GraphAn: Graph-based Subsequence Anomaly Detection

Paul Boniol, Themis Palpanas, Mohammed Meftah, Emmanuel Remy
EDF R&D, LIPADE, Université de Paris
{paul.boniol, mohammed.meftah, emmanuel.remy}@edf.fr; themis@mi.parisdescartes.fr

ABSTRACT

Subsequence anomaly detection in long sequences is an important problem with applications in a wide range of domains. However, the state-of-the-art approaches have severe limitations: they either require prior domain knowledge, or become cumbersome and inefficient/ineffective in situations with recurrent anomalies of the same type. We recently proposed Series2Graph, a novel method based on a graph representation of a low-dimensionality embedding of subsequences, that detects anomalous subsequences. The experimental results, on the largest set of synthetic and real datasets used to date, demonstrate that the proposed approach correctly identifies single and recurrent anomalies of various types without any prior knowledge of the characteristics of these anomalies, outperforming by a large margin several competing approaches in accuracy, while being up to orders of magnitude faster. In this demonstration, we present GraphAn, a system based on Series2Graph, showcase the challenges of the problem, and demonstrate the advantages of the proposed system.

PVLDB Reference Format:
Paul Boniol and Themis Palpanas and Mohammed Meftah and Emmanuel Remy. GraphAn: Graph-based Subsequence Anomaly Detection. *PVLDB*, 13(12): 2941-2944, 2020.
DOI: https://doi.org/10.14778/3415478.3415514

Keywords

Time series, Data series, Subsequence anomalies, Outliers.

1. INTRODUCTION

Time series[1] anomaly detection is a crucial problem with application in a wide range of domains. Examples of such applications can be found in manufacturing, astronomy, engineering, and other domains [13, 14]. This implies a real need by relevant applications for developing methods that can accurately and efficiently achieve this goal.

[Anomaly Detection in Sequences] Anomaly detection is a well studied task [4, 16, 19, 11, 5] that can be tackled by either examining single values, or sequences of points. In the specific context of sequences, which is the focus of this paper, we are interested in identifying anomalous subsequences [19, 15, 11, 5, 6], which are not single abnormal values, but rather an abnormal *sequence* of values. In real-world applications, this distinction becomes crucial: in certain cases, even though every individual point may be normal, the trend exhibited by the sequence of these same values may be anomalous. Evidently, failing to identify such situations could lead to severe problems that are only detected when it is too late [3].

[Limitations of Previous Approaches] Some existing techniques explicitly look for a set of pre-determined types of anomalies [10, 1]. These are techniques that have been specifically designed to operate in a particular setting, they require domain expertise, and cannot generalize. Other techniques identify as anomalies the subsequences with the largest distances to their nearest neighbors (termed *discords*) [19, 15, 11]. The assumption is that the most distant subsequence is completely isolated from the "normal" subsequences. However, this definition fails in the case where an anomaly repeats itself (approximately the same). In this situation (sometimes called the twins freaks problem [17]), anomalies will have other anomalies as close neighbors, and will not be identified as discords. In order to remedy this situation, the mth*discord* approach has been proposed [18], which takes into account the multiplicity m of the anomalous subsequences that are similar to one another, and marks as anomalies all the subsequences in the same group. However, this approach assumes that we know the cardinality of the anomalies, which is not true in practice (otherwise, we need to try several different m values, increasing drastically the execution time). Furthermore, the majority of the previous approaches require prior knowledge of the anomaly length, and their performance deteriorates significantly when the correct length value is not used.

[Proposed Approach] In order to address the aforementioned problems, we proposed Series2Graph [7], an unsupervised graph-based approach for subsequence anomaly detection in large data series. This approach first converts the data series into a directed graph (in which a node is a subsequence) and then computes a normality score for each path

[1] A time series, or data series, or sequence, is an ordered sequence of real-valued points. If the dimension that imposes the ordering is time then we talk about *time series*, but it could also be mass, angle, position, etc. We will use the terms *time series*, *data series*, and *sequence* interchangeably.

Proceedings of the VLDB Endowment, Vol. 13, No. 12
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3415478.3415514

(i.e. subsequences) in the resulted graph. This graph is built in three steps, which consist on (i) embedding every subsequence of the data series in a 2-dimensional space, followed by (ii) the extraction of the densest parts of that space, to which we assign nodes, (iii) that are finally connected together by edges that represent the number of time subsequences followed each other. An experimental evaluation has shown that Series2Graph outperforms current state-of-the-art anomaly detection methods, even though being an order of magnitude faster [7].

[Contributions] We present a graph-based subsequence anomaly detection system (GraphAn) based on Series2Graph. It is a web application that enables the users to upload their own data series, run and visualize the inner steps of Series2Graph, and then compare its accuracy and time execution to other anomaly detection algorithm.

2. SERIES2GRAPH APPROACH

2.1 Problem Formulation

We formulate an approach for subsequence anomaly detection based on the data series representation into a Graph.

We first define several basic elements related to graphs. We define a Node Set $\mathcal{N}$ as a set of unique integers. Given a Node Set $\mathcal{N}$, an Edge Set $\mathcal{E}$ is then a set composed of tuples (x_i, x_j), where $x_i, x_j \in \mathcal{N}$. $w(x_i, x_j)$ is the weight of that edge. Given a Node Set $\mathcal{N}$, an Edge Set $\mathcal{E}$ (pairs of nodes in $\mathcal{N}$), a Graph G is an ordered pair $G = (\mathcal{N}, \mathcal{E})$. A directed graph or digraph G is an ordered pair $G = (\mathcal{N}, \mathcal{E})$ where $\mathcal{N}$ is a Node Set, and $\mathcal{E}$ is an ordered Edge Set.

We now provide a new formulation for subsequence anomaly detection. The idea is that a data series is transformed into a sequence of abstract states (corresponding to different subsequence patterns), represented by nodes $\mathcal{N}$ in a directed graph, $G(\mathcal{N}, \mathcal{E})$, where the edges $\mathcal{E}$ encode the number of times one state occurred after another. Under this formulation, paths in the graph composed of high-weight edges and high-degree nodes correspond to normal behavior. Then, the Normality of a data series is defined as follows.

Definition 1 (θ-Normality). *Let a node set be defined as $\mathcal{N} = \{N_1, N_2, ..., N_m\}$. Let also a data series T be represented as a sequence of nodes $\langle N^{(1)}, N^{(2)}, ..., N^{(n)} \rangle$ with $\forall i \in [0, n], N^{(i)} \in \mathcal{N}$ and $m \leq n$. The θ-Normality of T is the subgraph $G^{\nu}_{\theta}(\mathcal{N}_\nu, \mathcal{E}_\nu)$ of $G(\mathcal{N}, \mathcal{E})$ with $\mathcal{E} = \{(N^{(i)}, N^{(i+1)})\}_{i \in [0, n-1]}$, such that: $\mathcal{N}_\nu \subset \mathcal{N}$ and $\forall (N^{(i)}, N^{(i+1)}) \in \mathcal{E}_\nu, w((N^{(i)}, N^{(i+1)})).(deg(N^{(i)}) - 1) \geq \theta$.*

Similarly, we define an anomaly as follows.

Definition 2 (θ-Anomaly). *Let a node set be defined as $\mathcal{N} = \{N_1, N_2, ..., N_m\}$. Let a data series T be represented as a sequence of nodes $\langle N^{(1)}, N^{(2)}, ..., N^{(n)} \rangle$ with $\forall i \in [0, n], N^{(i)} \in \mathcal{N}$ and $m \leq n$. The θ-Anomaly of T is the subgraph $G^{\alpha}_{\theta}(\mathcal{N}_\alpha, \mathcal{E}_\alpha)$ of $G(\mathcal{N}, \mathcal{E})$ with $\mathcal{E} = \{(N^{(i)}, N^{(i+1)})\}_{i \in [0, n-1]}$, such that: $G^{\nu}_{\theta}(\mathcal{N}_\nu, \mathcal{E}_\nu) \cap G^{\alpha}_{\theta}(\mathcal{N}_\alpha, \mathcal{E}_\alpha) = \emptyset$.*

We now define the membership criteria of a subsequence to a θ-Normality subgraph.

Definition 3 (θ-Normality Membership). *Given a data series T represented as a sequence of abstract states $\langle N^{(1)}, N^{(2)}, ..., N^{(n)} \rangle$, a subsequence $T_{i,\ell}$, represented by $\langle N^{(i)}, N^{(i+1)}, ..., N^{(i+\ell)} \rangle$, belongs to the θ-Normality of T if and only if $\forall j \in [i, i+\ell], (N^{(j)}, N^{(j+1)}) \in \theta\text{-}Normality(T)$. On the contrary, $T_{i,\ell}$ belongs to the θ-Anomaly of T if and only if $\exists j \in [i, i+\ell], (N^{(j)}, N^{(j+1)}) \notin \theta\text{-}Normality(T)$.*

Based on the above definitions, using θ-Normality subgraphs naturally leads to a ranking of subsequences based on their "normality". For practical reasons, this ranking can be transformed into a score, where each rank can be seen as a threshold in that score. We used such a score in GraphAn to detect abnormal subsequences. Note that given the existence of graph G, the above definitions imply a way for identifying the anomalous subsequences. The problem is now how to construct this graph, and formalized as follows.

Problem 1 (Pattern graph construction). *Given a data series T, we want to automatically construct the graph $G(\mathcal{N}, \mathcal{E})$.*

2.2 Series2Graph Framework

We now briefly describe Series2Graph [7], our unsupervised solution to the subsequence anomaly detection problem. For a given data series T, the overall Series2Graph process is divided into four main steps as follows.

[Subsequence Embedding]: As illustrated in Figure 1(a), we project all the subsequences (of a given length ℓ) of T in a three-dimensional space that corresponds to the three most important components of the Principal Component Analysis (PCA). This space is subsequently transformed into a two-dimensional space, composed of two components $\vec{r_y}, \vec{r_z}$ corresponding to two orthogonal vectors of $\vec{v_{ref}}$, an axis composed of every flat sequence ($a * \mathbf{1}_{\ell-\lambda}$ with $a \in \mathbb{R}$). In the later space, the shape similarity is preserved [7]. For instance, Figure 1 depicts three subsequences T_1, T_2 and T_3. T_1 and T_2 have the same shape and thus the same location in the embedding space, whereas T_3 (which has a different shape) is not.

[Node Creation]: We then create a node for each one of the densest parts of the above two-dimensional space (see Figure 1(b)). The space is first discretized by a set of radius $\mathcal{I}_\psi$ of angle ψ. We then estimate the density of subsequences along each one of these radius using Gaussian kernels. The maximal values of the estimated densities are assigned to nodes and form our Node Set $\mathcal{N}$. These nodes summarize all major patterns of length ℓ that occurred in T [7].

[Edge Creation]: We then retrieve all transitions between pairs of subsequences represented by two different nodes: each transition corresponds to a pair of subsequences, where one occurs immediately after the other in the input data series T. As shown in Figure 1(d), we represent transitions with an edge between the corresponding nodes, and we thus form the Edge Set $\mathcal{E}$. The edge weights are set to the number of times the corresponding pair of subsequences was observed in T. Finally we build our graph $G_\ell(\mathcal{N}, \mathcal{E})$.

[Subsequence Scoring]: We compute the normality (or anomaly) score of a subsequence of length $\ell_q \geq \ell$ (within or outside of T), based on the previously computed edges/nodes and their weights/degrees. Formally, for a subsequence T_{i,ℓ_q} of T, represented by a path in $G_\ell(\mathcal{N}, \mathcal{E})$, $P_{th} = \langle N^{(i)}, N^{(i+1)}, ..., N^{(i+\ell_q)} \rangle$, the normality score is defined as: $Norm(P_{th}) = \sum_{j=i}^{i+\ell_q-1} \frac{w(N^{(j)}, N^{(j+1)})(deg(N^{(j)})-1)}{\ell_q}$, where $w(e)$ and $deg(n)$ are the weight of edge e and the degree of node n, respectively.

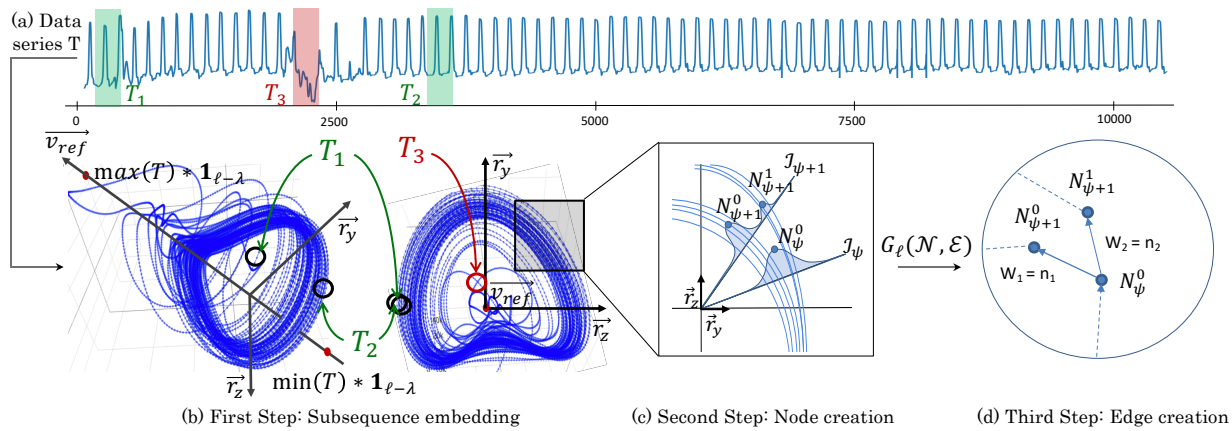


Figure 1: Series2Graph steps in order to build the graph from a data series (a): embed the subsequences (b), create the nodes (c), and extract the edges (d).

3. GRAPHAN OVERVIEW

The GraphAn GUI is a stand alone web application, developed using Python 3.6 and the Dash framework. Figure 2 displays the different frames of GraphAn. The main frame is shown in Figure 2(a). Once the user opens the web application, they can upload a dataset (as well as the anomaly annotations, if available) that will appear as in Figure 2(a.1). The user can then change the values of ℓ and ℓ_q by clicking in the Series2Graph dropdown in the navigation bar in the middle, and subsequently, visualize and rotate/zoom in the embedding space (Figure 2(a.2)) and the resulted graph $G_\ell(\mathcal{N}, \mathcal{E})$ (Figure 2(a.3)). By clicking on the points in the embedding space, the user can visualize the corresponding subsequences (Figure 2(a.2.1)). Similarly, the user can click on a node in the graph in order to see which subsequences belong to it (Figure 2(a.3.1)). Once these steps are performed, the user can perform the Series2Graph anomaly score computation, which will be displayed under the uploaded data series (Figure 2(a.4)). The user can also run other anomaly detection methods: STOMP [9], Isolation Forest [12] (IF) and Local Outlier Factor [8] (LOF). Their anomaly scores will be shown together with the Series2Graph anomaly scores (Figure 2(a.4)). If annotations are provided, a performance analysis can be done by clicking on the performance button: a new frame will appear (Figure 2(b)) displaying accuracy and time execution evaluations. The accuracy evaluation graphs are as follows: a first graph depicting Recall (number of correctly detected anomaly points divided by the total number of anomalous points, shown in Figure 2(b.1)); a second graph depicting Precision (number of correctly detected anomaly points divided by the total number of points detected, shown in Figure 2(b.2)); and a third graph depicting $F1 = \frac{2*Recall*Precision}{Recall+Precision}$ (shown in Figure 2(b.3)). The $F1$ Area Under the Curve (AUC) and the maximal value is computed and summarized for all methods in bar plots (respectively Figure 2(b.5) and Figure 2(b.6)). Finally the execution time for every method is summarized in another bar plot (Figure 2(b.4))).

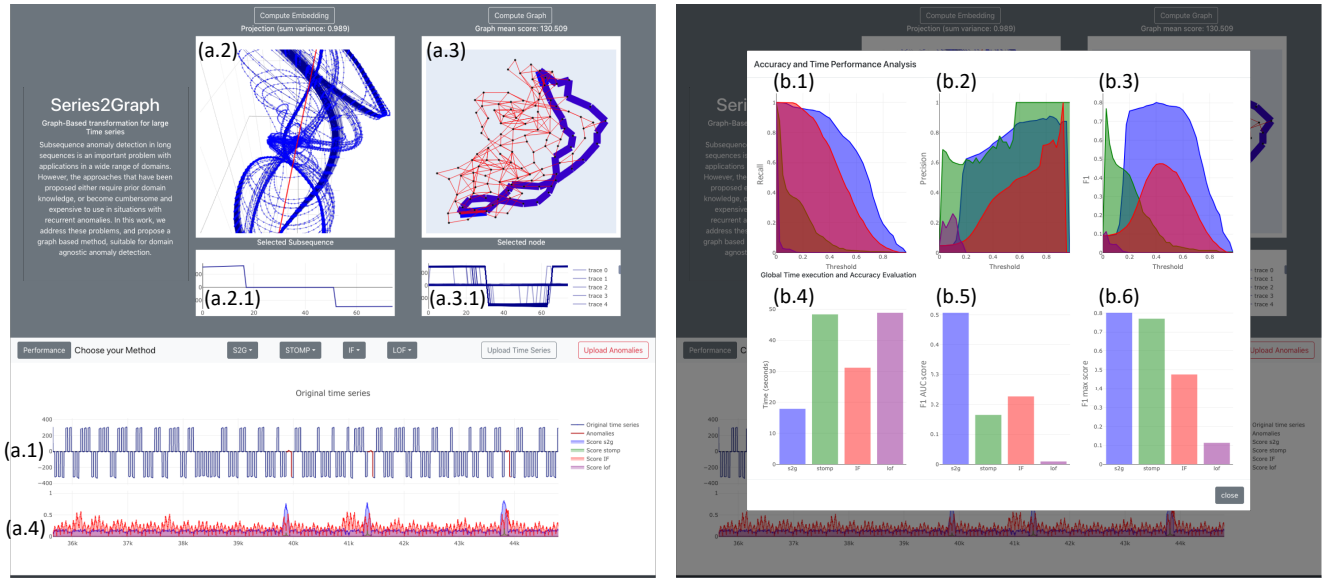
4. DEMONSTRATION SCENARIOS

This demonstration has 4 goals: (i) showcase the effectiveness of GraphAn and compare it to competing approaches in term of anomaly detection accuracy and execution time; (ii) enable the user to understand and interpret the intermediate steps of the Series2Graph method using the uploaded data series, by visualizing the embedding space and the graph; (iii) challenge the user to identify recurrent anomalies by navigating and investigating the nodes of the graph; and (iv) allow the user to appreciate the difficulty of the problem and the usefulness of automated tools, by asking them to try to manually identify subsequence anomalies.

[Scenario 1: Effectiveness] This scenario begins with a long data series (100,000 points with 50 anomalies of length approximately equal to 100 points each) representing simulated engine disks data (SED), provided by the Rotary Dynamics Laboratory at NASA [2]. We will first run Series2Graph by displaying the intermediate, inner steps before finally computing and displaying the anomaly scores for all candidate subsequences. Then, we will run the competing approaches and display their anomaly scores as well. Through the use of the ground truth, we will show that Series2Graph is both faster and more accurate, making it more suitable for long data series analysis.

[Scenario 2: System Internals] The second scenario will allow the user to examine the Series2Graph inner workings. Directly through the main frame, the user will be able to visualize both the embedding space (by being able to zoom, move and rotate it) and the computed graph. Then, the user will also be able to inspect the embedded subsequences, by clicking on the points in the embedded space representation in order to visualize the corresponding subsequences of the original series. The user will be able to apply the same process on the graph, by checking which subsequences correspond to each one of the nodes in the graph. Finally, by selecting a subsequence in the original data series, the corresponding path in the graph will be highlighted.

[Scenario 3: Discovering Recurrent Subsequences] The third scenario will focus on the analysis of the discovered anomalies. This task will go beyond the anomaly detection task, to finding (the positions of) all subsequences of the original series that are similar to the detected anomaly. The advantage that the graph structure provides to the user is that it groups similar subsequences under the same node (vertex). Thus, with a single click on one of the graph nodes, the user can see both the anomaly score of the subse-



(a) Screenshot of GraphAn

(b) Screenshot of GraphAn when Performance button is pressed

Figure 2: GraphAn screenshots. (a) Main frame with data series (a.1), embedded space (a.2), and graph (a.3). (b) Performance frame with accuracy (b.1-3,5-6) and execution time (b.4).

quence, as well as all similar subsequences of the same data series. Therefore, in this scenario, the user will be able to navigate through the graph, investigate subsequences (infrequent/frequent, anomalous/normal) and visualize when and how many times they occurred.

[Scenario 4: Manual Anomaly Detection] The last scenario begins with two datasets. The first dataset is the New York City Taxi and Limousine Commissions dataset (NTC)[2] (10,000 points with 8 anomalies); the second dataset is record 820 of the MIT-BIH Supraventricular Arrhythmia Database (100,000 points with 76 anomalies). We will challenge participants to look for and identify anomalies in these datasets. The participants will be able to visualize the entire sequences, as well as zoom in/out and pan left/right. This exercise will help participants appreciate the difficulties and challenges of subsequence anomaly detection, especially when there are multiple anomalies, when these anomalies are subtle, and when the overall size of the sequence is large.

5. CONCLUSIONS

We proposed GraphAn, a novel, unsupervised system for subsequence anomaly detection that is based on the representation of the data series into a directed graph. This graph summarizes and highlights crucial information that enables us to detect and group together abnormal subsequences. In our future work, we plan to compare GraphAn to SAD [6].

Acknowledgments

Work partially supported by EDF R&D and ANRT French program.

References

[1] D. Abboud, M. Elbadaoui, W. Smith, and R. Randall. Advanced bearing diagnostics: A comparative study of two powerful approaches. *MSSP*, 114, 2019.

[2] A. Abdul-Aziz, M. R. Woike, N. C. Oza, B. L. Matthews, and J. D. lekki. Rotor health monitoring combining spin tests and data-driven anomaly detection methods. *SHM*, 2012.

[3] J. Antoni and P. Borghesani. A statistical methodology for the design of condition indicators. *MSSP*, 2019.

[4] V. Barnet and T. Lewis. *Outliers in Statistical Data*. John Wiley and Sons, Inc., 1994.

[5] P. Boniol, M. Linardi, F. Roncallo, and T. Palpanas. Automated anomaly detection in large sequences. In *ICDE*, 2020.

[6] P. Boniol, M. Linardi, F. Roncallo, and T. Palpanas. SAD: an unsupervised system for subsequence anomaly detection. In *ICDE*, 2020.

[7] P. Boniol and T. Palpanas. Series2graph: Graph-based subsequence anomaly detection for time series. *PVLDB*, 13(11), 2020.

[8] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. In *SIGMOD*, 2000.

[9] Y. Z. et al. Matrix profile II: exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins. In *ICDM 2016*.

[10] M. Hadjem, F. Naït-Abdesselam, and A. A. Khokhar. St-segment and t-wave anomalies prediction in an ECG data using rusboost. In *Healthcom*, 2016.

[11] M. Linardi, Y. Zhu, T. Palpanas, and E. J. Keogh. Matrix Profile Goes MAD: Variable-Length Motif And Discord Discovery in Data Series. In *DAMI*, 2020.

[12] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *ICDM*, ICDM, 2008.

[13] T. Palpanas. Data series management: The road to big sequence analytics. *SIGMOD Rec.*, 44(2):47–52, Aug. 2015.

[14] T. Palpanas and V. Beckmann. Report on the First and Second Interdisciplinary Time Series Analysis Workshop (ITISA). *ACM SIGMOD Record*, 48(3), 2019.

[15] P. Senin, J. Lin, X. Wang, T. Oates, S. Gandhi, A. P. Boedihardjo, C. Chen, and S. Frankenstein. Time series anomaly discovery with grammar-based compression. In *EDBT*, 2015.

[16] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online outlier detection in sensor data using non-parametric models. In *VLDB 2006*, pages 187–198, 2006.

[17] L. Wei, E. J. Keogh, and X. Xi. Saxually explicit images: Finding unusual shapes. In *ICDM*, 2006.

[18] D. Yankov, E. J. Keogh, and U. Rebbapragada. Disk aware discord discovery: finding unusual time series in terabyte sized datasets. *Knowl. Inf. Syst.*, 17(2):241–262, 2008.

[19] C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. J. Keogh. Matrix profile I: all pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In *ICDM*, pages 1317–1322, 2016.

[2] `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`