets, we hope for a wide adoption of our toolkit in future TSAD evaluations. The toolkit's extensibility and its highly configurable dataset generator should also enable the creation of novel and particularly challenging experiments that open new research directions.

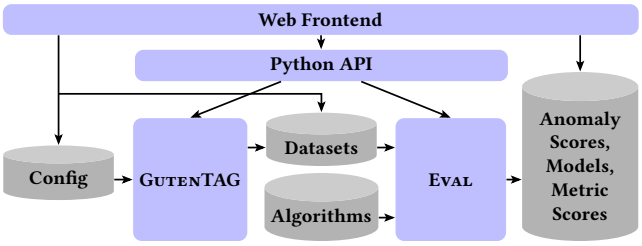

Figure 2: Architecture of the TimeEval toolkit.

3 TIMEEVAL ARCHITECTURE

The open-source TimeEval toolkit[1] consists of four components: The dataset generator GutenTAG, the evaluation engine Eval, a powerful Python API, and an interactive web frontend. Figure 2 visualizes the interaction between these components. By providing both an API and a frontend, TimeEval can be integrated as a code package into other software projects, and it can also be used as a stand-alone tool. We offer GutenTAG and Eval as separate components so that users can utilize them independently, i. e., they can generate time series with certain properties for external projects, and they can evaluate TSAD algorithms on already given datasets.

For the evaluations, TimeEval resorts to a pool of external resources that can be extended without programmatically changing the toolkit. As initial resources, we provide 1,354 benchmark datasets and 71 TSAD algorithms[2]. For documentation purposes, TimeEval stores all generated artifacts, i. e., configurations, log messages, models, anomaly scorings, and algorithm scores, on disk.

In the following sections, we describe GutenTAG (Section 3.1), Eval (Section 3.2), the Python API (Section 3.3), and the frontend (Section 3.4) in more detail.

3.1 Dataset Generation with GutenTAG

The GutenTAG component is a time series generator that can generate new training and testing time series according to the users' desired dataset characteristics and anomalies. Figure 3 depicts GutenTAG's data generation process. Every time series consists of one or more base oscillations (BOs), optional trends, and optional random white noise. To increase complexity, BOs can be combined and stacked to multivariate time series. Once the basic time series is created, GutenTAG injects a user-defined number of anomalies. Because some anomalies must be aligned to the BO's periodicity and patterns, it takes the already generated BOs into account. Afterwards, GutenTAG combines the four elements into one uni- or multivariate anomalous time series. It then generates some time series metadata and index-files for TimeEval; the final result is written to disk. The entire generation process is based on a user-defined, declarative configuration, which can be provided either from a configuration file or through the Python API. If the pre-defined oscillation and anomaly types are insufficient, the user can define new types or implement a custom post-processing routine via plugins.

[1] https://github.com/hpi-information-systems/timeeval-gui.
[2] Datasets and algorithms are published at https://hpi.de/naumann/s/time-series-anomaly-detection-evaluation.

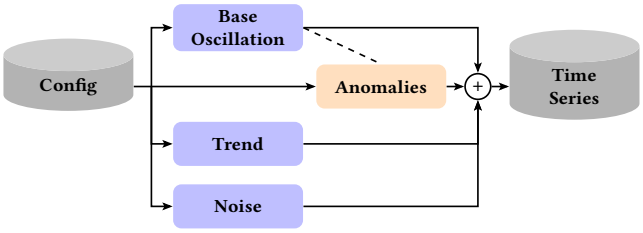

Figure 3: Process of generating a time series with GutenTAG.

3.1.1 Base Oscillations. GutenTAG can generate time series based on 6 pre-defined BOs: *Sine Wave (Sine)* and *Polynomial (Poly)* BOs are generated from their respective mathematical representations; *Electrocardiogram (ECG)* BOs are generated using the NeuroKit2 Python package; and *Chaotic* BOs can be generated based on Random Walk (RW), Cylinder Bell Funnel (CBF) [7], and Random Mode Jump (RMJ) types. GutenTAG provides a special seventh BO type, called *Formula*, that uses an arithmetic language to build mathematical combinations of already defined channels to create correlated multivariate time series.

3.1.2 Anomaly Types. GutenTAG can inject anomalies of 10 types into the BOs. The anomaly types can alter the position of an anomalous pattern (*Mean*, *Pattern-Shift*, *Trend*, *Mode Correlation*), change the shape of a reoccurring pattern (*Amplitude*, *Frequency*, *Pattern*), and inject anomalous artifacts into the BO (*Extremum*, *Variance*, *Platform*). Anomaly types can also be combined with each other to generate, for example, a *Frequency* anomaly with larger *Amplitude*.

3.2 Evaluation of TSAD Algorithms with Eval

The Eval engine takes an algorithm, a (custom or GutenTAG) dataset and a run configuration, which contains the algorithm parameters (windows sizes, thresholds etc.) and the experiment configurations (timeouts, memory limits, metrics, nodes etc.); it then executes the experiment, calculates the quality metrics, and stores all results. The quality metrics capture the *execution time* and the Area under the *Receiver Operating Characteristics* Curve [2, 6] (AUC-ROC), Area Under the *Precision-Recall* Curve [4, 9] (AUC-PR), and Area Under the *ranged-based Precision, range-based Recall* Curve [12] (AUC-P_TR_T) scores, which are the most common, threshold-agnostic evaluation measures for TSAD algorithms [10]. Additional metrics can be added using the Python API. Eval ends an execution when the final results are generated, an error occurred, or a user-defined time or memory limit was exceeded.

Multiple experiments are run isolated from each other in parallel and, if possible, even distributed on a compute cluster. They can conveniently be defined in a single batch configuration; Eval then executes the entire run as the Cartesian product of all algorithms, datasets, and configurations.

To enable the execution of algorithms implemented using different programming languages and runtimes, such as Python, Java, or R, Eval uses an adapter architecture with a flexible calling interface for algorithms. One special adapter is the Docker-adapter that allows the execution of arbitrary code within Docker containers [5]; this adapter is also used for the 71 provided algorithms. The

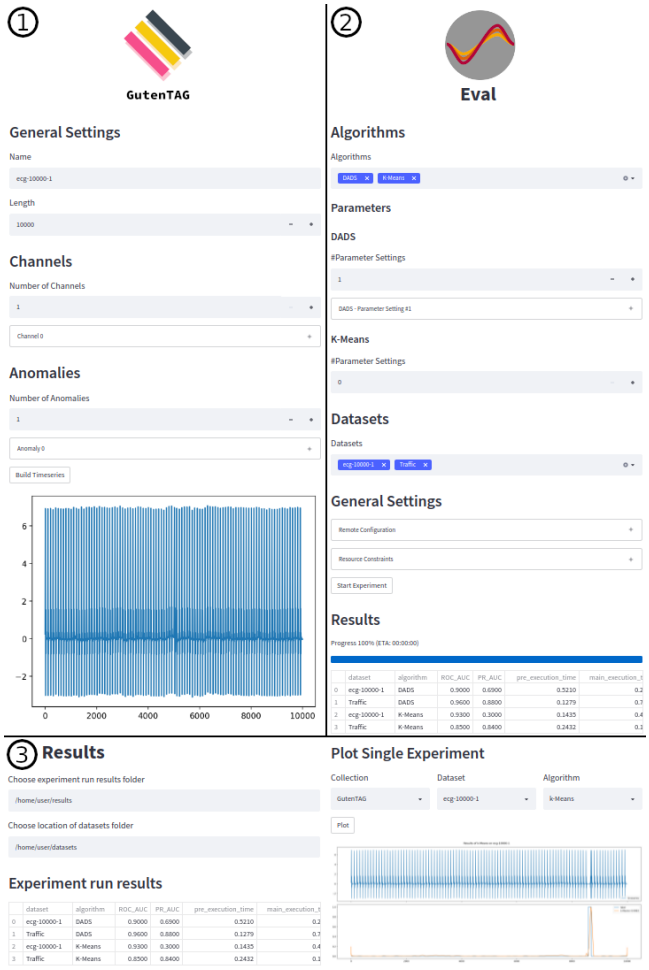

Figure 4: Screenshot of the TimeEval web frontend components: GutenTAG (left column), Eval (right column), and the results view (bottom)

adapters also allow the addition of new algorithms to TimeEval without changing its implementation.

3.3 Dataset and Task Definitions with the API

The Python API is a uniform and comprehensive user-facing interface for the entire TimeEval tookit. It allows the configuration and execution of both GutenTAG and Eval in a programmatic and automated way; both can be imported as separate Python packages. GutenTAG's API provides the functions to generate time series and inject anomalies. Eval's API provides the functions to define and execute experiment configurations, add new algorithms and datasets, and retrieve execution results.

3.4 Interaction with TimeEval's Frontend

The frontend component joins the GutenTAG (Section 3.1) and Eval (Section 3.2) components in a single Graphical User Interface (GUI) and is built on top of the Python API (Section 3.3). It

provides the functionality of the Python API and also reads and writes artifacts (configurations, datasets, metadata, results etc.) to show and manipulate them visually. This allows the configuration, monitoring, and result inspection in an interactive workflow. The frontend uses the Streamlit [11] Python package, with which we redirect user inputs to the GutenTAG and Eval API calls.

Figure 4 shows a screenshot of TimeEval's three frontend pages: GutenTAG ①, Eval ②, and Results ③. The GutenTAG page ① allows the user to define a to-be-generated time series by its properties, such as length, name, number of channels, BO attributes, and various anomaly settings. An export button triggers the generation and saving of the time series. During parameterization, generated time series can be previewed using an interactive plot. The Eval page ② contains the control panels for the configuration of evaluation experiments, such as the selection of algorithms, their parameter settings, and the datasets. The second part of the Eval page provides users with general settings around the experiment execution including remote configuration and resource constraints. The *Start Experiments* button then triggers the execution of the specified experiments. While Eval executes all experiments, a progress bar shows the user how long the experiments are expected to take. Once the entire run has finished, the results can be inspected in a separate, interactive results view ③. This view shows the results table, which lists important insights of an evaluation run, such as the algorithms' quality scores, their runtimes, and an overview about potential errors and timeouts; it also provides visualizations of the anomaly scores of selected algorithms on selected datasets. For additional, in-depth inspections, the results can automatically be loaded into a prepared Jupyter Notebook [8] template.

4 SYSTEM DEMONSTRATION

Our demonstration consists of two parts: First, we show our TimeEval setup consisting of 71 TSAD algorithms and 1,354 datasets that was used to perform the evaluation of [10]; then, we guide our participants through the algorithm selection use case.

In the first part, we demonstrate the programmatic configuration of TimeEval for a systematic evaluation using the Python API. Our goal is to show how easy and concise huge evaluation tasks can be configured in TimeEval. We inspect the results of our extensive evaluation [10] and let attendees discover interesting insights, such as that there is no one-size-fits-all solution, and that different algorithms can detect different kinds of anomalies better or worse (cf. Figure 1). We want to show the variety of existing approaches and encourage researchers to perform their own systematic evaluations.

In the second part, we demonstrate how TimeEval supports the selection of effective TSAD algorithms for specific use cases using its interactive web frontend. According to a pre-defined exercise anomaly detection use case, attendees use GutenTAG (① in Figure 4) to generate proxy/surrogate-datasets with use-case-specific properties and anomalies. They, then, choose algorithm candidates and evaluate them on the generated datasets using the Eval engine (② in Figure 4). When TimeEval displays the evaluation results in the frontend (③ in Figure 4), users can apply filters and sortations (by, e. g., lexicographical order, metric scores, or runtime) to find particularly interesting results. In a further step, attendees can visualize the anomaly scores of the algorithms on a selected dataset and compare their performances, strengths, and weaknesses in detail. While experimenting with TimeEval, users learn how difficult choosing an algorithm for a use case is, what strengths and weaknesses existing TSAD algorithms have, and how important an effective parameterization is.

We hope that our demonstration can contribute to the existing discussion on how to evaluate TSAD algorithms and improve possible future work.

ACKNOWLEDGMENTS

The work was funded by the German government as part of the LuFo VI call I program (Luftfahrtforschungsprogramm) under the grant number 20D1915. The management of Rolls-Royce Deutschland Ltd. & Co. KG is gratefully acknowledged for supporting the work and permitting the presentation of results.

We would like to thank all of our students who supported us in the development of TimeEval and GutenTAG.

REFERENCES

[1] Sardar Ansari, Negar Farzaneh, Marlena Duda, Kelsey Horan, Hedvig B. Andersson, Zachary D. Goldberger, Brahmajee K. Nallamothu, and Kayvan Najarian. 2017. A Review of Automated Methods for Detection of Myocardial Ischemia and Infarction Using Electrocardiogram and Electronic Health Records. *IEEE Reviews in Biomedical Engineering* 10 (2017), 264–298. https://doi.org/10.1109/RBME.2017.2757953

[2] Andrew P Bradley. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition* 30, 7 (1997), 1145–1159. https://doi.org/10.1016/S0031-3203(96)00142-2

[3] Haibin Cheng, Pang-Ning Tan, Christopher Potter, and Steven Klooster. 2009. Detection and Characterization of Anomalies in Multivariate Time Series. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*. 413–424. https://doi.org/10.1137/1.9781611972795.36

[4] Jesse Davis and Mark Goadrich. 2006. The Relationship between Precision-Recall and ROC Curves. In *Proceedings of the International Conference on Machine Learning (ICML)*. 233–240. https://doi.org/10.1145/1143844.1143874

[5] Docker Inc. 2022. Empowering App Development for Developers | Docker. https://www.docker.com

[6] James A Hanley and Barbara J McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* 143, 1 (1982), 29–36. https://doi.org/10.1148/radiology.143.1.7063747

[7] Mohammed Waleed Kadous. 1999. Learning Comprehensible Descriptions of Multivariate Time Series. In *Proceedings of the International Conference on Machine Learning (ICML)*. 454–463.

[8] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. 2016. Jupyter Notebooks – a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. 87–90. https://doi.org/10.3233/978-1-61499-649-1-87

[9] Vijay Raghavan, Peter Bollmann, and Gwang S. Jung. 1989. A Critical Investigation of Recall and Precision as Measures of Retrieval System Performance. *ACM Transactions on Information Systems* 7, 3 (1989), 205—-229. https://doi.org/10.1145/65943.65945

[10] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. 2022. Anomaly Detection in Time Series: A Comprehensive Evaluation. *Proceedings of the VLDB Endowment (PVLDB)* 15, 9 (2022), 1779–1797. https://doi.org/10.14778/3538598.3538602

[11] Streamlit Inc. 2022. Streamlit • The fastest way to build and share data apps. https://streamlit.io

[12] Nesime Tatbul, Tae Jun Lee, Stan Zdonik, Mejbah Alam, and Justin Gottschlich. 2018. Precision and Recall for Time Series. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*. 1920–1930.

[13] Mark Woike, Ali Abdul-Aziz, and Michelle Clem. 2014. Structural health monitoring on turbine engines using microwave blade tip clearance sensors. In *Smart Sensor Phenomena, Technology, Networks, and Systems Integration*. 167–180. https://doi.org/10.1117/12.2044967

[14] Renjie Wu and Eamonn J. Keogh. 2020. *Current Time Series Anomaly Detection Benchmarks Are Flawed and Are Creating the Illusion of Progress*. arXiv:2009.13807 [cs, stat] http://arxiv.org/abs/2009.13807