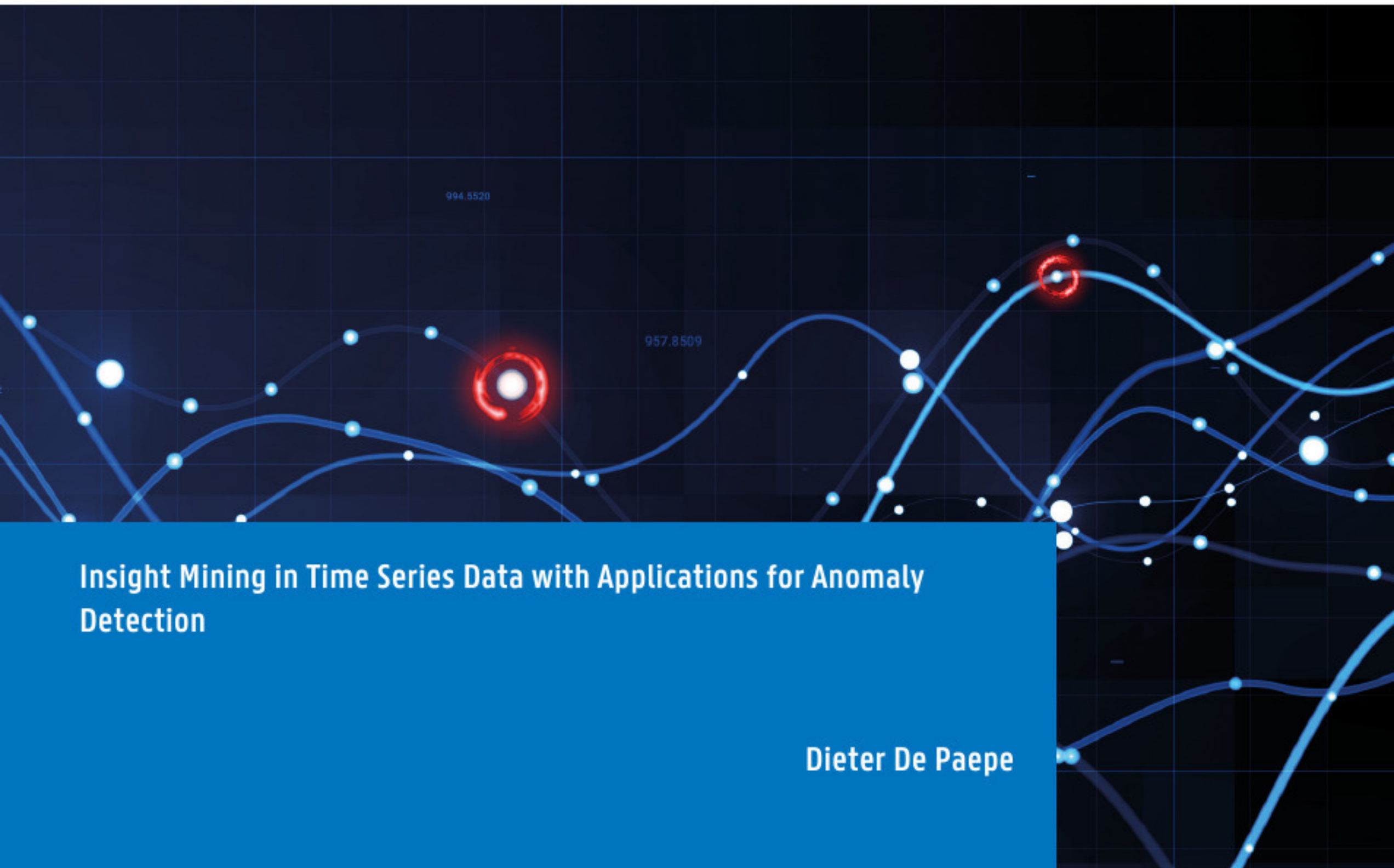
FACULTY OF ENGINEERING
AND ARCHITECTURE



Insight Mining in Time Series Data with Applications for Anomaly Detection

Dieter De Paepe

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Computer Science Engineering

Supervisors

Prof. Sofie Van Hoecke, PhD - Prof. Erik Mannens, PhD

Department of Electronics and Information Systems
Faculty of Engineering and Architecture, Ghent University

June 2021

GHENT
UNIVERSITY

FACULTY OF ENGINEERING
AND ARCHITECTURE

Insight Mining in Time Series Data with Applications for Anomaly Detection

Dieter De Paepe

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Computer Science Engineering

Supervisors
Prof. Sofie Van Hoecke, PhD - Prof. Erik Mannens, PhD

Department of Electronics and Information Systems
Faculty of Engineering and Architecture, Ghent University

June 2021

GHENT
UNIVERSITY

ISBN 978-94-6355-497-8
NUR 984
Wettelijk depot: D/2021/10.500/45

Members of the Examination Board

Chair

Prof. Patrick De Baets, PhD, Ghent University

Other members entitled to vote

Prof. Tom Dhaene, PhD, Ghent University
Olivier Janssens, PhD, Colruyt Group
Prof. Eamonn Keogh, PhD, University of California, USA
Prof. Femke Ongenae, PhD, Ghent University
Joeri Ruyssinck, PhD, ML2Grow

Supervisors

Prof. Sofie Van Hoecke, PhD, Ghent University
Prof. Erik Mannens, PhD, Ghent University

Preface

> One of life's greatest blessings is the freedom to pursue one's goals.
>
> Ryoji Mochizuki

I certainly did not follow the standard path to obtain a PhD degree, if such a path even exists. When I graduated eleven years ago, I specifically decided not to pursue one because of how much I struggled to put my thoughts into writing. And yet, somehow, this is how it turned out. Six years ago, I applied for a job at Ghent University. I mainly wanted to gain experience with new technologies and the university seemed like the best place for that. Prof. Erik Mannens offered me this chance, with his enthusiasm and positive energy, he convinced me to join the semantic web research group in IDLab. A few months had passed and Erik convinced me to start a PhD and see where it leads me. Very soon I would submit my first papers and visit several conferences.

Two years later, my interests had shifted towards machine learning, and I joined the research group of Prof. Sofie Van Hoecke, who became my main promoter. Sofie was a great mentor and is most certainly the reason I was able to finish this PhD. I'm sure there have been multiple times where she had more faith in me than I did myself. The amount of dedication she puts in her work and team is staggering, and for that she has earned my deepest respect. Sofie, you're a gem.

To the members of the examination board: Prof. Patrick De Baets, Prof. Femke Ongenae, Prof. Tom Dhaene, Prof. Eamonn Keogh, Dr. Olivier Janssens and Dr. Joeri Ruyssinck, thank you for your time and the acknowledgement of my work. Your feedback helped this book reach its final form.

Over the years, I got to know many wonderful colleagues. A big thank you for any help along the way and the fun times spent together. From the semantic web team: Ruben V., Joachim, Dieter DW., Anastasia, Pieter C., Pieter H., Ben, Gerald, Martin, Sven, Ruben T., Brecht, Julian, Doërthe and Laurens. From the Predict team: Sander, Tamir, Diego, Nathan, Colin, Emiel, Jonas, Marija, Pieter, David, Matthias, Emile, Jarne and Jeroen. From other teams: Hannes, Martijn, Glenn, Johan, Steven, Florian, Bram, Azarakhsh, Davi, Geoffroy, Laura and Kristof.

Thanks to all my friends, old or new, for playing a part in my life, and allowing me to play a part in yours.

I also want to show my appreciation to my family. Mom, dad, Davy, you helped me become who I am today. You supported me when needed, and gave me the freedom to find my own way.

And finally, Tatjana, thank you for being a beacon of light in my life.

Dieter De Paepe
June 2021

Contents

Summary

Sensors are affecting ever more facets of our personal lives. Personal activity trackers monitor our exercise and sleeping regimes, weather forecasts are based on detailed atmospheric measurements, and car sensors use a wide variety of colorful indicators to inform us when something needs attention. Similarly, sensors become more widespread in industry as well. Machine sensors can determine when maintenance is due, and virtual sensors help secure the online services we depend on everyday. Thanks to the decreasing cost of data storage and the rise of cloud service providers data can now be stored in higher quantities and higher resolutions. Of course, the goal of companies is to use data analytics to gain insights and improve their products and services.

A wide range of data analytics techniques exists to gain insights from data. Thanks to the finer resolution and higher detail in measurement data, we can now treat more and more data as (time-)series, where data is a sequence of evolving measurements rather than a collection of independent data points, opening a whole new world of data analytics techniques to be explored. This dissertation discusses various data analytics methods for working with series data, focusing on pattern-based techniques. Visualisations show similarity throughout time, anomaly detection flags suspicious patterns, and repeating patterns can be easily found. All methods deal with insight extraction in some way and can be applied to a wide range of domains.

Chapter 1 familiarizes the reader with relevant concepts to understand the later chapters. We discuss the four categories of data analytics which we will use to situate our contributions and divulge on the characteristics of (time-)series and anomaly detection. Next, we provide an introduction to a series processing technique, i.e. the Matrix Profile (MP), which is a state-of-the-art time series analysis technique and forms an important building block in our techniques presented in later chapters. Finally, we provide an overview of our publications.

In Chapter 2 we present a method for detecting poor quality welds in a steel mill production line using welding current data. Here, we use a classical approach to incorporating time series data, i.e. we calculate statistical features from the measurements obtained during the welding process. These features are combined with metadata describing the characteristics of the steel coils to train a model that predicts the welding current. Because we use a hybrid model, i.e. one where we combine machine learning

with knowledge of physical laws, the parameters of the trained model give insight into the resistivity factors for each type of steel. Furthermore, this model can automatically adjust to sudden or gradual changes of the process, attributed to machine maintenance and slight variations in the chemical composition of welded steel.

Chapter 3 marks the start of our pattern-based approach to time series. It introduces the Series Distance Matrix (SDM) framework, whose ideas and code base form the foundation for later chapters. SDM is a type of plug-and-play framework that allows freely combining various data analytics techniques related to Matrix Profile. It highlights the utility of the distance matrix as a core functionality upon which other techniques can be based. In this way, we treat the traditional Matrix Profile as one specific setup of SDM. This chapter also introduces the Contextual Matrix Profile (CMP) as a new technique that fits within SDM. Whereas the Matrix Profile represents a one-dimensional reduction of the distance matrix, the CMP represents a two-dimensional reduction. We demonstrate the utility of the CMP for visualization purposes, where it can highlight trends throughout time, and for anomaly detection, by finding deviations from those trends.

Next, Chapter 4 focuses on the use of the z-normalised Euclidean distance as distance measure in SDM, which is the original similarity measure used for the MP as it mainly compares the shape of subsequences. We analyze the link with the Pearson correlation, which allows us to normalize the distance and allows comparing distances when using different subsequence lengths. An undesired side effect of this measure is seen when data contains flat subsequences and has some amount of noise present, two properties that are quite common in realistic data. In this case, reported distances will differ greatly from human intuition and hinders the use of SDM techniques. By estimating the impact of this effect on the distances, we can easily negate this effect. We show the benefit of this approach for three different use cases and data sets, i.e. visualization, semantic segmentation and anomaly detection.

Finding repetitions is an a valuable insight when working with series. Where the MP and CMP focus on finding the single best repetition in series, one recent algorithm finds the (top-k) best repetitions over a collection of series. In Chapter 5, we extend this technique and define the Radius Profile (RP), as another component that fits into the SDM framework. The RP gives insight into how well every subsequence is repeated over a collection of series. The common-k Radius Profile is yet another proposed variation for finding repetitions when working with a single series rather than multiple. This technique can be used to find the most common patterns in a series, i.e. the subsequences that are best preserved over a number of repetitions.

Where the previous chapters mainly treat data analytics methods as independent techniques, Chapter 6 shows how these methods can be used as part of a larger data processing system. It describes a use case from the Dyversify project, where a full stack data analytics prototype was built that combines machine learning (data-driven) techniques with semantic (knowledge-driven) techniques. The goal was to automatically detect anomalies or events in data originating from ventilation units. A dynamic dashboard matches the semantically enriched anomalies with suitable visualisation widgets

using semantic reasoning, thereby reducing the workload of its operator. Through interaction with the dashboard, the operator can inspect these anomalies and indicate which anomalies are relevant for them. The first anomalies detected by this system will be discords, i.e. unique patterns that can be seen as potential anomalies, and are found by a streaming MP component. When patterns occur a second time, they are no longer detected as discords. However, if the operator indicated the corresponding pattern as relevant, a pattern detector will report an event for the second occurrence. The detection components utilise these confirmed anomalies as pattern templates to detect matching events in future data. Chapter 6 describes the overarching architecture of this system. It details how data is ingested in the Obelisk system, transported using Kafka, semantified using the RMLStreamer, processed using the SDM framework and semantic processing techniques, and ultimately reaches the dashboard where user interaction feeds back into previous components. This chapter also bundles our experiences with combining semantic and machine learning techniques.

We summarize our research in Chapter 7 and discuss directions for future work.

Samenvatting

Sensoren beïnvloeden meer en meer aspecten van ons persoonlijk leven. Persoonlijke activiteitstrackers monitoren onze beweging- en slaapregimes, weersvoorspellingen zijn gebaseerd op gedetailleerde atmosferische metingen, en sensoren in auto's gebruiken allerlei kleurrijke indicatoren om ons te informeren dat iets onze aandacht vereist. Het gebruik van sensoren in de industrie neemt eveneens toe. Sensoren in machines kunnen bepalen wanneer onderhoud nodig is en virtuele sensoren dragen bij tot het beveiligen van de online diensten waar we dagelijks gebruik van maken. Dankzij de dalende kosten van gegevensopslag en de opkomst van cloud service providers kunnen gegevens nu in grotere hoeveelheden en hogere resoluties opgeslagen worden. Het doel van bedrijven is uiteraard om data-analyse te gebruiken om inzichten te verkrijgen en producten en diensten te verbeteren.

Er bestaat een breed scala aan data-analysetechnieken om inzichten uit data te extraheren. Dankzij de fijnere resolutie en betere detail in meetgegevens kunnen we nu meer data als tijdsreeksen behandelen, hier bestaan metingen uit een reeks evoluerende waarden in plaats van een verzameling onafhankelijke datapunten. Dit opent een hele nieuwe wereld aan data-analysetechnieken om te ontdekken. Dit proefschrift bespreekt verschillende data-analystechnieken voor het werken met tijdsreeksen, met een nadruk op patroongebaseerde technieken. Visualisaties laten toe om gelijkenissen doorheen de tijd te vinden, anomaliedetectie kan gebruikt worden om afwijkende patronen te vinden, en andere technieken dienen om repetitieve patronen te vinden. Alle besproken technieken hebben te maken met het verwerven van inzichten en zijn niet gebonden aan één bepaald domein.

Hoofdstuk 1 maakt de lezer vertrouwd met relevante concepten om de latere hoofdstukken te begrijpen. We bespreken vier categorieën van data-analyse die we zullen gebruiken om onze bijdragen te situeren, verder gaan we in op de kenmerken van tijdsreeksen en anomaliedetectie. Vervolgens verdiepen we ons in technieken rond seriesverwerking waaronder de zogenaamde Matrix Profile. Dit is een state-of-the-art techniek voor tijdsreekanalyse en vormt een belangrijke basis voor de technieken die in latere hoofdstukken worden voorgesteld. Ten slotte geven we een overzicht van onze publicaties.

In Hoofdstuk 2 stellen we een methode voor die lassen van slechte kwaliteit kan detecteren in productielijnen van staalfabrieken, gebaseerd op de gemeten lasstroom.

Hier gebruiken we een klassieke benadering om met tijdreeksgegevens om te gaan: we gebruiken statistische eigenschappen van de metingen tijdens het lasproces. Deze statistische waarden worden gecombineerd met metadata van de stalen spoelen om een model te trainen dat de lasstroom voorspelt. Omdat we een hybride model gebruiken, namelijk een model waarbij we machine learning combineren met kennis van fysische wetten, geven de parameters van het getrainde model inzicht in de weerstandsfactoren voor elk type staal. Bovendien kan dit model zich automatisch aanpassen aan plotselinge of geleidelijke veranderingen in het gedrag van het lasproces, dergelijke veranderingen worden toegeschreven aan machineonderhoud en kleine variaties in de chemische samenstelling van het staal.

Vanaf Hoofdstuk 3 leggen we de nadruk op technieken die gebruik maken van patronen om tijdsreeksen te analyseren. We introduceren het Series Distance Matrix (SDM) framework, waarvan de ideeën en codebasis de fundamenten vormen voor latere hoofdstukken. SDM is een plug-and-play framework waarmee verschillende data-analysetechnieken gerelateerd aan de Matrix Profile vrij kunnen worden gecombineerd. Het kern idee van SDM ligt in de erkenning van het nut van de afstandsmatrix als kernfunctionaliteit waarop andere technieken kunnen worden gebaseerd. Op deze manier kunnen we de traditionele Matrix Profile zien als één specifieke opstelling van SDM. In dit hoofdstuk wordt ook het Contextual Matrix Profile (CMP) geïntroduceerd als een nieuwe techniek die past binnen SDM. Waar de Matrix Profile kan gezien worden als een eendimensionale reductie van de afstandsmatrix, gebruikt de CMP een tweedimensionale reductie. We demonstreren het nut van de CMP voor visualisatiedoeleinden, waar het trends doorheen de tijd kan blootleggen, en voor anomaliedetectie door afwijkingen van die trends te vinden.

Vervolgens richt Hoofdstuk 4 zich op het gebruik van de z-genormaliseerde Euclidische afstand als afstandsmetriek in SDM. Aangezien deze metriek erg geschikt is om de vorm van sequenties te vergelijken, is dit tevens de oorspronkelijke metriek die werd gedefinieerd voor de Matrix Profile. We onderzoeken de link met de correlatiecoëfficiënt, waardoor we afstanden kunnen normaliseren en zo onderling afstanden kunnen vergelijken tussen sequentieparen van verschillende lengtes. Een ongewenst neveneffect van deze afstandsmetriek komt voor wanneer tijdsreeksen vlakke sequenties bevatten en er enige hoeveelheid ruis aanwezig is. Aangezien deze twee kenmerken vrij vaak voorkomen in realistische gegevens, heeft dit neveneffect een grote impact. Meer bepaald zullen de gerapporteerde afstanden sterk verschillen van de menselijke intuïtie de optimale werking van SDM-technieken belemmeren. Door de impact van ruis op de z-genormaliseerde Euclidische afstand in te schatten, kunnen we dit effect gemakkelijk corrigeren. We demonstreren de voordelen van deze techniek voor drie verschillende use-cases en datasets, namelijk visualisatie, semantische segmentatie en anomaliedetectie.

Kennis over herhalende patronen is een waardevol inzicht bij het werken met tijdseries. Waar de MP en CMP zich richten op het vinden van de meest gelijkaardige herhalingen in tijdseries, focust een recent werk zich op het vinden van de (top-k) beste herhalingen over een verzameling tijdseries. In Hoofdstuk 5 breiden we deze

techniek uit en definiëren we de Radius Profile (RP) als een nieuwe component die past in het SDM-framework. De RP geeft inzicht in hoe nauwkeurig elke sequentie in een tijdsreeks een aantal keer herhaald is doorheen een verzameling tijdsreeksen. Het common-k Radius Profile is nog een andere voorgestelde techniek voor het vinden van herhalingen bij één enkele tijdsreeks in plaats van meerdere. Deze techniek kan worden gebruikt om de meest voorkomende patronen in een tijdsreeks te vinden, d.w.z. de sequenties die het best bewaard blijven over een aantal herhalingen.

Waar in de voorgaande hoofdstukken data-analysemethoden voornamelijk als onafhankelijke technieken worden behandeld, laat hoofdstuk 6 zien hoe deze methoden kunnen worden gebruikt als onderdeel van een groter data-analyse systeem. Dit hoofdstuk beschrijft een use-case uit het Dyversify project, waar een full-stack data-analyse prototype werd gebouwd dat data-gedreven machine learning technieken combineert met kennis-gedreven semantische technieken. Het doel was om automatisch anomalieën of voorvallen te detecteren in gegevens afkomstig van particuliere ventilatie eenheden. Een dynamisch dashboard koppelt semantisch verrijkte anomalieën met geschikte visualisatiewidgets met behulp van semantisch redeneren, waardoor de werklast van de operator wordt verminderd. Via interactie met het dashboard kan de operator deze anomalieën onderzoeken en aangeven welke anomalieën relevant zijn. De eerste detecties door dit systeem zijn zogenaamde discords, d.w.z. unieke patronen in de tijdsreeks die kunnen worden gezien als mogelijke anomalieën. Deze worden gedetecteerd door een MP-component die streaming data verwerkt. Wanneer patronen een tweede keer voorkomen, worden ze niet langer als discord beschouwd. Echter, als de operator het overeenkomstige patroon als relevante anomalie heeft gelabeld, zal een patroondetector dit patroon registreren als een relevant voorval. De detectiecomponenten gebruiken dus de bevestigde anomalieën als referentiepatronen om overeenkomende voorvallen in nieuwe binnenkomende metingen te detecteren. Hoofdstuk 6 beschrijft de overkoepelende architectuur van dit systeem. Het beschrijft hoe gegevens ontvangen worden in het Obelisk-systeem, getransporteerd worden met behulp van Kafka, semantisch geconverteerd worden door de RMLStreamer, verwerkt worden met behulp van het SDM-framework en semantische regelgebaseerde technieken, en uiteindelijk het dashboard bereiken waar interacties van gebruikers teruggekoppeld worden naar eerdere componenten. Dit hoofdstuk bundelt ook onze ervaringen met het combineren van semantische en machine learning technieken.

Hoofdstuk 7 bevat een samenvatting van ons onderzoek en bespreekt mogelijkheden voor toekomstig onderzoek.

Chapter 1

Introduction

"Data is the new oil". A statement often repeated by business leaders when outlining their newest business strategy, where they will focus more on gathering and analyzing data. Data analytics by itself is as old as mankind and was used to learn about the laws of physics long before the first computers were around. With the rise of computers, data analytics was able to further mature in research facilities and tech companies. The big tech companies we know today, such as Facebook, Amazon or Google, can attribute their success to early adoption of data analytics.

Focusing on the present, we find more and more companies incorporating data analytics into their core business. Data storage has cheapened and cloud provider services further lowered the risk of initial investment. This has allowed businesses to measure and store a wider variety of metrics, and has even enabled the rise of a whole new range of smart devices whose functionality relies on data exchange and analysis, the so called Internet of Things (IoT). As the amount of captured data that can be analyzed increases, data processing capabilities have also significantly improved, and continue to do so. Though Moore's law is starting to reach its limit, hardware will become faster by exploiting 3D architectures, parallelization and specialisation [1]. On the software side, both academia and industry continue to find new or improve existing data analytics techniques. Finally, the human factor remains an important one as well. Here, we see the increasing popularity of data science resources, communities (e.g.: Kaggle), and the hailing of data science as the *sexiest job of the 21st century* [2]. All of these observations clearly indicate the high expectations and growing importance of data analytics

1.1 Data Analytics

Data analytics is a broad term that covers a range of practices, in essence data analytics comes down to trying to make sense of data in some way. Analytics can be subdivided into four categories, each with their own focus. To highlight the differences between

these categories, we exemplify each category with the use case of examining black box data from a crashed airplane.

Descriptive Analytics focuses on knowing *what* is happening in the available data. This includes understanding every part of a data set, summarizing it to an understandable format, and being aware of any changes that are captured in it. Common techniques are summary statistics, visualizations, statistical methods and data mining, which are often used jointly and iteratively to refine initial assumptions and adjust techniques accordingly. It is often useful to visualize data in different ways, as each different perspective may reveal new findings [3]. Visualizations make the data more natural for the human mind to comprehend and therefor easier to identify trends. In fact, visualizations may reveal patterns that are easily missed by investigating statistical properties [4]. Visualizations are also a great way to highlight differences between sets, for example, taken over different years. We should note that understanding the data to some degree is a requirement for each category, though a more lightweight exploratory data analysis can also suffice in the other cases. In our example use case, descriptive analytics would focus on mapping and understanding the state of all airplane systems throughout the recorded flight time.

Diagnostic Analytics tries to explain *why* one or more events in the data occurred. This comes down to trying to find the chain of relevant events in the data, the process is similar to root cause investigations in domains with strict safety regulations such as healthcare or aviation (though these are typically done through interviews). Statistical methods such as correlations or data mining can be used to find dependencies or co-occurring events in the data. Regression methods can be used to find temporal dependencies in signals [5], whereas anomaly detection can help locate unexpected behavior that might be part of the event chain and event detection methods can reduce the amount of data to a more manageable set of concrete events. To emphasize that techniques can never prove causality, the term *Granger causality* is typically used in literature. In the aviation use case, diagnostic analytics tries to find the series of events that lead to the plane crash.

Predictive Analytics aims to create a model of the underlying system that can *predict* system behavior. Though this somewhat implies future predictions, predictions can equally be used to fill in unknown data of the past or present. A wide range of machine learning methods can applied here. Regression methods such as linear models, neural networks or random forests are useful to predict continuous or ordered values, whereas classification methods such as logistic regression, support vector machines or clustering methods aim to predict discrete categories, i.e. one option from a limited set. Two major assumptions made by all techniques are that available data is representative for unavailable data and that relevant metrics needed for the prediction are captured in the data. In our example, we would obtain a model that captures how failing airplane systems affect each other, so one could run simulations across multiple scenarios using this model.

Prescriptive Analytics deals with the best *actions* to take. This category focuses on combining business intelligence with insights in order to determine the best suitable

actions. Techniques that can express uncertainty are especially useful here, as they can be used for risk analysis where the cost and likelihood of different scenarios is quantified. Examples include timing marketing campaigns, investments, or in our aviation case, finding the most economical way to prevent future crashes.

While each category has a clear theoretical focus, categories will often overlap in practice because insights from one category can be reused in another. For example, the patterns (descriptive) or causalities (diagnostic) found in data can be extrapolated to future behavior and used as a predictive model. Another common example is how automated root cause analysis techniques, a common research topic for diagnosing problems (diagnostic) in complex IoT environments, do this by classifying the most likely cause (predictive). As a final example, consider how the planning of marketing campaigns in a new location (prescriptive) also relies on knowing the demography of candidate locations (descriptive) and being able to predict adoption for each (predictive).

In this dissertation, we touch all four types of analytics, but mainly focus on descriptive analytics methods applicable to time series. But what differentiates time series from other data?

1.2 Time Series

Most data used in machine learning can be represented in a two-dimensional matrix, often called the *feature matrix*. One dimension corresponds the different observations made, the other dimension represents the different features or attributes that were measured for that observation. As an example, assume we want to study our driving behavior and collect a dataset of all rides made with our car in the current year. Here, one observation would describe a single ride. The features would be chosen in function of what exactly we want to study, e.g. some interesting features could be: time of departure, time of arrival, fuel consumption and the number of passengers. Sometimes, additional features can be derived afterwards as well. For example, we can derive the travel time of each ride based on departure and arrival time.

Time series add a third dimension to this feature matrix, the time dimension. This means we can track a single feature of a single observation throughout time, capturing how the feature evolves throughout the observation rather than being restricted to a single value[1]. Coming back to our example, we could record our driving speed throughout every car ride. Due to practical limitations, we typically use *sampling* to get a subset of the data by periodically measuring our feature, where the sampling frequency again depends on the intended use case. Time series can contain continuous values (e.g. driving speed) as well as discrete values (e.g. the radio station we listen to). Every feature can be represented as a time series, though not all features evolve over time. For example, if we assume that any type of passenger exchange indicates

[1] Time series are also often defined as a series of instant observations. However, by assuming observations are non-instant, we better retain the similarity between traditional and time series data.

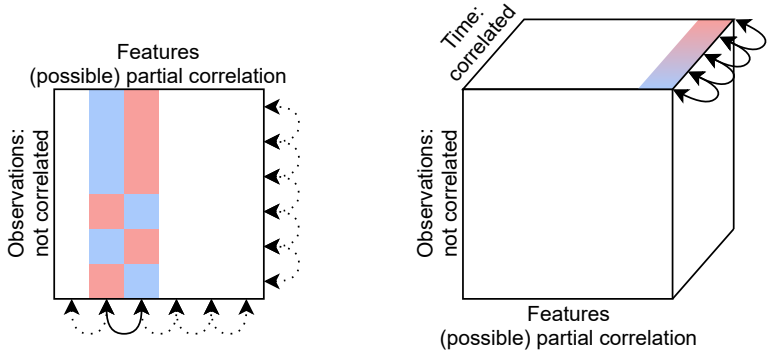


Figure 1.1: Feature matrix for traditional data (left) and time series data (right). Features represent the attributes being measured, and may or may not be correlated, as represented by the arrows (e.g. humidity and temperature are often correlated). Observations represent different measurements and are assumed to be uncorrelated (e.g. measurements in different locations). Finally, the time (or spatial) dimension represents a continuous measurement where measurement values are assumed to be correlated (e.g. the current temperature will be very similar to the temperature recorded one minute before).

the end of a ride, the number-of-passengers feature will remain constant for the entire ride, and there is little value to represent it as a time series.

While we use the term "time series", it is possible to obtains series without using a temporal dimension. For example, we can obtain series by measuring a feature along a spatial dimension (e.g. the cross-sectional thickness of a steel plate) or even by transforming object outlines in still images to series [6]. However, most series are temporal and the term has a strong foundation in literature.

The values in time series observations differ from traditional observations in two ways. First, they have an explicit ordering and secondly, whereas observations are independent from each other, the time series values are typically correlated with nearby values (i.e. each value is similar to nearby values). A summarizing visualization is shown in Figure 1.1. A wide range of techniques exist to incorporate time series into machine learning [7], we discuss two main categories, i.e. statistical features and patterns. This dissertation uses both approaches, but discusses pattern-based techniques more in-depth.

1.2.1 Statistical Features

One common way to incorporate time series data in traditional machine learning techniques is by converting the time series values into statistical features, such as the minimum, standard deviation, mean, skewness, and so on. The choice of which statistical

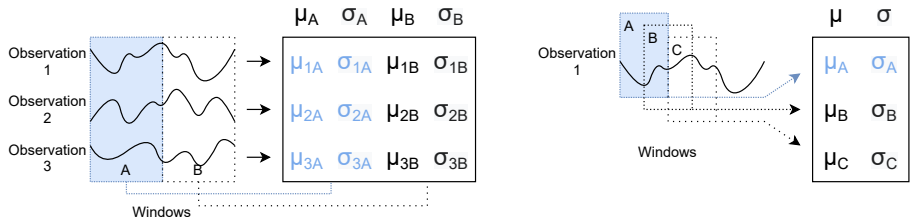


Figure 1.2: Different window-based feature calculation methods for statistical features μ and σ. Left: Windows are defined over ranges of the series. This approach is more common when analyzing series as a whole (e.g.: to find anomalous series). Right: Use of sliding windows. This approach is used when analyzing the progress of series (e.g.: to predict future values).

feature to use will depend on the target use case. For example, we could convert the driving speed time series into an average speed and standard deviation if we are interested in predicting our fuel consumption, or into a maximum speed if we are interested in predicting our likelihood of getting a ticket. As another example, the time of departure and arrival could be seen as the minimum and maximum of the "current time" time series.

Instead of calculating statistical values over the whole series, one can also split the series into different parts (windows) and calculate statistical values for each part, as shown in Figure 1.2 (left). This can be done time-wise (e.g. the mean over the first half and second half of the series), but also based on different criteria. For example, we could calculate the average driving speed per type of road (e.g. offroad, local road, highway) we are traversing instead of having a less meaningful overall average driving speed. One limitation does play a part here, namely that many machine learning techniques assume that each considered observation has an equal number of features. This means we have to define the considered windows appropriately, as to not violate this assumption.

Instead of purely (descriptive) analytics of statistical features over (different) time series, we can go one step further and also perform predictive analytics within a single series using those statistical features. For example, given a single time series we can try to predict the next values, or determine whether the next value is anomalous or not. A common approach to include statistical features in this case is by using sliding windows. Here, a sliding window typically has a fixed size and contains the most recent measured values. For example, in regression cases the sliding window will include the measured values preceding the value to be predicted. This principle is shown in Figure 1.2 (right).

One limitation of most statistical values is that they treat all data points independently. In a way, they disregard the information captured in the order in which the data points occur when applied to time series. Some techniques, like auto-correlation, can be seen as statistical values that work over ordered data, but also as pattern-based

techniques, which are discussed next.

1.2.2 Patterns

A different way to incorporate time series in machine learning is by using patterns to extract information from the series. Here, patterns are short sequences of ordered values that may or may not (exactly) match a part of the time series. Similar as to how humans can easily recognise and extract individual heartbeats from an electrocardiogram (EGC) signal, automatic techniques can compare and identify patterns in time series.

To automatically match patterns, we need *distance measures* that capture how much one sequence of values differs from another. Lower distances indicate a better match, with zero indicating the best possible match. Note that as we use distance measures rather than distance metrics, the best possible match does not necessarily indicate that the two sequences are exactly equal). We present a select few of commonly used measures, though many more exist [8], each having their own use cases:

- The **Manhattan distance** (also known as the taxicab distance or L1-distance) simply sums the difference between all points. This measure is suitable for working with time series where the scale and zero point of the signal are constant.
- The well known **Euclidean distance** (or L2-distance) equals the root of the squared sums of the differences. When matching sequences, this means that larger point-wise differences will have more effect on the resulting distance. This measure can be used for the same series as the Manhattan distance, but emphasizes more on extreme differences.
- The **cosine distance** is based on the dot product of both normalized sequences. Alternatively, when treating the sequences as vectors, this distance equals the Euclidean distance between the normalized vectors. An interesting property of this distance measure is that it disregards the positive scaling of sequences. This measure allows comparison when dealing with time series where the scale may vary, for example due to miscalibrated sensors.
- The **z-normalized Euclidean distance** consists of the Euclidean distance between the z-normalized (mean = 0 and standard deviation = 1) sequences. This measure disregards positive scaling and shifting when comparing sequences, effectively focusing on the shape of both sequences. This measure is suited for comparing time series where the zero-point and scale may vary, as is common in biological signals or in cases where sensor drift may arise.
- **Dynamic time warping** (DTW) is a meta-measure that is often combined with the Euclidean or Manhattan distance. DTW compares both sequences while allowing non-linear shifts in either sequence as to minimize the resulting distance.

It can be used when the speed of the measured behavior may vary throughout time.

Given a specific pattern and distance measure, two approaches to convert series to features are commonly used. We can slide the pattern over a series, calculating the distance for each location, and use the minimum of these distances as a feature, this corresponds to the distance of the best match in the series. When dealing with predictive analytics, we can also take the distances over the most recent values instead, as was described for the statistical features.

While it is possible to use traditional techniques by converting time series to features, specialised techniques have found other ways to derive insights from series. Many of these techniques are also pattern-based, and use information such as the location and quantity of pattern matches.

1.3 Visualizations & Insight Mining

Many insights related to time series deal in some degree with repetition. Previously unknown repetitions may reveal an unexpected but interesting structure, while locating known repetitions can help to validate assumptions regarding the data or underlying system. The topic of *motif discovery* investigates how similar subsequences, i.e. motifs, can be found in large data sets even when these occurrences are not exact matches. While several definitions are in use [9], we use the similarity-based definition where a motif is the subsequence that has the best possible matching subsequence elsewhere in the series, excluding itself and nearby trivial matches. Example applications of motif discovery include sound matching (such as music or bird songs), image outline classification and weather prediction. Early motif discovery techniques relied mainly on indexing the series using symbolic aggregate approximation (SAX) or other lower dimensional spaces [9]. Later works extended motif discovery in various ways, including top-k motif search [10], multidimensional data [11], parallel computation [12], and mining of consensus motifs, i.e. subsequences that are repeated in a collection of series [13]. Motif discovery is a broad research topic of which only highlights are presented here. A more detailed overview can be found in the survey paper by Torkamani and Lohweg [14].

A more recent branch of research is the *Matrix Profile*, which also allows for motif discovery. Given two series ($S1$, $S2$) and a subsequence length m, the matrix profile is a new series of length $|S1| - m + 1$ where each value is the distance of each subsequence of $S1$ to its nearest matching subsequence in $S2$. When considering only a single series ($S1 = S2$), this is referred to as a self-join, and trivial matches have to be taken into account to avoid any subsequence matching a slightly shifted version of itself. The matrix profile was originally defined with the z-normalized Euclidean distance measure as many time series from natural sources do not have a fixed scale or zero point, but later works suggested other distance measures as well [15, 16]. Motifs can be trivially located using the matrix profile; the top-k motifs are the subsequences

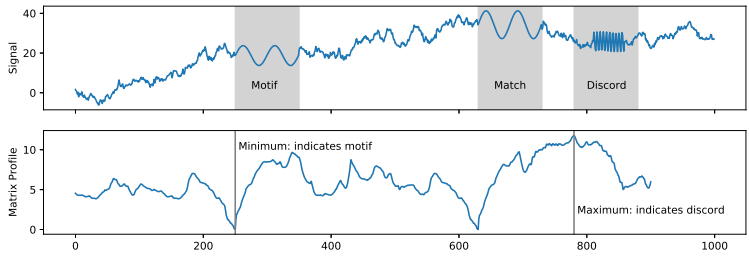


Figure 1.3: An artificial signal and corresponding (self-joined) matrix profile for a subsequence length of 100. Low values in the matrix profile indicate subsequences for which a good match can be found; i.e. motifs, high matrix profile values correspond to subsequences that are dissimilar from others, i.e. discords.

corresponding to the top-k minima in the matrix profile (taking into account neighbouring trivial matches). Besides most similar subsequences, i.e. motifs, the matrix profile also captures the most dissimilar subsequences, i.e. discords. Discords are those subsequences that differ most from any other subsequences and can be considered outliers or anomalies in the series. Figure 1.3 shows this principle for an artificial time series.

The matrix profile technique forms the basis of many state-of-the-art *descriptive analytics* techniques, further described below, as well as most work presented in this dissertation. Besides new analytics techniques, many works have introduced variants that expand on the applicability of the matrix profile as well. Examples of this include approximate calculations [17], support for multidimensional time series [18], support for missing data [19] and derived distance measures [8].

While repeated patterns are interesting, slowly evolving patterns may be too. *Time series chains* are explained as a temporally ordered set of patterns, where each pattern is similar to its preceding and following pattern, but the start and end of the chain are significantly different. Chains may indicate a slow change in the underlying system, such as machine wear down causing a deviating pattern. Figure 1.4 shows an example of chains in a gait dataset. They can be found using the matrix profile by looking for bidirectional paths in the graph that links each subsequence to its best match [20]. A later work uses a unidirectional path with angular restrictions instead [21].

Another application of pattern matching involves the *classification* of time series, where series are divided into distinct classes or categories. Going back to the running example, one can imagine that a traffic jam manifests itself as a particular pattern in the measured driving speed of a car. That means we could automatically classify our recorded car rides as a normal or jammed ride by looking for the proper pattern. The shapelet classification technique [22] is based upon this principle and consists of two phases. First, discriminative patterns are mined in a collection of labeled time se-

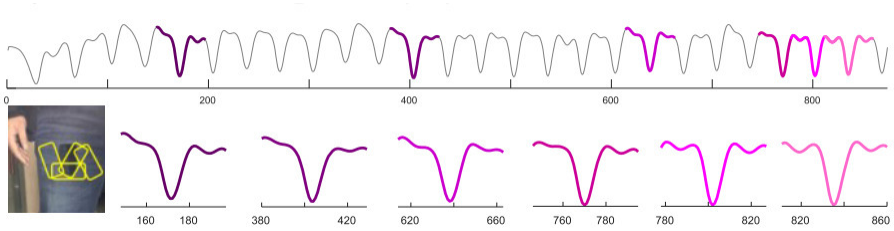


Figure 1.4: Application of time series chain discovery on a gait dataset, recorded by a mobile phone. The chain consists of a series of slowly evolving patterns, highlighted in the top. The chain can be explained by the phone obtaining a stable pocket position as the subjects walks. This figure originates from the work by Zhu et al. [20]

ries. Next, a decision tree is created, where at each step a specific pattern is compared against all subsequences in the series. Later works stepped away from a decision tree and instead suggested to create a feature matrix consisting of the distances to a set of meaningful patterns, i.e. a shapelet transform [23]. This generalization allowed the use of any classic classification method and provided significant better results. Similarly, many new methods have been introduced to mine the shapelet patterns based on gradient descent [24], evolutionary algorithms [25] or generalized eigenvectors [26]. At first sight, shapelets are a form of predictive analytics. However, they can also be seen as a form of descriptive analytics, by recognizing that a shapelet classifier discriminates classes using patterns that are interpretable by humans. In other words, the resulting patterns may prove insightful and demonstrate previously unknown differences between the different classes.

Time series segmentation is an analysis technique that divides a series into homogeneous regions, as shown in Figure 1.5. Consider for example a person wearing an activity sensor at a gym, where different types of exercises are performed on different devices. Segmentation techniques could be used to automatically detect the transitions between the different exercises. FLOSS (fast low-cost online semantic segmentation) [27] is a segmentation technique based on the matrix profile. It assumes that homogeneous segments will show repeated behavior, and splits a series where the number of repetitions differs most from the expected number.

Visualizations are perhaps one of the most common forms of descriptive analytics. However, many traditional visualization techniques are not directly applicable to time series. Multidimensional scaling (MDS) is a popular technique for visualising multidimensional data in a two-dimensional graph. MDS can be applied to subsequences of a series, but randomly selected subsequences will often result in a generic disc-like plot because these sequences lack structure and distances carry no meaning, as shown in Figure 1.6 [28]. If instead, salient subsequences are extracted based on their pairwise similarity (using the matrix profile), the resulting MDS plot does provide relevant insights.

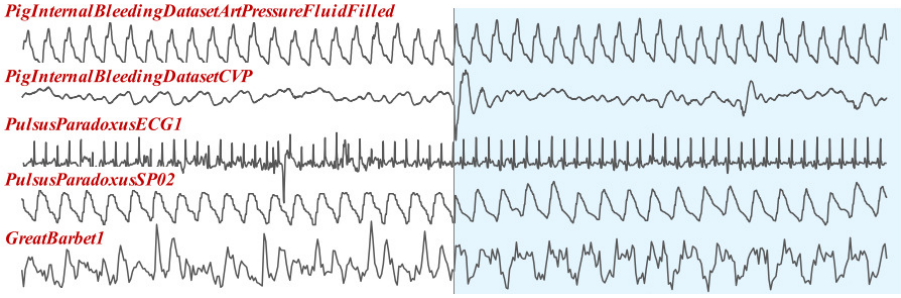


Figure 1.5: Examples of time series where a sudden behavioral change occurs, resulting in two homogeneous parts. Each example is a snippet from a larger dataset and is centered on the change point. Time series segmentation techniques can automatically find this transition. This figure is adapted from the work of Gharghabi et al. [27]

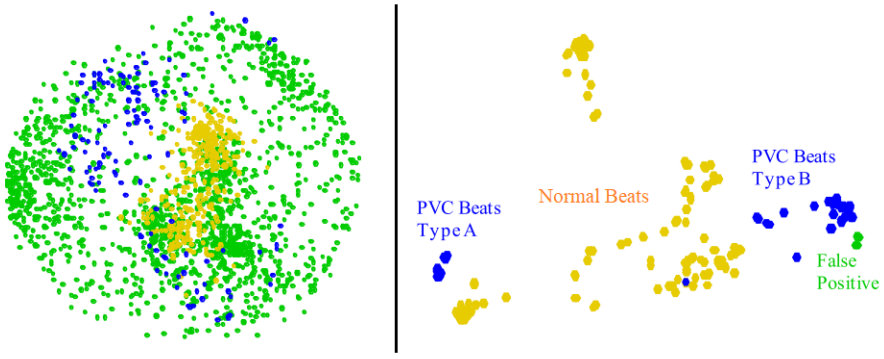


Figure 1.6: Multidimensional scaling used to visualize a dataset containing hearth beats, each point represents an extracted subsequence and is colored according to available expert annotations. The lefts visualizes randomly selected subsequences and captures no insightful structure. The right visualizes salient subsequences selected using a matrix profile based technique. This figure is adapted from the work of Yeh et al. [28]

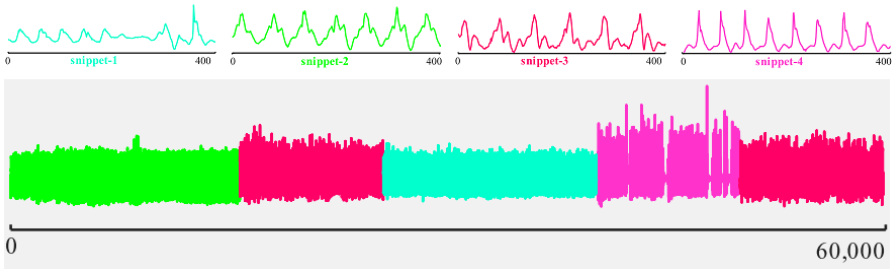


Figure 1.7: Summarizing visualisation of the PAMAP dataset, which contains accelerometer data of subjects performing various activity. The four mined snippets correspond to a specific activity and can be used to represent the dataset. This figure is adapted from the work of Imani et al. [29]

Summarizations are another type of visualisation, where the goal is to extract a representative or distinguishing visual for the time series. In the music domain, a common thumbnailing technique is the extraction of the most repeated excerpt, such as the chorus. One work explains how an Euclidean based matrix profile can be used to find this excerpt [15]. Time series snippets have been introduced as a more domain-agnostic approach [29]. The approach works by finding a small number of representative sequences, i.e. snippets, that explain the majority of the series. These snippets are then used as visual summaries, as shown in Figure 1.7. Snippet extraction is based on the more forgiving MPDist distance measure, which treats two sequences as similar when they have many similar subsequences [8].

A typical aspect of descriptive analytics is that it is *highly iterative*. Insights may lead to more questions or the need to validate former assumptions, a process that is repeated until the data is fully understood. Due to this iterative nature, it is often useful to obtain fast but less inaccurate results to help aim the direction of further investigation. Anytime algorithms have exactly this property, they produce solutions that become more accurate as they are given more time to complete. An anytime version of the matrix profile algorithm is based on a diagonal-wise distance computation and allows representative results in a fraction of the normal calculation time [17]. Another work has suggested the use of an annotation vector to refine the matrix profile after it has already been calculated [30]. An analyst can construct the annotation vector based on properties of the original signal (e.g. the energy in the signal) or any externally available knowledge (e.g. timing when the signal was recorded). The annotation vector is used to shift values in the matrix profile, so motifs correspond to the part of the signal indicated by the analyst.

The techniques in this dissertation improve existing or allow for new types of visualization, segmentation, motif mining and event detection. A detailed overview of the demonstrated insights is provided in Section 1.5. Another valuable type of insight in scope of this dissertation is anomaly detection. When examining a dataset, it

can also be interesting to find data points that are unexpected and can be considered anomalous. Anomalous data points may indicate some kind of measurement error, an error made during the processing of the data, or it may simply indicate a previously unknown behavior of the system. While anomaly detection can be seen as yet another aspect of insight mining, the vast research attention it has obtained does validate it as an independent topic.

1.4 Anomaly Detection

Anomaly detection is a broad research domain with a wide range of applications and concerns itself with finding things that are, simply put, *different*. In other words, anomaly detection techniques seek to identify data instances that do not fit what is defined as normal behavior, or more formally, instances that do not fit the representative data distribution. Besides being a research area, anomaly detection also plays a major role in industry, as demonstrated by services like Amazon CloudWatch or Microsoft Azure Anomaly Detector. As such, anomaly detection has a wide range of applications domains such as quality control in production settings or machine monitoring for predictive maintenance. Anomaly detection also plays an important role in everyday life, though not always apparent, with applications such as spam detection or surveillance. Because of this wide range of applications, anomaly detection has become a major research domain with a rich history. Still, as collected data continue to grow in both scale and complexity, anomaly patterns become more varied and challenging to detect, fueling the need for ever more advanced techniques that focus on more detailed scenarios.

Anomaly detection methods can be categorised in a number of ways, one interesting categorisation relates to the data requirements for each technique. Supervised *fault detection* techniques rely on labeled data, meaning that it is known whether each example is considered anomalous or normal. They are commonly used when looking for specific types of anomalies that are common or have been well documented, such as phishing mails in spam detection or typical maintenance issues in industrial machines. Example techniques include classifiers, which assign input values to one of multiple classes (normal and faults), and expectation-based regression models. However, fault detection approaches typically suffer from a lack of available labeled data or a high imbalance between normal and abnormal examples.

Unsupervised *outlier detection* techniques do not require labeled data, instead they rely on self-structuring mechanisms and similarities between data points. Outliers may be the result of measurement errors but may also signify previously unknown behavior. One common assumption here is that outliers occur rarely, so anomalies can be found using density based method, where they occur as isolated points, or as distinct instances using similarity-based strategies. Outlier detection cannot distinguish different types of anomalies like fault detection, but is more widely applicable to use cases because labels are not needed.

Finally, semi-supervised *novelty detection* methods try to combine the best of both worlds and are typically used when one dataset defines baseline behavior and new data needs to be scanned for anomalies. Semi-supervised techniques utilize data sets where a limited set of labels is available, often it suffices to have examples of normal data. Partition-based classifiers like isolation forests or property-based methods like auto-encoders succeed in creating a model that can test for normality in unseen data, which can be used to detect anomalies.

Finally, the term *noise detection* is also used in literature when finding and removing abnormal values in a dataset before training a machine learning model. Noise detection is similar to outlier detection, but treats outliers as undesired rather than interesting. Since the presence of anomalous values can degrade the performance of machine learning models, detecting and removing noise is a common practice when anomaly detection is not the goal.

Anomaly detection is typically accompanied with a number of practical challenges, irrespective of the application domain being considered:

1. **Rarity**: Anomalies are, by their very definition, rare in nature. In general, this results in highly unbalanced datasets that limits the number of useful techniques to find them. Evaluation also becomes more challenging, since evaluating a small number of examples is statistically less reliable.

2. **Lack of labels**: Supervised techniques require examples of anomalies in order to detect similar cases. However, labeled data is often not available and labeling data is often costly for more complex cases. Semi-supervised techniques relax this requirement and only depend on data containing normal behavior. Still, the collection of this type of data can remain impossible or non-straightforward for specific domains, such as those where anomalies can go unnoticed. The lack of labels also affect evaluation.

3. **Dynamicity**: Anomalies can become more dynamic as the data becomes more complex. This means that collecting an exhaustive list of all possible anomalies is often impossible.

4. **Subjectivity**: Defining what constitutes an anomaly is a highly subjective question that depends on the background and expectations of the observer. It is closely related to the subtle distinction between outliers (rare events), faults (undesired behavior) and events (known behavior). In time series, there can also be disagreement as to when exactly the anomaly occurs, which has implications in labeling and evaluation.

5. **Context**: The surrounding context of a process can heavily influence the resulting measurements. What is considered normal under one specific context may not be under a different context. This includes a wide range of factors that

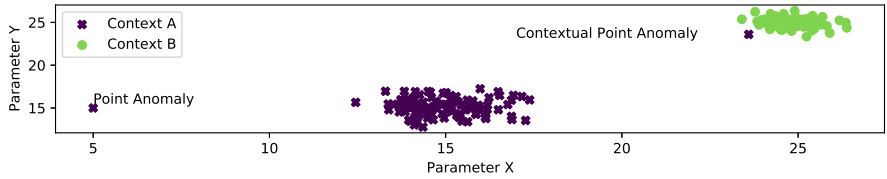


Figure 1.8: Example of a non-time series dataset containing a point anomaly and contextual point anomaly. Where the point anomaly is a clear outlier, the contextual anomaly can only be spotted by considering the context information (color/symbol) into account.

may or may not be (directly) captured in the data. For example, computer network security applications may see significant changes between weekdays and weekends.

1.4.1 Types of Anomalies

Anomalies can be categorized into different types, where some techniques may only be suited to detect one specific type. These types are strongly related to the shape of the feature matrix, as discussed in Section 1.2.

1.4.1.1 Traditional Data

If we consider traditional, non-time series data, two types of anomalies are considered: point anomalies and contextual point anomalies, an example of each is visualized in Figure 1.8 and Table 1.1. *Point anomalies* are observations that fall outside the data distribution of the majority of the data and can be recognized by one (uni-variate) or a combination of multiple (multi-variate) feature values that are outliers. They are often easy to spot using visualizations in low-dimensional data sets or simple statistical methods such as histograms.

Contextual anomalies are observations whose feature values fit nicely in the global data distribution, but are anomalous when also considering their context information. While Figure 1.8 uses two discrete context types, contexts can be very complex in practice. Examples include the effect of the time of the year when observing outdoor temperatures, where context is cyclical and non-discrete, or the different rooms in a hospital where equipment is monitored, where context can only be expressed as the similarity between different setups.

A schematic representation of these anomaly types can be seen in Figure 1.9. Note that all anomalous values span across the features axis. It would also be possible that anomalies span across the observation axis, consider for example the situation where at one point during the recording process, the measuring tool for one specific feature begins to malfunction. However, a common assumption is that all observations are

Context	Param X	Param Y	
A	**5.0**	15.0	Point anomaly
A	14.1	16.3	
A	15.5	14.7	
A	23.6	23.6	Contextual Point Anomaly
B	23.8	25.4	
B	25.6	23.8	

Table 1.1: Feature matrix of an extract of the data visualized in Figure 1.8. The point anomaly can be easily spotted by the X value that falls outside regular range. However, the contextual point anomaly can only be spotted if we compare observations against observations of the same context.

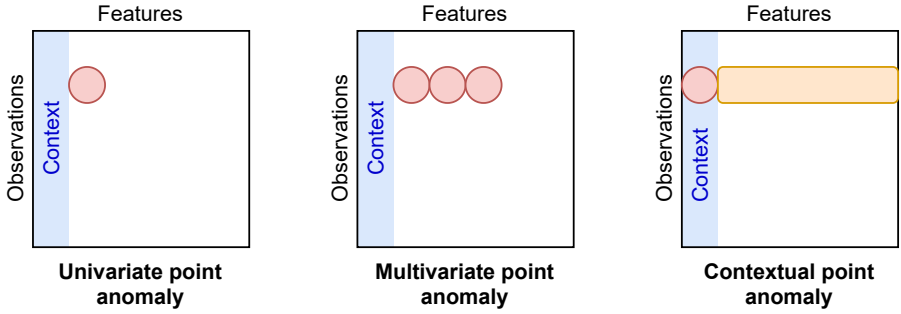


Figure 1.9: Schematic representation of the feature matrix with different anomaly types in traditional non-time series data.

considered independent from each other, meaning that this situation would simply be considered as multiple point anomalies. We can use this representation to compare the differences with anomalies in time series data.

1.4.1.2 Time Series Data

As mentioned before, working with time series implies that observations can be considered as measurements over a certain range (spatial or temporal). This is translated as a third dimension in the feature matrix, and means that we can differentiate additional types of anomalies. This is shown schematically in Figure 1.10, or in Figure 1.11 for a collection of fictional time series.

The first type is the *point anomaly*, which is very similar to the traditional type. In the context of a time series, this is a single value in a series that is anomalous. Point anomalies can typically be found using the same methods as for non-time series. Multivariate point and contextual point anomalous are also possible, but are less com-

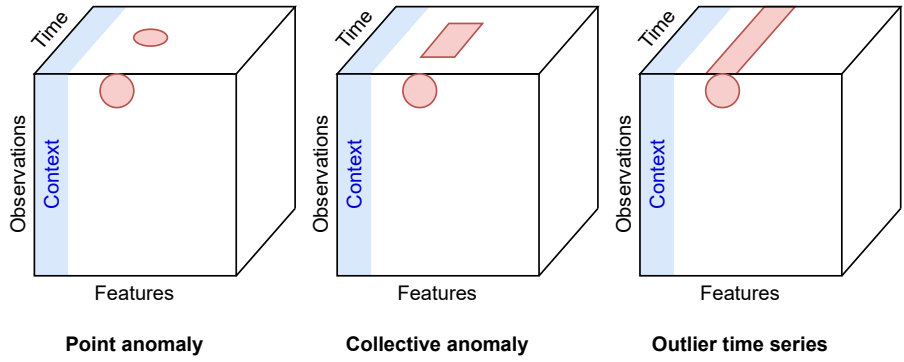


Figure 1.10: Schematic representation of the feature matrix with different anomaly types in time series data. For brevity, we omit multivariate and contextual variants.

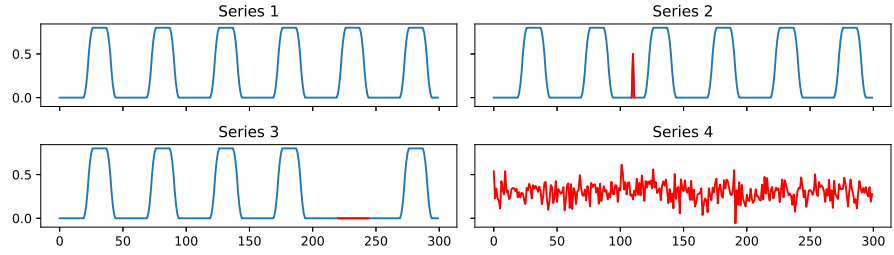


Figure 1.11: Four time series demonstrating different anomaly types. Series 1 shows normal behavior. Series 2 contains a point anomaly, series 3 contains a collective anomaly and series 4 is an outlier time series with respect to the other 3 series.

monly addressed in literature [31]. The reason for this might be that time series often originate from automated and continuous high-resolution measuring processes, so the chance of a single outlier point is very small.

On the other side of the spectrum, we have the *outlier time series*, where an entire series is considered anomalous. This type of anomaly detection can be useful when one has a collection of time series that describe a common system but may contain series that do not belong in that collection. In this case, an outlier time series may be due to a mislabeling or a failed sensor. Again, multivariate and contextual group anomalies are possible, though research on this type of anomalies is still relatively limited [31]. Two possible explanations could be the rarity of this type of error, or the fact that more simple approaches suffice to detect this type of data problem.

Finally, *collective anomalies* (also known as subsequence anomalies or group anomalies) represent consecutive values in a series that together form an anomaly, even

though individual values may valid. As an example, consider a seasonal signal where one repetition is left out. Collective anomaly detection, including multivariate and contextual variants, is especially relevant for industrial applications, where services or machines are continuously monitored. Note that some methods to detect collective anomalies may also be useful to detect point anomalies. Consider for example the second series in Figure 1.11. While this series contains only a single anomalous value, it also forms a distinct pattern with neighbouring values.

1.4.2 Anomaly Detection Strategies

Alternatively, it is interesting to categorize anomaly detection algorithms by their strategy. Four major strategies can be considered [32]: rule-based, case-based, expectation-based and property-based.

1.4.2.1 Rule-based Methods

Rule-based methods qualify data by executing explicit or implicit rules. Domain experts often have a good feel about the behavior of the considered system and what can be qualified as normal behavior. This knowledge can be converted to explicit rules, either through direct implementation in the detection system, or indirectly through a rule-based reasoning system such as Drools[2] or semantic reasoning. Alternatively, rules can be derived from data using supervised classification techniques such as decision trees or association rule mining. Rules have the potential to be interpretable, which is a desired trait in many domains. However, rule-based methods will often be incomplete, as not every scenario is readily available to domain experts or present in data. As a collection of rules grows, management becomes a burden since rules may overlap or conflict. Finally, rules may be less suited for dynamic domains, where the definition of normal gradually changes, due to their rigid nature.

1.4.2.2 Case-based Methods

Where rule-based methods gather or find rules to differentiate abnormal from normal behavior, case-based methods look for example reference cases to make this decision. New data is classified based on the similarity with these reference cases. Case-based systems can become more accurate over time as more different cases are ingested and annotated by a user, provided that feedback is accurate and relevant. The selected similarity measure has a big impact on the performance. It can be made to accommodate a wide array of complex data types, though tweaking the distance measure for optimal results is not straightforward and may prove challenging. One other downside of this method is computational complexity, as there is no single streamlined model to classify the data. Example techniques include nearest neighbor techniques, signature-based techniques and pattern-based techniques like the matrix profile.

[2] `https://www.drools.org/`

1.4.2.3 Expectation-based Methods

Models using an expectation-based strategy look for anomalies by modeling the monitored output under normal conditions. For example, any regression method can be used to predict a specific feature that is present in the data. When the difference between predicted and observed values is too large, an anomaly is reported. Probabilistic techniques such as Gaussian mixture models, that determine the distribution of valid values are also straightforward to use, and might be more applicable for anomalies concerning correlated output variables. By estimating the distribution of specific features, it is straightforward to determine values with low probability as anomalies. Other techniques, like Gaussian processes, combine regression with uncertainty ranges. Overall, this strategy relies on the ability to fit a proper model over the data. The abundance of regression techniques, ranging from simple linear models to complex long short-term memory (LSTM) neural networks, shows that finding a proper technique is not always straightforward. Furthermore, selecting a proper anomaly threshold is application dependent, and requires either expert knowledge or enough anomalous examples to make an informed decision.

1.4.2.4 Property-based Methods

Property-based methods are based on the assumption that normal data exhibits certain latent properties. Unsupervised property-based methods are trained on data that is known to contain no anomalies and form a model of normality. New data is evaluated by seeing how well it fits withing the learned model. The best known methods under this category are reconstruction-based methods such as principal component analysis or auto-encoders. These dimensionality reduction techniques effectively create a way to compress data, while also being able to recreate the original data from this compressed form with a minimal error. Because the compression was learned using normal data, it will work as expected for new normal data, but will result in a high reconstruction error for anomalous data. Supervised property-based methods, like (deep) neural networks, learn to distinguish discriminative patterns that can be used for classifying different types of anomaly. The challenges for property-based methods remains in finding the best suited properties and determining the specific criteria for flagging an anomaly.

1.5 Chapter Overview

This chapter provided a high level overview of different types of data analytics, anomalies and anomaly detection approaches. This background is used to better frame the challenges tackled in the following chapters, as discussed next. Figure 1.12 summarizes the chapter topics and highlights how the chapters are related to one another.

Chapter 2 proposes an innovative *expectation-based* model to detect anomalous welds during a specific stage in the steel production process. Many of the common

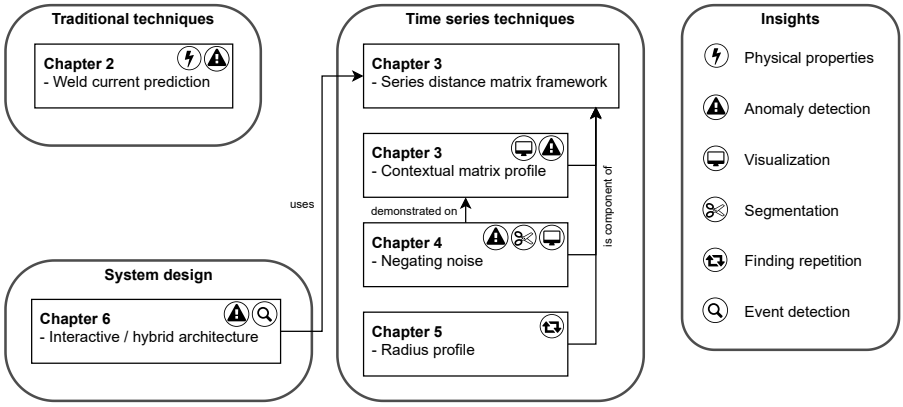


Figure 1.12: A schematic overview showing the link between the different chapters.

challenges present in industrial settings apply here, including a *lack of labels* and *changes to the underlying system*, which we can observe through *descriptive analytics*. Existing methods are based on a collection of statistical models that are slow to react to these changes. The proposed method captures the dynamics of the system in a single interpretable model that adapts faster, resulting in less false positive detected anomalies, while still using the same statistical features.

Starting from Chapter 3, the focus shifts to the Swiss army knife of time series data analytics, i.e. the matrix profile. Many of the analytics that were mentioned can be based on or derived from this technique, making this technique a vital tool for any analyst. However, analysts are limited in their ability to apply this technique or its many possible variations described in literature, this is because different techniques are implemented independently and are therefor hard to combine or customize. This chapter introduces the series distance matrix framework as a way to freely and efficiently combine various distance measures with the various matrix profile related techniques. This chapter also introduces the contextual matrix profile as an example extension based on this framework. The contextual matrix profile is a new technique that can be used for *descriptive analytics* such as *visualizations* between two series and can be used to find *contextual* temporal anomalies.

The flexibility that the series distance matrix framework brings, open many options for data analytics by freely combining distance measures and techniques. Chapter 4 focuses on the most often used distance measure in matrix profile related works, i.e. the z-normalized Euclidean distance. This measure can be used to compare the shape of subsequences, irrespective of scaling or translation, which makes it useful in applications with natural signals. However, this measure is quite unintuitive for noisy signals lacking distinctive shapes where the noise effectively determines the shape of the subsequences. Because these types of signals are common in industrial environments, a method is suggested to negate the detrimental effects of noise. This improves the

applicability and utility of this distance measure, as demonstrated on an *case-based* anomaly detection case and two data *descriptive analytics* cases: *segmentation* and *visualization*.

Most data analytics techniques presented so far relate to one or two time series. Consensus motif mining is one technique that finds repeating subsequences within a larger collection of time series. Chapter 5 introduces the concept of the radius profile, a derived series that can be used to easily extract consensus motifs, similar to how the matrix profile can be used to extract motifs. In further likeness to the matrix profile which has been used for many analytical techniques, the radius profile may provide a foundation for further multi-series techniques. Furthermore, this chapter also introduces a variation of the radius profile, to be used for finding patterns that are well preserved across several repetitions irregardless of where they occur, and which can be used on one or more time series.

Descriptive analytics are most useful when exploring a new problem domain. In contrast, the situation in industry domains is often more mature as expert knowledge is more readily available. While expert knowledge is vital, it is seldom complete, leaving a gap that can be filled by data driven techniques. Chapter 6 discusses a system architecture where *rule-based* techniques based on expert knowledge and *case-based* anomaly and event detection are applied side-by-side. As data is ingested, events are automatically detected and visualized by a dynamic dashboard that assists the user in finding proper *visualizations*. As the user interacts with the dashboard, feedback is looped back to the detectors to further improve utility.

Finally, Chapter 7 concludes the work of this dissertation and suggests possibilities regarding future work.

1.6 Publications

The following list provides an overview of publications made during my PhD.

1.6.1 Publications in international journals

1. Carina Veeckman, Karel Jedlička, **Dieter De Paepe**, Dmitrii Kozhukh, Štěpán Kafka, Pieter Colpaert, and Otakar Čerba. "Geodata interoperability and harmonization in transport: a case study of open transport net". In: *Open Geospatial Data, Software and Standards* 2.1 (2017), p. 3. ISSN: 2363-7501. DOI: 10.1186/s40965-017-0015-6. URL: http://dx.doi.org/10.1186/s40965-017-0015-6

2. Raf Buyle, Ziggy Vanlishout, Serena Coetzee, **Dieter De Paepe**, Mathias Van Compernolle, Geert Thijs, Bert Van Nuffelen, Laurens De Vocht, Peter Mechant, Bjorn De Vidts, and Erik Mannens. "Raising interoperability among base registries: The evolution of the Linked Base Registry for addresses in Flanders". In: *Journal of Web Semantics* 55 (2019), pp. 86–101. ISSN: 1570-8268. DOI:

https://doi.org/10.1016/j.websem.2018.10.003. URL: http://www.sciencedirect.com/science/article/pii/S1570826818300519

3. Sander Vanden Hautte, Pieter Moens, Joachim Van Herwegen, **Dieter De Paepe**, Bram Steenwinckel, Stijn Verstichel, Femke Ongenae, and Sofie Van Hoecke. "A Dynamic Dashboarding Application for Fleet Monitoring Using Semantic Web of Things Technologies". In: *Sensors* 20.4 (2020). ISSN: 1424-8220. DOI: 10.3390/s20041152. URL: https://www.mdpi.com/1424-8220/20/4/1152

4. **Dieter De Paepe**, Sander Vanden Hautte, Bram Steenwinckel, Filip De Turck, Femke Ongenae, Olivier Janssens, and Sofie Van Hoecke. "A generalized matrix profile framework with support for contextual series analysis". In: *Engineering Applications of Artificial Intelligence* 90 (2020), p. 103487. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2020.103487. URL: http://www.sciencedirect.com/science/article/pii/S0952197620300087

5. Bram Steenwinckel, **Dieter De Paepe**, Sander Vanden Hautte, Pieter Heyvaert, Mohamed Bentefrit, Pieter Moens, Anastasia Dimou, Bruno Van Den Bossche, Filip De Turck, Sofie Van Hoecke, and Femke Ongenae. "FLAGS: A methodology for adaptive anomaly detection and root cause analysis on sensor data streams by fusing expert knowledge with machine learning". In: *Future Generation Computer Systems* 116 (2021), pp. 30–48. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2020.10.015. URL: http://www.sciencedirect.com/science/article/pii/S0167739X20329927

6. **Dieter De Paepe**, Sander Vanden Hautte, Bram Steenwinckel, Pieter Moens, Jasper Vaneessen, Steven Vandekerckhove, Bruno Volckaert, Femke Ongenae, and Sofie Van Hoecke. "A Complete Software Stack for IoT Time Series Analysis that Combines Semantics and Machine Learning – Lessons Learned from the Dyversify Project". In: *Journal of Systems and Software* (Submitted Jan 2021)

7. **Dieter De Paepe**, Andy Van Yperen-De Deyne, Jan Defever, and Sofie Van Hoecke. "An Incremental Physics-Inspired Current Regression Model for Anomaly Detection of Resistance Mash Seam Welding in Steel Mills". In: *Computers in industry* (Submitted Apr 2021)

1.6.2 Publications in international conference proceedings

1. **Dieter De Paepe**, Tobias Maschler, Fridolin Wild, Erico Ferro, Jesse Marsh, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. "Textile and Clothing Business Labs". In: *Proceedings of the 13th Extended Semantic Web Conference: Project Networking*. June 2016. URL: http://2016.eswc-conferences.org/sites/default/files/papers/EU-Project-

Networking - Session / Textile %20and %20Clothing %20Business %20Labs.pdf

2. **Dieter De Paepe**, Ruben Verborgh, and Erik Mannens. "Rule-Based Reasoning using State Space Search". In: *Proceedings of the 15th International Semantic Web Conference: Posters and Demos*. Ed. by Takahiro Kawamura and Heiko Paulheim. Vol. 1690. CEUR Workshop Proceedings. Oct. 2016. URL: http://ceur-ws.org/Vol-1690/paper55.pdf

3. Raf Buyle, Laurens De Vocht, Mathias Van Compernolle, **Dieter De Paepe**, Ruben Verborgh, Ziggy Vanlishout, Björn De Vidts, Peter Mechant, and Erik Mannens. "OSLO: Open Standards for Linked Organizations". In: *Proceedings of the International Conference on Electronic Governance and Open Society: Challenges in Eurasia*. Nov. 2016, pp. 126–134. ISBN: 978-1-4503-4859-1. DOI: 10.1145/3014087.3014096. URL: https://dl.acm.org/authorize?N20629

4. Raf Buyle, Laurens De Vocht, **Dieter De Paepe**, Mathias Van Compernolle, Geraldine Nolf, Ziggy Vanlishout, Björn De Vidts, Erik Mannens, and Peter Mechant. "The Public Sector DNA on the web: semantically marking up government portals". In: *Proceedings of the Workshop on Smart Descriptions & Smarter Vocabularies*. Nov. 2016. URL: https://www.w3.org/2016/11/sdsvoc/SDSVoc16_paper_1

5. **Dieter De Paepe**, Geert Thijs, Ruben Verborgh, Erik Mannens, and Raf Buyle. "Automated UML-Based Ontology Generation in OSLO²". In: *Proceedings of the 14th ESWC: Posters and Demos*. Ed. by Eva Blomqvist, Katja Hose, Heiko Paulheim, Agnieszka Ławrynowicz, Fabio Ciravegna, and Olaf Hartig. Vol. 10577. Lecture Notes in Computer Science. Springer, May 2017, pp. 93–97. ISBN: 978-3-319-70407-4. DOI: 10.1007/978-3-319-70407-4_18. URL: https://doi.org/10.1007/978-3-319-70407-4_18

6. Raf Buyle, Mathias Van Compernolle, **Dieter De Paepe**, Jens Scheerlinck, Peter Mechant, Erik Mannens, and Ziggy Vanlishout. "Semantics in the wild : a digital assistant for Flemish citizens". eng. In: *Proceedings of the 11th International Conference on Theory and Practice of Electronic Governance*. Galway, Iceland, 2018, pp. 1–6. ISBN: 9781450354219. URL: http://dx.doi.org/10.1145/3209415.3209421

7. Bram Steenwinckel, Pieter Heyvaert, **Dieter De Paepe**, Olivier Janssens, Sander Vanden Hautte, Anastasia Dimou, Filip De Turck, Sofie Van Hoecke, and Femke Ongenae. "Automated extraction of rules and knowledge from risk analyses : a ventilation unit demo". eng. In: *Proceedings of the ISWC 2018 posters & demonstrations, industry and blue sky ideas tracks, co-located with 17th international Semantic Web conference (ISWC 2018)*. Ed. by Marieke van Erp, Medha

Atre, Vanessa Lopez, Kavitha Srinivas, and Carolina Fortuna. Vol. 2180. Monterey, CA, USA, 2018, p. 4. URL: http://ceur-ws.org/Vol-2180/paper-63.pdf

8. Bram Steenwinckel, Pieter Heyvaert, **Dieter De Paepe**, Olivier Janssens, Sander Vanden Hautte, Anastasia Dimou, Filip De Turck, Sofie Van Hoecke, and Femke Ongenae. "Towards adaptive anomaly detection and root cause analysis by automated extraction of knowledge from risk analyses". eng. In: *Proceedings of the 9th international semantic sensor networks workshop, co-located with 17th international semantic web conference (ISWC 2018)*. Vol. 2213. Monterey, CA, USA, 2018, pp. 17–31. URL: http://ceur-ws.org/Vol-2213/paper2.pdf

9. **Dieter De Paepe**, Olivier Janssens, and Sofie Van Hoecke. "Eliminating Noise in the Matrix Profile". In: *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM,*. INSTICC. SciTePress, Feb. 2019, pp. 83–93. ISBN: 978-989-758-351-3. DOI: 10.5220/0007314100830093

10. **Dieter De Paepe**, Diego Nieves Avendano, and Sofie Van Hoecke. "Implications of Z-Normalization in the Matrix Profile". In: *Pattern Recognition Applications and Methods*. Ed. by Maria De Marsico, Gabriella Sanniti di Baja, and Ana Fred. Cham: Springer International Publishing, 2020, pp. 95–118. ISBN: 978-3-030-40014-9. DOI: https://doi.org/10.1007/978-3-030-40014-9_5

11. **Dieter De Paepe** and Sofie Van Hoecke. "Mining Recurring Patterns in Real-Valued Time Series using the Radius Profile". In: *Proceedings of the 20th International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 984–989. DOI: 10.1109/ICDM50108.2020.00113

References

[1] Amir Yazdanbakhsh, Christof Angermueller, Berkin Akin, Yanqi Zhou, Albin Jones, Milad Hashemi, Kevin Swersky, Satrajit Chatterjee, Ravi Narayanaswami, and James Laudon. *Apollo: Transferable Architecture Exploration*. 2021. arXiv: 2102.01723 [cs.LG].

[2] Thomas H Davenport and DJ Patil. "Data scientist". In: *Harvard business review* 90.5 (2012), pp. 70–76.

[3] Sakinah S.J. Alhadad. "Visualizing Data to Support Judgement, Inference, and Decision Making in Learning Analytics: Insights from Cognitive Psychology and Visualization Science". In: *Journal of Learning Analytics* 5.2 (Aug. 2018), pp. 60–85. DOI: 10.18608/jla.2018.52.5. URL: https://learning-analytics.info/index.php/JLA/article/view/5815.

[4] Itai Yanai and Martin Lercher. "A hypothesis is a liability". In: *Genome Biology* 21.1 (Sept. 2020), p. 231. ISSN: 1474-760X. DOI: 10.1186/s13059-020-02133-w.

[5] Jakob Runge et al. "Inferring causation from time series in Earth system sciences". In: *Nature Communications* 10.1 (June 2019), p. 2553. ISSN: 2041-1723. DOI: 10.1038/s41467-019-10105-3. URL: https://doi.org/10.1038/s41467-019-10105-3.

[6] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. "Classification of time series by shapelet transformation". In: *Data mining and knowledge discovery* 28.4 (2014), pp. 851–881.

[7] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. "Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package)". In: *Neurocomputing* 307 (2018), pp. 72–77. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2018.03.067. URL: https://www.sciencedirect.com/science/article/pii/S0925231218304843.

[8] Shaghayegh Gharghabi, Shima Imani, Anthony Bagnall, Amirali Darvishzadeh, and Eamonn Keogh. "MPdist: A Novel Time Series Distance Measure to Allow Data Mining in More Challenging Scenarios". In: *IEEE Int. Conf. on Data Mining (ICDM)*. 2018, pp. 965–970. ISBN: 978-1-5386-9159-5.

[9] Abdullah Mueen. "Time series motif discovery: dimensions and applications". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 4.2 (2014), pp. 152–159.

[10] Hoang Thanh Lam, Ninh Dang Pham, and Toon Calders. "Online discovery of top-k similar motifs in time series data". In: *Proceedings of the 2011 SIAM International Conference on Data Mining*. SIAM. 2011, pp. 1004–1015.

[11] Amy McGovern, Derek H Rosendahl, Rodger A Brown, and Kelvin K Droegemeier. "Identifying predictive multi-dimensional time series motifs: an application to severe weather prediction". In: *Data Mining and Knowledge Discovery* 22.1 (2011), pp. 232–258.

[12] Ankur Narang and Souvik Bhattacherjee. "Parallel exact time series motif discovery". In: *European Conference on Parallel Processing*. Springer. 2010, pp. 304–315.

[13] Kaveh Kamgar, Shaghayegh Gharghabi, and Eamonn Keogh. "Matrix Profile XV: Exploiting Time Series Consensus Motifs to Find Structure in Time Series Sets". In: *International Conference on Data Mining (ICDM)*. IEEE. 2019, pp. 1156–1161.

[14] Sahar Torkamani and Volker Lohweg. "Survey on time series motif discovery". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 7.2 (2017), e1199.

[15] Diego F Silva, Chin-chia M Yeh, Yan Zhu, Gustavo E A P A Batista, and Eamonn Keogh. "Fast Similarity Matrix Profile for Music Analysis and Exploration". In: *IEEE Transactions on Multimedia* 21.1 (Jan. 2019), pp. 29–38. ISSN: 1520-9210. DOI: 10.1109/TMM.2018.2849563. URL: https://ieeexplore.ieee.org/document/8392419/.

[16] Reza Akbarinia and Bertrand Cloez. "Efficient matrix profile computation using different distance functions". In: *arXiv preprint arXiv:1901.05708* (2019).

[17] Yan Zhu, Chin-Chia Michael Yeh, Zachary Zimmerman, Kaveh Kamgar, and Eamonn Keogh. "SCRIMP++: Time Series Motif Discovery at Interactive Speeds". In: *IEEE Int. Conf. on Data Mining (ICDM)*. 2018, pp. 837–846. ISBN: 978-1-5386-9159-5.

[18] Chin-Chia Michael Yeh, Nickolas Kavantzas, and Eamonn Keogh. "Meaningful Multidimensional Motif Discovery". In: *IEEE Int. Conf. on Data Mining (ICDM)*. IEEE, 2017, pp. 565–574. ISBN: 978-1-5386-3835-4.

[19] Yan Zhu, Abdullah Mueen, and Eamonn Keogh. "Admissible Time Series Motif Discovery with Missing Data". In: (2018). arXiv: 1802.05472. URL: https://arxiv.org/pdf/1802.05472.pdf.

[20] Yan Zhu, Makoto Imamura, Daniel Nikovski, and Eamonn Keogh. "Time Series Chains: A New Primitive for Time Series Data Mining". In: *IEEE Int. Conf. on Data Mining (ICDM)*. 2017, pp. 695–704. ISBN: 978-1-5386-3835-4.

[21] Makoto Imamura, Takaaki Nakamura, and Eamonn Keogh. "Matrix Profile XXI: A Geometric Approach to Time Series Chains Improves Robustness". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. New York, NY, USA: ACM, Aug. 2020, pp. 1114–1122. ISBN: 9781450379984. DOI: 10.1145/3394486.3403164. URL: https://dl.acm.org/doi/10.1145/3394486.3403164.

[22] Lexiang Ye and Eamonn Keogh. "Time Series Shapelets: A New Primitive for Data Mining". In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '09. Paris, France: Association for Computing Machinery, 2009, pp. 947–956. ISBN: 9781605584959. DOI: 10.1145/1557019.1557122. URL: https://doi.org/10.1145/1557019.1557122.

[23] Jason Lines, Luke M. Davis, Jon Hills, and Anthony Bagnall. "A Shapelet Transform for Time Series Classification". In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '12. Beijing, China: Association for Computing Machinery, 2012, pp. 289–297. ISBN: 9781450314626. DOI: 10.1145/2339530.2339579. URL: https://doi.org/10.1145/2339530.2339579.

[24] Josif Grabocka, Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. "Learning time-series shapelets". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 392–401.

[25] Gilles Vandewiele, Femke Ongenae, and Filip De Turck. "GENDIS: Genetic Discovery of Shapelets". In: *Sensors* 21.4 (2021), p. 1059.

[26] Lu Hou, James Kwok, and Jacek Zurada. "Efficient learning of timeseries shapelets". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 1. 2016.

[27] Shaghayegh Gharghabi, Yifei Ding, Chin-Chia Michael Yeh, Kaveh Kamgar, Liudmila Ulanova, and Eamonn Keogh. "Matrix Profile VIII: Domain Agnostic Online Semantic Segmentation at Superhuman Performance Levels". In: *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, Nov. 2017, pp. 117–126. ISBN: 978-1-5386-3835-4.

[28] Chin-Chia Michael Yeh, Helga Van Herle, and Eamonn Keogh. "The matrix profile allows visualization of salient subsequences in massive time series". In: *Proc. IEEE Int. Conf. on Data Mining, ICDM* (2017), pp. 579–588. ISSN: 2374-8486.

[29] Shima Imani, Frank Madrid, Wei Ding, Scott Crouter, and Eamonn Keogh. "Time Series Snippets: A New Primitive for Time Series Data Mining". In: *IEEE Int. Conf. on Big Knowledge (ICBK)*. IEEE, 2018, pp. 382–389. ISBN: 978-1-5386-9125-0.

[30] Hoang Anh Dau and Eamonn Keogh. "Matrix Profile V: A Generic Technique to Incorporate Domain Knowledge into Motif Discovery". In: *Proc. 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017, pp. 125–134. ISBN: 9781450348874.

[31] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A Lozano. "A review on outlier/anomaly detection in time series data". In: *arXiv preprint arXiv:2002.04236* (2020).

[32] Chengqiang Huang. "Featured Anomaly Detection Methods and Applications". In: (2018).

Chapter 2

An Incremental Physics-Inspired Current Regression Model for Anomaly Detection of Resistance Mash Seam Welding in Steel Mills

While Chapter 1 divides analytics into four different categories with different focuses, some topics naturally bring several of these categories together. Quality control in production industry is one such topic. Descriptive analytics are often a prerequisite, as data needs to be fully understood so a strategy for control can be setup. Next, predictive analytics can be applied to estimate quality using the available data and business rules will determine how to use this information as part of prescriptive analytics.

This chapter demonstrates this amalgamation of analytics for a quality control use case originating from the steel industry. Specifically, we focus on the welding of steel plates in the annealing line, where various time series are recorded during the welding process. In a first stage, descriptive analytics reveal some shortcomings in the data capturing methods and allows us to make an informed decision to focus on weld current prediction. Next, we create a model that accurately predicts weld current (predictive analytics). By using a gray-box model, we are even able to gain insight into the underlying process (diagnostic analytics). Finally, the model is used to detect unreliable welds, which are rewelded to minimize the risk of weld breaks (prescriptive analytics).

My contributions can be summarized as follows:

1. Proposal of a fully interpretable regression model for weld current with similar predictive power as non-insightful state of the art techniques.

2. An incremental version of this model which outperforms state of the art, because it can adapt more quickly to both minor and major changes attributed to machine maintenance and variations in steel composition.

An Incremental Physics-Inspired Current Regression Model for Anomaly Detection of Resistance Mash Seam Welding in Steel Mills

D. De Paepe, A. Van Yperen-De Deyne, J. Defever, and S. Van Hoecke

Submitted to "Computers in Industry"

Abstract Annealing and galvanization lines in steel mills run continuously to maximize production throughput. As a part of this process, individual steel coils are joined end-to-end using mash seam welding. Because weld breaks result in a production loss of multiple days, non-destructive tests are used to detect and replace poor quality welds. Here, statistical tests are most common as they use data readily available from the welding machine and require no specialised or hard to install equipment. However, these models do not provide insight into the underlying process and are slow to adapt for changes caused by machine maintenance or varying material composition. We developed a welding current prediction model that uses the same data as analytical models, but is based on known physical laws. In this work we detail our model and show evaluation results using industrial welding data collected over a period of 15 months. Our incremental model outperforms statistical models due to its rapid adaptability to changes over time. Our model is straightforward, insightful towards experts and resulted in two thirds less rejected welds on our data, which contained shifts in the welding current attributed to maintenance.

2.1 Introduction

At one point in the steel sheet production process, sheets are annealed to increase ductility. To maximise production throughput at this stage, coiled sheets are welded end-to-end to form a continuous steel sheet that is pulled through the annealing furnace at a constant rate, after which the individual sheets are again cut out and coiled. Mash seam welding (also known as narrow lap welding) is used to weld the steel coils together. Here, the ends of two coils are overlapped over a very short distance. Next, two electrified cylinders roll over this seam with high pressure. Due to the electrical resistance of the metal, the steel heats, softens and can be mashed together by the rollers. Shortly after the welding cylinders, two planning wheels follow to further flatten the weld, completing the process in no more than 30 seconds.

If a weld breaks while passing through the furnace, the sheet becomes jammed and the production line has to be stopped. To restore the line, workers have to enter the furnace after it has been cooled and vented, exposing them to a high-risk environment. As the downtime typically lasts multiple days, it also incurs major costs for the otherwise continuous production line. In an effort to avoid weld breaks, each new weld is inspected by the operator. Because manual inspection is subjective and somewhat hindered by the machine hardware, the operator is typically supported by an automatic

system that produces a quality score for each weld. If the operator finds the weld unsatisfactory, they cut out the weld and repeat the welding procedure. Because the annealing furnace has a constant throughput, the time for (re-)welding and inspection is limited and only one or two re-welds can be made without stopping the production line.

Many factors influence weld quality. Most critical are the settings used by the welding machine, such as the welding pressure, voltage, speed and others. As these settings differ depending on the properties of the steel being welded, they are bundled into so-called *welding programs*. Because welding programs are well tested before being put in use, they are seldom the cause of poor welds. Factors related to the material (such as thickness deviations or surface pollutants) or related to the welding machine (such as alignment errors or residue on the weld wheels) are more common, but hard to pinpoint.

Our work is structured as follows. First, we give an overview of weld quality assessment techniques in Section 2.2. Section 2.3 describes the data we used for our experiments and evaluations. We describe our base model in Section 2.4, and extend it to an incremental model in Section 2.5. Finally, we conclude our findings in Section 2.6.

2.2 Related Literature

Works discussing quality monitoring exist for all types of welding: laser welding [1], arc welding [2], spot welding [3, 4], resistance seam welding [5, 6, 7] and mash seam welding [8, 9]. Here, mash seam welding is a type of resistance seam welding where the overlap between welded sheets is minimal [10]. Furthermore, resistance seam welding can be seen as a series of (overlapping) spot welds [7]. Though the other welding techniques use different principles, quality assurance techniques are somewhat transferable over all types of welding and are mainly chosen in function of restrictions imposed by the intended use case.

While destructive testing such as the Erichsen cupping test [11] can be used to acquire detailed insights in general welding practices, non-destructive testing is more interesting as it can be used for production quality control. Non-destructive testing includes methods based on external tools such as acoustics [2, 3, 12], Eddy currents [9, 13], radiography [14, 15], temperature [5] and visual checks, as well as checks using data from the welding process itself such as dynamic resistance [3, 4] or the monitoring of welding parameters [3, 4, 5].

Surprisingly, almost all documented cases of the steel production industry use parameter monitoring techniques. This can be somewhat explained by the restrictions of the continuous production process. For example, the use of acoustic methods may be unreliable in the noisy factory setting [16]. One specific work does focus on acoustics for mash seam welding for galvanizing lines, but only evaluates a prototype in a lab settings [12]. Radiography techniques require a specialised setup that may be diffi-

cult to include in a production line that was not designed for it, and may additionally introduce specific safety implications [16]. Other techniques may be similarly hard to install, or affected by the high electromagnetic influence of the welding machine [3]. Furthermore, some techniques require labeled examples of weld defects which requires additional labeling effort. The low number of bad welds in continuous lines further complicates data gathering as well, resulting in extremely unbalanced datasets and creating a need for data augmentation [15].

Works tracking weld parameters generally report weld speed, pressure, power and temperature as most influential for weld quality [7, 17]. Additionally, a strong connection between temperature and power has been reported, meaning it might suffice to utilise only one of these signals [7]. The use of parameter checks can be divided in two main categories. The first category focuses on the pattern of the signal throughout the weld, where a high variation is indicative of a bad weld [1, 3, 6]. The second category focuses on the values of the signal. Here, weld values are compared to similar previous welds and abnormal values are flagged as bad welds [3, 4, 5].

Four publications specifically focus on the same use case as our work, i.e. weld quality for continuous lines in steel mills. All were developed and deployed at the ArcelorMittal site in Asturias, Spain. In the first work [17], the average temperature and current of each weld is compared against historical welds of similar welding programs. Using the outer 1 and 5 percentile values as reference, a score is calculated for both signals. These scores are aggregated and used to classify the welds. The next work [16] uses a pre-filtering check to find operating errors in setup voltage, speed and pressure using historical data. Next, weld temperature values are assigned an assessment score (based on the most recent welds in the same welding program), sorted, and the 10-percentile value is used as a score. This approach comes down to checking how many datapoints fall below or above a threshold defined by historical data. The third work [8] combines and expands the previous two. For the speed, welding pressure, flattening pressure, temperature and current signals, three assessment scores are calculated based on the magnitude, slope and noise. Again, the score mechanism is defined per welding program and uses four percentile based thresholds calculated on the last N welds. All assessment scores are aggregated and compared against an expert defined threshold to obtain a final classification. The most recent work [18] uses statistical features that are taken from six different signals, as well as several geometrical and chemical features of the steel coils. These features are transformed so they follow a normal distribution. Univariate and multivariate Gaussian models estimate the probability of the features for each weld, probabilities lower than a specified threshold are flagged as bad welds. Notably, this approach does not use separate models per welding program. Their experimental setup did require high computational power and was performed using a high-performance computing facility.

In this work, we present a novel way to estimate the expected weld current, using the weld program settings. Our model works incrementally, i.e. it can adjust for underlying trends caused by maintenance or material variations. As such, our method is compatible with any of these four aforementioned works by utilising the estimation in

the assessment calculation for the welding current. Of course, applying our suggested method to additional features is also possible, though this is not in scope of this work.

2.3 Data Description

Our data originates from the welding machine used in the continuous annealing and galvanization line of ArcelorMittal Belgium and spans a period from March 2019 up to August 2020. For each weld, a record was made of the metadata of both coils, setup welding parameters, used welding parameters, as well as statistical values for all recorded welding measurements. Metadata included thickness, width, material type, and the identifier of both coils, as well as a timestamp. Setup parameters include all settings for the welding machine, as defined by the selected welding program, such as weld speed, machine voltage, welding pressure, planning pressure, and settings for the overlap of both coils. Used parameters are the settings effectively used for the welding process, these match the setup values unless the operator has intervened. The recorded values encompass everything measured during the actual welding process, such as weld speed, pressures, current, temperature, and others. During the welding process, these features are sampled at a 50Hz rate and sent to the process computer. Here, the part of the signal corresponding to the actual weld is extracted and used to compute statistical features such as the average, median and standard deviation.

As the data collection process is fully automated, weld tests and welds made during maintenance periods are also included. Unfortunately, due to the way the data collection worked, records for these welds turned out to be unreliable due to a software flaw where data of new coils was loaded before rewelding was completed. As there was no feature available that described the type of weld (i.e. normal, reweld or weld tests), we used various sanity checks and data originating from other parts of the production line to filter out welds where records may have been incorrect. For ease of evaluation in later experiments, we also removed any welding programs where less than 10 welds were present in the data set. After filtering we obtained records for 19910 welds, comprising over 111 different welding programs.

During preliminary data analysis, we discovered two peculiarities in the data. The first one is related to the welding current, and is visualised in Figure 2.1. This figure shows the median measured welding current for all welds made using one specific welding program. While we see slight trends throughout time which might be explained due to slight changes in the chemical properties of the steel over time [17], two large jumps stand out. The first jump (September 2019) corresponds with a maintenance of the welding machine, where a copper conductor was replaced. The second jump (March 2020) corresponds to another maintenance period, though no parts were replaced. After discovering the first current jump, a new copper conductor was ordered and installed in July 2020 to correct this anomaly, though without any observable effect. In order to take these jumps into account, we added a *maintenance period* feature to all welds, indicating to which timespan they belong, as shown in Figure 2.1. As

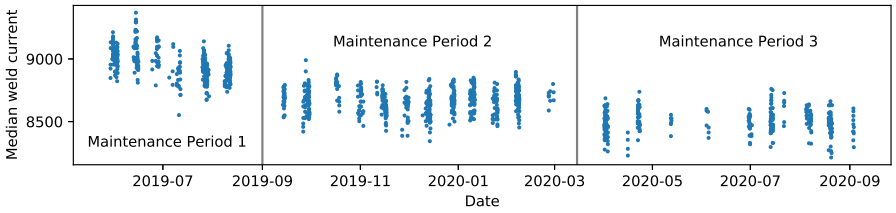


Figure 2.1: Median welding currents for a single welding program. Besides minor localized trends, there are two major jumps visible. The first jump coincides with the replacement of a welding machine component, while the second jump has no clear explanation. A later maintenance (July 2020) to replace the same part and correct the first jump has no visible effect. We utilise the maintenance period information in the first version of our model.

no rise in the number of rewelds or broken welds was noticed, the question remains whether this effect is due to changes in how the current is measured, or whether the welding current effectively dropped without affecting weld quality.

We discovered a similar peculiarity for the measured welding temperature with a higher frequency rate, sometimes as short as two weeks, which are probably related to slight changes in position of the sensor. Because the temperature measurements were unreliable, they are not considered in the remainder of this work.

While imperfect data collection and unexplained data anomalies are undesired, they are not uncommon. Instead, techniques that can deal with these challenges need to be found. As we will show later in this work, our technique can adapt to changes such as those described here, while still alerting operators of major behavioral changes.

2.4 Current Prediction Model

The method described in this work started from a desire to improve the interpretability of the weld quality system. At the time of writing, ArcelorMittal Belgium uses a similar weld quality system as described for their Spanish site, where aggregated measurements are compared against historical welds made using the same welding program. This approach works, but provides no insight as to why the historical records are what they are. An insightful model is useful as it allows engineers to have more trust in the method. Furthermore, any insights may help to understand other parts of the welding process. After initial experiments using linear models, we found that a model based on known physical laws provided good prediction capabilities.

2.4.1 Physics-Inspired Model

Our model is based on two physical laws. The first is the well known Ohm's law, shown in Equation 2.1, which specifies the connection between a voltage V (measured

in Volts), a current I (Amperes) and a resistance R (Ohms). In the welding process, the voltage is determined by the voltage setting (a value in the range $[0..100]$) as specified in the welding program. The resistance represents the combined resistance of the welding machine, the steel plates and any contact resistance.

$$I = \frac{V}{R} \tag{2.1}$$

The second law we utilise is Pouillet's law, shown in Equation 2.2. This law gives the resistance R of a material in function of the contact surface A (in square metres), the length of the material l between both contact points (metres), and the resistivity ρ of the material (Ohm metres). We use this formula to estimate the resistance of the steel coils during the welding process. Note that Pouillet's law describes an ideal case with uniform contact between the conductors and the material, which is certainly not the case when using round welding wheels, but we found that incorporating this formula worked well in practice.

$$R = \rho \frac{l}{A} \tag{2.2}$$

The resulting model, which we will use for weld current prediction, is shown in Equation 2.3. Here, $\hat{I}$ is the prediction of the measured current, V_{out} is the voltage applied according to the welding program, R_{mach} is the combined resistance of the welding machine and welding wheels (which is affected by the maintenance period) and R_{coil} represents the resistance created by the coils. The coil resistance is estimated using a resistance factor for the type of steel ρ, the thickness t of the coil, the surface created by the weld wheel A and a linear correction factor B in function of the welding pressure p_{weld}.

$$\begin{aligned} \hat{I} &= \frac{V_{out}}{R_{mach} + R_{coil}(head) + R_{coil}(tail)} \\ R_{coil}(x) &= \frac{\rho(x)t(x)}{A + p_{weld}B} \end{aligned} \tag{2.3}$$

2.4.2 Training the Model

At this point, the prediction formula still contains many unknown terms. We could try to measure or estimate these terms, though this proves difficult in practice. For example, the resistance factor of steel could be measured outside of the welding process, but this would neglect the effect of the high welding temperature on the resistance. Instead, we determine suitable values for all model variables using gradient descent, a technique commonly used for training neural networks.

We implemented the formula from Equation 2.3 in a TensorFlow [19] model, and defined trainable variables for the unknown terms in the formula. These terms are: V_{out} (a non-linear mapping that we discuss in the next section), R_{mach} (a scalar or

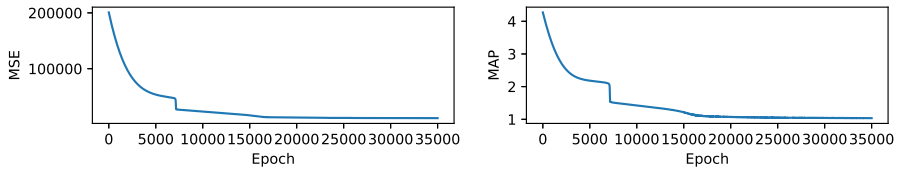


Figure 2.2: Mean squared error (MSE) and mean absolute percentage error (MAP) during training for the experiment described in Section 2.4.4. The sudden jump corresponds to stabilization of the B parameter.

vector, depending on the experiment setup), ρ (a vector with one value per type of steel considered), A (a scalar) and B (a scalar). This resulted in a model with at most 24 trainable parameters.

We initialise the model using rough estimates for all trainable values and use TensorFlow's Adam optimizer to minimize the prediction error on training data. We selected this optimizer as it is well established in the Tensorflow community and provides good performance without the need for extensive parameter tuning. We found a learning rate of 7.5×10^{-4}, epsilon (a small constant for numerical stability) of 0.1, and the mean squared error as optimisation metric worked best, even though we will evaluate mainly using the mean absolute percentage (MAP) error, as this error value is more interpretable. The model converges after around 30000 epochs, which takes less than an hour on a moderate desktop (Intel i7 920, 6GB RAM). The training process is visualised in Figure 2.2.

2.4.3 Modeling the Output Voltage

The welding voltage is actually a non-linear function of the value specified in the welding program. This is because the setup value actually defines the range during which the welding thyristor connects the voltage (a rectified sinusoidal signal), as shown in Figure 2.3 (Left). Based on this, the theoretical output voltage can be calculated using the formula given in Equation 2.4, where s represents the starting angle, defined by the setup value V_{in}.

$$V_{out} \propto \int_{s}^{\pi} \sin x \, dx \tag{2.4}$$

However, we found measurements made during welds deviated from theoretical values, which can be attributed to the inductive properties of the welding setup. We experimented with multiple methods to model the output voltage, including variants of Equation 2.4 that better resembled our measurements. However, we obtained the best results using 11 reference points, spread uniformly over the range of setup values. For these points, we trained the corresponding output voltages as described in the previous

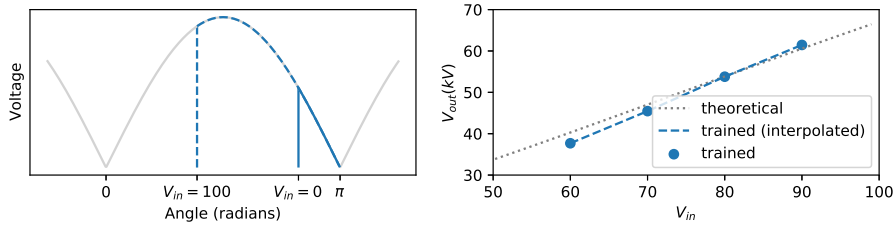


Figure 2.3: Left: Visualisation of how setup voltage V_{in} determines the period during which the circuit is completed by the thyristor, passing the voltage. Right: Theoretical mapping of V_{in} to V_{out} versus the values obtained by training reference values in our model. Only 4 out of 11 values are shown, as all welding programs use values within this range.

section. Remaining setup voltages are mapped to output voltages using linear interpolation using these 11 reference values. This way, reference voltages affect a range of input values, which helps to prevent overfitting and keeps the number of trainable parameters low. The difference between the theoretical and learned output voltages is shown in Figure 2.3 (Right).

2.4.4 Evaluation - Predictive Power

In this first experiment we validate that the physics model is able to correctly model the welding current. Additionally, we introduce the average-based model, which we use as a baseline model throughout this work. We train both models using a common methodology for regression problems where we determine model variables using training data and evaluate on test data. We show both models have a similar performance, which is mainly interesting as a reference point for the later experiments.

We split the available weld data in train and test data following a 80/20 ratio and keeping the ratio of weld programs in both sets similar. We trained our model as described in Section 2.4.2, with R_{mach} defined as a vector of length 3, one value per maintenance period.

The average-based model is a second current prediction model based on state of the art literature. While no work focuses specifically on welding current prediction, most works similar to our use case use statistical methods to compare weld signals (including current) to determine weld quality [8, 16, 17, 18]. Based on these, we define the average-based model as a model that stores a single prediction value per welding program, i.e. the average current of the training welds made using the corresponding welding program. Additionally, we enhance this model with the maintenance period feature, as we did for the physical model. The resulting model has one trainable parameter per welding program per maintenance period, totalling at 325 parameters for our data (this is not a multiple of three since not all programs were used in each period).

	Physical Model	Average-based Model
Train MSE	11 544.88	11 090.66
Train MAP	1.04	1.02
Train Reject (%)	2.7	2.5
Test MSE	10 844.92	10 800.81
Test MAP	1.01	1.01
Test Reject (%)	2	2.3

Table 2.1: Prediction errors in mean squared error (MSE), mean average percentage error (MAP) and percentage of rejected welds for the first version of our model versus the average-based model. Welds were rejected if the difference between measured and predicted current was higher than 3%. We see that the predictive power of both models is similar.

We use the same train/test datasets for evaluating the average-based model as we did for the physical model.

Table 2.1 shows the prediction errors for both models. We see that both models have a similar performance, with the physical model having a slightly higher error on the training data. Overall, the predictions are very good, with an average prediction error close to one percent. Of course, the MSE and MAP errors give no insight as to how many welds would be rejected using both models. To do this, we set a prediction error threshold of 3 percent, which was advised by a process engineer. This means that any weld where the difference between predicted and measured welding current is greater than 3 percent would be rewelded. Table 2.1 confirms that both models reject a similar number of welds.

At this point, we have shown that our physical model has a similar performance as the average-based baseline model. However, the physical model is fully explainable and uses less than a tenth of the parameters of the average-based model. In the next experiment, we further validate our claim that our model is correct from a physics point of view, and can therefor be interpreted.

2.4.5 Evaluation - Physical Soundness

In this experiment, we test whether the current jumps shown in Figure 2.1 can be explained using only the machine resistance parameter. We first train the physical model, i.e. all model parameters, using all welds from a single maintenance period $P1$. Next, we adjust the trained model to predict welds from a different maintenance period $P2$. As we assume the difference between maintenance periods is caused by changes to the machine resistance, it should suffice to update only this single model parameter. We verify this by randomly selecting a (small) number of welds from $P2$ as training data and updating the machine resistance parameter. Using this updated model, we predict the current for all remaining welds in $P2$.

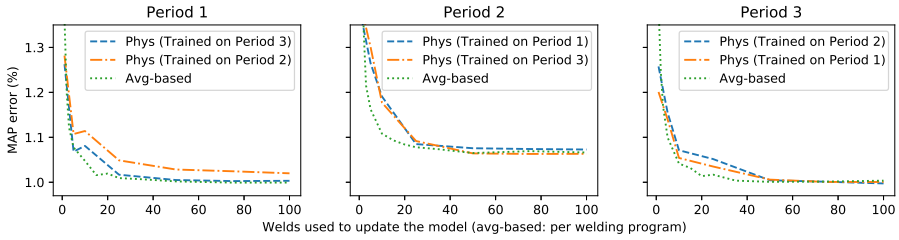


Figure 2.4: Evaluation results (averaged over 10 runs) where we trained the physical model on one maintenance period $P1$, updated the machine resistance parameter using a small number of welds from a different period $P2$, and predicted all remaining welds of $P2$. The average-based model was trained using only $P2$. We see how the prediction error converges to that of the average-based model, indicating that tweaking the machine resistance parameter is in fact sufficient to reuse the model across maintenance periods.

We compare performance against the average-based model. Because the average model would actually be at a disadvantage if it retained data from $P1$ to predict $P2$, we assume the average model only uses the available data from $P2$. Additionally, we select the training data for this model in such a way that all welding programs are equally represented. While this does inflate data requirements disproportionately, it gives the best possible prediction capability, which is the focus of this experiment.

We repeated the experiment ten times, averaging the results. The results are shown in Figure 2.4, where we see how the physical model achieves good predictions after updating using as little as 50 welds. This confirms that all model parameters can be reused across the maintenance periods. The results seems to hint that the average-based model stabilises using fewer welds, but actually the opposite is true, as the actual weld count should be multiplied by the number of welding programs. We should note it is difficult to determine the number of welds required for reliable predictions when using randomly sampled data due to the welding program imbalance in the data, as the effect on the prediction score will be larger if more commonly used welding programs are sampled. A comparison for a more realistic setting can be made using the experiment in the next section.

In the next section, we further extend our model to automatically update the machine resistance parameter, so it can be more easily deployed in a production setting.

2.5 Incremental Current Prediction Model

In the previous section, we have explained how the physical model uses less internal parameters and is more understandable for experts, while still having similar prediction results as the average-based model. Nevertheless, the model would still be difficult to implement in a production setting. In the previous experiments we have used the knowledge of the transitions either as a feature or as a trigger to retrain the model, but

this would not be possible in a real production setting. We could act reactively, i.e. once a transition is suspected, a data scientist would need to confirm this suspicion and retrain the model, and the new model would have to be updated on the process computer. Because such interventions cannot be planned ahead, several days may pass before the model is eventually corrected after a transition. Alternatively, we could periodically retrain the model, though this is somewhat complicated as the predictions are coupled to the process computer, which is not suited for the specialised training process. Furthermore, as we will show in this section, local temporal trends are better captured by a dynamic model. Next, we present a simple approach to update the resistance parameter of our physical model in an effective way.

2.5.1 Updating the Model

Updating the machine resistance parameter in the model involves two mechanisms. First, we have to determine the value that we want to evolve towards. Secondly, we need to update the value in a suitable manner. The first part is straightforward and is shown in Equation 2.5, which we derive from Equation 2.3. Here, we use the measured current I of each weld to calculate the ideal machine resistance $\hat{R}_{mach}$, i.e. the value that would have predicted the welding current perfectly.

$$\hat{R}_{mach} = \frac{V_{out}}{I} - R_{coil}(head) - R_{coil}(tail) \tag{2.5}$$

Next, we use $\hat{R}_{mach}$ to update the resistance value in the model using an exponential smoothing function, as shown in Equation 2.6. Here, R'_{mach} represents the updated value, R_{mach} the old value, and α is a value in $[0 \ldots 1]$. By updating the model in this way, we minimize the effect of high-frequency noise while still following the underlying trend. Alternatively we could use a sliding average over the last N values, but by using exponential smoothing we effectively put more emphasis on recent values.

$$R'_{mach} = (1 - \alpha)R_{mach} + \alpha\hat{R}_{mach} \tag{2.6}$$

2.5.2 Evaluation

In this last experiment we want to fully capture the effects of underlying temporal trends. As such, we predict all welds in chronological order, as would be done in a production setting. We assume all welds in the first maintenance period (4679 welds) are available for training purposes, and perform a warm-up run for all models by predicting all welds from this period. We only report errors for the unused welds of period 2 (9942 welds) and 3 (5289 welds).

All model parameters (except α) of the incremental physical model were determined using the training data, afterwards the resistance value was updated by predicting all welds chronologically. After each prediction, the resistance weight was updated using the actual measured weld current.

We compare against two incremental versions of the average-based model and an oracle model. The first average-based model (Avg) predicts welding current as the average of the last N measured welds in the same welding program. The second model (Avg ES) works similarly as the first, but uses exponential smoothing to update the prediction value once the model has N historic values for the welding program. We only use exponential smoothing after N values to minimize the effect of noise on the first predictions. Finally, the oracle model is a non-updating version of the physics model that was trained on all data of all three periods and tracks a different resistance for each period. This means the oracle model is evaluated on the same welds it was trained on.

For all models, we calculate the MAP error for all welding currents in period 2 and 3. To estimate how quickly each model adapts, we also calculate the MAP error when ignoring a predefined number of welds after either transition. Table 2.2 lists the prediction errors and gives several interesting insights.

Looking at both variants of the average-based model, we see better performance when the model can adapt more quickly. This is demonstrated in the first variant for the lower window size N and in the second variant for the higher α value. Furthermore, we see that all average-based models perform better if we exclude more welds after either maintenance. This shows that these models would need several days to adjust to the changed behavior.

Looking at the physics model, we see improved predictions for faster reactivity (higher α), but do not see major improvement after skipping 50 to 100 welds. This demonstrates the faster update speed of the physical model over the average model. The incremental physical model clearly outperforms both variants of the average-based model and even the oracle model. The former can be attributed to the single shared parameter in the physics model, which adjusts the model for all welding programs, whereas the average model has to update these independently. The latter can be attributed to the fact that the oracle assumes the absence of local trends.

To visualise performance over time, we again define rejected welds as those with a prediction error greater than 3% and plot the total number of rejected welds over time in Figure 2.5. Here, we see that despite major differences in the slopes for all models, some patterns are conserved, meaning there is a degree of consensus between all models. Overall, the average-based models have more rejected welds. Noticeably, both transitions are followed by a sudden increase as the average-based models have to adjust the prediction value for all welding programs. In contrast, the physical models exhibit only minor jumps after either transition that further diminish for the more adaptive models. Note that small jump artefacts actually may be desired by the operator as a way to notice a fundamental change to the system. In this way, a series of rejected welds over a short period could instigate further investigation.

All findings point to the presence of both minor and major variations over time of the welding current that may be related to the welding machine or input materials over time. The updating physics model is significantly better at keeping up with these changes. In this experiment, the physics model would have rejected around 250

	MAP Error (%)				
Excluded welds	0	50	100	250	500
Incr. Physics ($\alpha = 0.005$)	0.984	0.976	0.968	0.956	0.954
Incr. Physics ($\alpha = 0.01$)	0.956	0.949	0.944	0.941	0.943
Incr. Physics ($\alpha = 0.02$)	0.931	0.925	0.923	0.923	0.925
Incr. Physics ($\alpha = 0.05$)	0.892	0.888	0.887	0.888	0.889
Incr. Physics ($\alpha = 0.10$)	0.860	0.857	0.856	0.857	0.858
Avg ($N = 25$)	1.165	1.160	1.156	1.126	1.094
Avg ($N = 50$)	1.263	1.256	1.249	1.215	1.171
Avg ($N = 100$)	1.394	1.388	1.379	1.342	1.293
Avg ES ($N = 25, \alpha = 0.01$)	1.564	1.556	1.548	1.511	1.461
Avg ES ($N = 25, \alpha = 0.02$)	1.344	1.338	1.330	1.295	1.249
Avg ES ($N = 25, \alpha = 0.05$)	1.166	1.161	1.156	1.125	1.091
Oracle (Non-Incr. Physics)	1.028	1.027	1.026	1.025	1.022

Table 2.2: Welding current prediction error of all model for all welds in maintenance period 2 and 3, excluding a number of welds that occur just after either transitions. All average-based models perform best when ignoring up to 500 welds after each transition, indicating they require a long time to adjust after a transition. In contrast, the physics-based models perform optimal after 50 to 100 welds. Overall, the physics model outperform the average-based models and even the oracle model. This shows that welding current is affected by local temporal changes captured by the updating physics model.

(1.6%) welds, whereas the best average-based models rejected over 800 welds (5.2%). The process engineer confirmed the value of these results, as it meant that the current monitoring system would be less affected by unexpected shifts, while still signaling these shifts.

2.6 Conclusion

In this work, we present a novel approach to model welding current in mash seam welding, applied to weld quality control in continuous production lines of steel mills. Where state of the art methods rely on statistical approaches, our model is based on physical laws, making it highly interpretable. Model parameters are determined using the open source TensorFlow framework, after which the model can be easily implemented in constrained environments, such as the process computer.

We evaluated our model using 15 months of collected data from the ArcelorMittal Belgium site, which exhibits both sudden and gradual changes in welding current. When comparing non-incremental models, our model achieves similar prediction scores as statistical methods. However, our incremental model clearly outperforms the incremental statistical methods, because it only updates a single parameter whereas

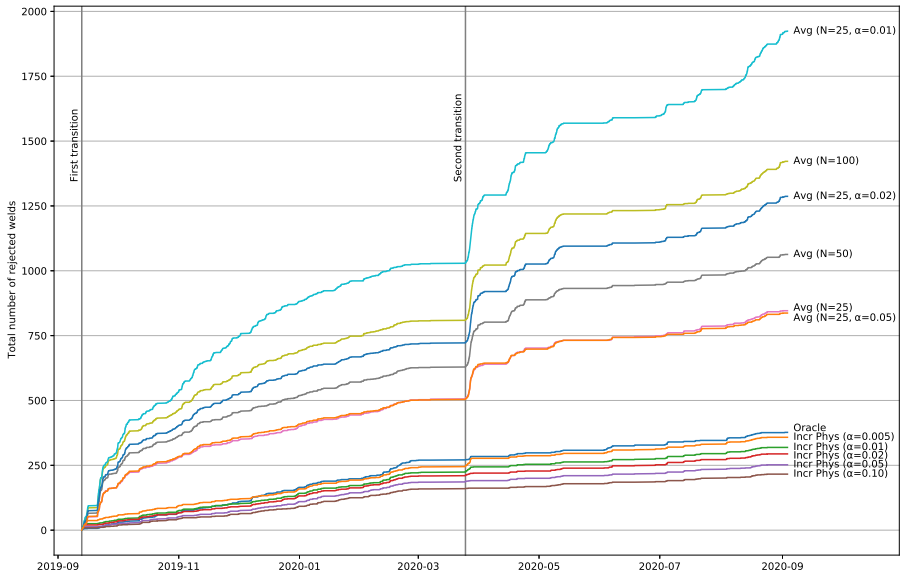


Figure 2.5: Cumulative number of rejected welds (on a total of 15231 welds) over time for models trained exclusively on data before the first transition. Welds are rejected when the prediction error is larger than 3%. The average-based models have noticeable jumps after both transition periods, the timings of the jumps can be attributed to batches that require different welding programs. The physical models only exhibit minor jumps from this effect.

the statistical models need to update one parameter per welding program.

Our work focuses solely on current prediction, but can be easily combined with other quality assurance techniques described in literature. At the time of writing, ArcelorMittal Belgium was incorporating our incremental model into their production process.

References

[1] Amit Kumar Sinha, Duck Young Kim, and Darek Ceglarek. "Correlation analysis of the variation of weld seam and tensile strength in laser welding of galvanized steel". In: *Optics and lasers in engineering* 51.10 (2013), pp. 1143–1152.

[2] A Sumesh, K Rameshkumar, K Mohandas, and R Shyam Babu. "Use of machine learning algorithms for weld quality monitoring using acoustic signature". In: *Procedia Computer Science* 50 (2015), pp. 316–322.

[3] Yanhua Ma, Pei Wu, Chuanzhong Xuan, Yongan Zhang, and He Su. "Review on techniques for on-line monitoring of resistance spot welding process". In: *Advances in Materials Science and Engineering* 2013 (2013).

[4] Kang Zhou and Ping Yao. "Overview of recent advances of process analysis and quality control in resistance spot welding". In: *Mechanical Systems and Signal Processing* 124 (2019), pp. 170–198.

[5] Vinicius Santos de Deus, Jose Adilson Castro, and Sandro Rosa Correa. "Correlation Among the Input Thermal Parameters and Thermography Measurements Data of the Resistance Seam Welding". In: *Materials Research* 23.1 (2020).

[6] T Mira-Aguiar, C Leitão, and DM Rodrigues. "Solid-state resistance seam welding of galvanized steel". In: *The International Journal of Advanced Manufacturing Technology* 86.5 (2016), pp. 1385–1391.

[7] Alireza Khosravi, Ayyub Halvaee, and Mohammad Hossein Hasannia. "Weldability of electrogalvanized versus galvanized interstitial free steel sheets by resistance seam welding". In: *Materials & Design* 44 (2013), pp. 90–98.

[8] Julio Molleda, Juan L Carús, Rubén Usamentiaga, Daniel F García, Juan C Granda, and José L Rendueles. "A fast and robust decision support system for in-line quality assessment of resistance seam welds in the steelmaking industry". In: *Computers in industry* 63.3 (2012), pp. 222–230.

[9] Rui Miao, Yuntian Gao, Liang Ge, Zihang Jiang, and Jie Zhang. "Online defect recognition of narrow overlap weld based on two-stage recognition model combining continuous wavelet transform and convolutional neural network". In: *Computers in Industry* 112 (2019), p. 103115.

[10] JE Gould. "Theoretical analysis of welding characteristics during resistance mash seam welding of sheet steels". In: *Welding journal* 82.10 (2003), pp. 263–267.

[11] A Kocańda and C Jasiński. "Extended evaluation of Erichsen cupping test results by means of laser speckle". In: *Archives of Civil and Mechanical Engineering* 16 (2016), pp. 211–216.

[12] SS Indimath, R Shunmugasundaram, S Balamurugan, M Dutta, SK Gudimetla, and K Kant. "Online ultrasonic technique for assessment of mash seam welds of thin steel sheets in a continuous galvanizing line". In: *The International Journal of Advanced Manufacturing Technology* 91.9 (2017), pp. 3481–3491.

[13] Javier García-Martín, Jaime Gómez-Gil, and Ernesto Vázquez-Sánchez. "Non-destructive techniques based on eddy current testing". In: *Sensors* 11.3 (2011), pp. 2525–2565.

[14] Wenhui Hou, Dashan Zhang, Ye Wei, Jie Guo, and Xiaolong Zhang. "Review on computer aided weld defect detection from radiography images". In: *Applied Sciences* 10.5 (2020), p. 1878.

[15] Haodong Zhang, Zuzhi Chen, Chaoqun Zhang, Juntong Xi, and Xinyi Le. "Weld defect detection based on deep learning method". In: *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2019, pp. 1574–1579.

[16] Ruben Usamentiaga, Julio Molleda, and Daniel F Garcia. "Real-time assessment of the reliability of welds in steel strips". In: *IEEE Transactions on Industry Applications* 46.1 (2009), pp. 81–88.

[17] Julio Molleda, Daniel F García, Diego González, Iván Peteira, and JA Go. "Fuzzy-based approach to real-time detection of steel strips defective welds". In: *CIMSA. 2005 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, 2005.* IEEE. 2005, pp. 169–174.

[18] Julio Molleda, Juan C Granda, Rubén Usamentiaga, Daniel F García, and Dave Laurenson. "A quality inspection system for resistance seam welds in endless production of steel coils using anomaly detection techniques". In: *2012 IEEE Industry Applications Society Annual Meeting*. IEEE. 2012, pp. 1–8.

[19] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

Chapter 3

A Generalized Matrix Profile Framework with Support for Contextual Series Analysis

Chapter 1 demonstrated the wide range of time series analytics methods that are based on the retrieval of similar subsequences using the matrix profile. However, similarity is quite a versatile constraint, as it is defined by one of many possible similarity measures such as those mentioned in Section 1.2.2. Conceptually, any measure can be used with any technique, though this proves difficult in practice as it would require custom implementations for each variant. Even with all variant techniques available, the situation is still not ideal since implementations are independent, meaning calculations are duplicated if an analyst chooses to apply several techniques. The first part of this chapter tackles this issue by introducing a plug-and-play framework for the matrix profile that allows users to freely and efficiently combine matrix profile techniques.

The second part of the chapter introduces a new technique that is can visualize repetitions in a series. Where multidimensional scaling (shown in Section 1.3) extracted motifs to visualize their similarity in a lower dimension, our technique shows which regions in time series are similar to each other. The visualization can be seen as a summarized distance matrix, guided by expert knowledge of the analyst. By visualizing the patterns in a time series, we can easily spot anomalies in time series that are not discords.

My contributions can be summarized as follows:

1. Introducing the Series Distance Matrix (SDM) framework for combining the various matrix profile variants, thereby facilitating data analytics. The open source implementation is used for all following chapters in this book.

2. Introducing the Contextual Matrix Profile (CMP) as a new data analytics technique with direct applications for visualization and anomaly detection.

A Generalized Matrix Profile Framework with Support for Contextual Series Analysis

D. De Paepe, S. Vanden Hautte, B. Steenwinckel, F. De Turck, F. Ongenae, O. Janssens, and S. Van Hoecke

Published in "Engineering Applications of Artificial Intelligence"

Abstract The Matrix Profile is a state-of-the-art time series analysis technique that can be used for motif discovery, anomaly detection, segmentation and others, in various domains such as healthcare, robotics, and audio. Where recent techniques use the Matrix Profile as a preprocessing or modelling step, we believe there is unexplored potential in generalizing the approach. We derived a framework that focuses on the implicit distance matrix calculation. We present this framework as the Series Distance Matrix (SDM). In this framework, distance measures (SDM-generators) and distance processors (SDM-consumers) can be freely combined, allowing for more flexibility and easier experimentation. In SDM, the Matrix Profile is but one specific configuration. We also introduce the Contextual Matrix Profile (CMP) as a new SDM-consumer capable of discovering repeating patterns. The CMP provides intuitive visualizations for data analysis and can find anomalies that are not discords. We demonstrate this using two real world cases. The CMP is the first of a wide variety of new techniques for series analysis that fits within SDM and can complement the Matrix Profile.

3.1 Introduction

The need for data analysis is increasing as more data is being recorded, stored and made available. One driving factor is the rise of the *Internet of Things* (IoT), where traditional *dumb* devices such as vehicles, household appliances or city infrastructure are enhanced with internet connectivity for monitoring and/or control. In 2018, there were an estimated 7 billion active IoT devices, and this number is expected to double in about 5 years [1]. Many sensors perform periodic monitoring, creating the need for a subdomain of data analysis: series analysis.

Series analysis techniques deal with ordered collections of data points, rather than independent data points. Time series are most common, measuring specific features across time. However, not all series are time series. For example, in [2], skull outlines in images are converted to a series for classification purposes. Unlike non-series, consecutive points in series carry meaning and patterns will often occur throughout the series. Finding and analyzing these patterns can allow better insights in the data.

From a business point of view, series analysis can lead to decreased costs. One such case is maintenance in industry [3]. Today, to prevent the high cost of unexpected machine breakdowns, machine owners perform preventive maintenance periodically. With condition-based maintenance, sensors monitor the health of a machine by recording and analysing time series data to gain insights. This way, machine health is known

and owners can better align planned maintenance with the actual need for maintenance, resulting in fewer interventions and decreased maintenance costs and machine downtime. A different business case can be made for trend prediction and anomaly detection [4]. Imagine an online service provider that monitors various metrics related to the usage and load of their services. If the provider is able to gain insight in the usage patterns of the service, he can anticipate certain trends and be made aware of unexpected behavioral patterns of their users. This not only allows the provider to allocate resources more dynamically, but also gives him more time to act on unexpected behavior that might lead to more severe issues.

One state-of-the-art series analysis technique is the Matrix Profile [5], introduced by Yeh et al. in 2016. Given two series $S1$ and $S2$, and a window length m, the Matrix Profile is a new series of length $|S1| - m + 1$ containing the distance between any window of $S1$ and its best matching window in $S2$. By itself, the Matrix Profile can be used to find the *top motifs* (the best matching subsequences in a series) and the *top discords* (the most unusual subsequences in a series). Subsequently, it can be used for anomaly detection in contexts where anomalies are defined by unique behavior. Since its inception, many techniques have been published that either extend the Matrix Profile or use it as a building block for new insights [6, 7, 8, 9, 10, 11, 12, 13, 14, 15].

While much progress has been made by going forward with the Matrix Profile, we believe there is also value in taking a step back. One of the implicit steps during the Matrix Profile calculation is the fragmented calculation of the distance matrix of all subsequences of the two input series. In this chapter we present the Series Distance Matrix (SDM) framework as the base building block on which specialized techniques can be built, rather than the Matrix Profile itself. To the best of our knowledge, we are the first to present such an overarching framework. Whereas several methods to calculate the distance matrix have been published [5, 6, 16, 13, 14], they have never been suggested as (part of) an overarching framework.

The presented SDM framework separates components that calculate distances between subsequences of input series (*SDM-generators*) and components processing these distances in a meaningful way (*SDM-consumers*). Existing Matrix Profile extensions from literature can be packaged as either SDM-generators or SDM-consumers and plugged into the SDM framework. By separating these components, it becomes easier to combine different techniques freely without additional effort or overhead, resulting in a much broader arsenal of techniques that can be tried on new challenges. Furthermore, distances can be generated once but processed by multiple consumers in combined calculations, resulting in an overall more efficient solution. Lastly, because of this decoupling, components will be smaller, simpler and can be optimized independently from each other.

We also introduce the Contextual Matrix Profile (CMP) and a new SDM-consumer to calculate the CMP. The CMP can be seen as a configurable, 2-dimensional version of the Matrix Profile, that tracks multiple matches across window regions of the series whereas the Matrix Profile tracks one match for each window. Besides data visualization, it can also be used for detecting *anomalies that are not discords*. As a component

of SDM, the CMP can be calculated for any distance measure and can be calculated in parallel with other techniques such as the Matrix Profile.

To summarize, our contributions in this work are as follows: First, we use a new interpretation of the distance matrix to form the generalized SDM framework, which retrofits many published techniques in SDM-generators or SDM-consumers. As second contribution, we introduce the Contextual Matrix Profile as a new SDM-consumer. As final contribution, we created an open source Python implementation of our SDM framework, our CMP-consumer and several Matrix Profile-based consumer and generator implementations based on literature [5, 6, 10, 12, 17, 16, 15]. To the best of our knowledge, this is be the first Python library that provides an implementation combining this many techniques.

The remainder of this chapter is structured as follows: Section 3.2 gives an overview of literature regarding the Matrix Profile. In Section 3.3, we describe our SDM framework. Section 3.4 describes our CMP as well as the new SDM-consumer to calculte it. Its value is demonstrated for data visualization and anomaly detection for two real world datasets in Section 3.5. Finally, we conclude our findings in Section 3.6.

3.2 Background and Related Work

In this section, we formalize the definitions used in this work, summarize the core details of the Matrix Profile and list related literature.

3.2.1 Definitions

We start by defining the common concepts of *series* and *subsequences*.

Definition 3.1 (Series) A series $\boldsymbol{S} \in \mathbb{R}^n$ is an ordered collection of n real values $(s_0, s_1 \dots s_{n-1})$.

Definition 3.2 (Subsequence) A subsequence $\boldsymbol{S}_{i,m}$ is the continuous subsequence of $\boldsymbol{S}$ starting at index i of length m: $(s_i, s_{i+1} \dots s_{i+m-1})$. The subsequence cannot be longer than the original series ($1 \leq m \leq n$) and has to fall completely within $\boldsymbol{S}$: ($0 \leq i \leq n - m$).

The distance measure used in the Matrix Profile is the *z-normalised Euclidean distance*. The reason for this is explained in the next subsection.

Definition 3.3 (Z-normalised series) The z-normalised series $\hat{\boldsymbol{S}}$ is constructed by transforming $\boldsymbol{S}$ so it has a mean $\mu = 0$ and standard deviation $\sigma = 1$: $\hat{\boldsymbol{S}} = \frac{\boldsymbol{S} - \mu_S}{\sigma_S}$.

Definition 3.4 (Z-normalised Euclidean distance) The z-normalised Euclidean distance $D_{ZE}(\boldsymbol{A}, \boldsymbol{B})$ between 2 series of equal length $A \in \mathbb{R}^m$ and $B \in \mathbb{R}^m$ is defined as the Euclidean distance D_E of the z-normalised series $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{B}}$.

$$D_{ZE}(\boldsymbol{A}, \boldsymbol{B}) = D_E(\hat{\boldsymbol{A}}, \hat{\boldsymbol{B}}) = \sqrt{(\hat{a}_0 - \hat{b}_0)^2 + \ldots + (\hat{a}_{m-1} - \hat{b}_{m-1})^2}$$

3.2.2 Matrix Profile

In 2016, Yeh et al. [5] published a novel technique to perform *series subsequence all-pairs-similarity-search* on two series, producing two new series: the Matrix Profile and the Matrix Profile Index. The Matrix Profile is defined as the vector containing the z-normalized Euclidean distances between each subsequence from the first series and its closest matching subsequence from the second time series. The Matrix Profile Index contains the subsequence index in the second series for each match.

Concretely, given two series $\boldsymbol{S1} \in \mathbb{R}^n$ and $\boldsymbol{S2} \in \mathbb{R}^k$ and a subsequence length m, the Matrix Profile $\boldsymbol{M} \in \mathbb{R}^{n-m+1}$ and Matrix Profile Index $\boldsymbol{I} \in \mathbb{R}^{n-m+1}$ are new series such that for each $i \in [0, n-m]$, $\boldsymbol{I}_i$ contains the index of the start of the subsequence of $\boldsymbol{S2}$ of length m that best matches $\boldsymbol{S1}_{i,m}$ and $\boldsymbol{M}_i$ contains the corresponding distance. In the case a *self-join* is performed where $\boldsymbol{S1} = \boldsymbol{S2}$, an additional constraint is added to prevent *trivial matches*, where subsequences match themselves or nearby subsequences.

The default distance measure used is the z-normalized Euclidean distance, which has been shown [18] to provide better results by removing the effect of a changing data offset over time and thus focussing more on shape instead of amplitude. Typical causes of a changing offset are wandering baselines in sensors or natural phenomena (e.g., the gradual change in temperature throughout seasons).

3.2.3 Related Work

Literature related to the Matrix Profile can be separated into 3 categories: related work focusing on a) the calculation of the Matrix Profile, b) techniques that gain insights from the Matrix Profile or the Matrix Profile Index, and finally, c) ideas from the Matrix Profile for tackling new problems.

a) Calculation of the Matrix Profile

The Matrix Profile was published together with the STAMP algorithm [5], an anytime algorithm to calculate the Matrix Profile (and corresponding Index) of a series of length n in $O(n^2 \log n)$ time. STAMP uses the MASS algorithm [19] to iteratively calculate the distances for each subsequence. Performance was later improved by the STOMP algorithm [6], which uses a dynamic programming technique to reduce the runtime to $O(n^2)$, at the cost of losing the anytime property. Another optimization came with the SCRIMP algorithm [16], which restores the anytime property while retaining the same complexity as STOMP. Finally, ACAMP provides another speed improvement by postponing some operations until the Matrix Profile is completed [13]. We extended the calculation to reduce the effects of noise when dealing with flat sequences [15,

20], others have made extensions for handling missing data points [21] and support for calculating the multidimensional Matrix Profile [10].

Several recent works have suggested different distance measures to be used in the Matrix Profile. Silva et al. [22] use the Matrix Profile with the (non-normalized) Euclidean distance to perform music recognition and thumbnailing. Akbarinia et al. [13] suggest that using the Euclidean distance, and more general p-norm might be more useful for data analysis in physics, statistics, finances and engineering. Though they present no evaluations, one can expect relevant results for cases where series are not subjected to wandering baselines [18], such as system monitoring. Another distance measure suggested is ψ-DTW [14]. The authors claim that for many application domains, the z-normalized Euclidean distance is too strict while looking for motifs and discords. The ψ-DTW measure performs a non-linear transformation along the (time) axis and can ignore a prefix or suffix of the subsequence being matched. The authors find improved results for domains such as motion tracking (e.g., athlete positioning, motion capture and gesture analysis) and music data mining, though they underline the difficulty of objectively evaluating the relevance of motifs and discords.

b) Gaining insights

Insight in a series can be gained using the Matrix Profile (Index). Motif and discord discovery consist of finding the top matching and worst matching subsequences in a series and can be solved quickly by finding the minima and maxima in the Matrix Profile [5]. Discord discovery can be interpreted as a form of anomaly detection (which has a wide range of applications in machine maintenance, healthcare or system monitoring). In cases where the user knows the type of pattern they are looking for, they can use the Annotation Vector [9] to transform the Matrix Profile before performing motif/discord discovery. Other insights are also possible such as finding gradually changing patterns [11] or finding changes in the underlying behavior being measured [12, 15].

c) Matrix Profile as a building block

The series motifs found by the Matrix Profile have been used for data visualization [7] and classification [8] techniques. Furthermore, a series summarization technique [23] has been published which uses *MPDist*, a distance measure that considers two sequences similar if they share many similar subsequences [24]. The calculation of MPDist involves finding the best match for all subsequences in both series. These could be found by performing a double Matrix Profile calculation, but can also be obtained in a single calculation by processing the subsequence distances in a different way.

As we can see, a wide range of techniques has emerged, most focusing on an aspect closely related to the Matrix Profile.

3.3 The Series Distance Matrix

Many of the works in Section 3.2 have started from the idea of the Matrix Profile and created a new algorithm to obtain one specific variation. Looking forward to the future,

we can expect the number of algorithms to rise dramatically as the different distance measures and processing methods are further expanded and combined. Instead, we propose to view these variations as instances of a more generalized framework which we call the *Series Distance Matrix* (SDM).

3.3.1 SDM: General Concept

We present SDM as a component based framework for deriving insights by processing pairwise distances of the subsequences of pairs of series (this includes self-joins by assuming two equal series). Given pairs of series, *SDM-generators* are responsible for calculating the distances between all pairs of subsequences. Because calculating the full distance matrix is not scalable, we instead calculate fragments of the distance matrix. These fragments are processed by the *SDM-consumers*, after which the fragment is discarded and a new fragment is calculated. Each consumer is responsible for processing all distance fragments in a way that provides certain insights.

Conceptually, the distance matrix fragments can take any form, however, columns and diagonals have proven to work well for the Matrix Profile. The column based approach is used by the STOMP algorithm [6], it has the advantage of being easier to implement and is more suited for cases where one series is being streamed in an online fashion, since each new data point results in one new column of distance matrix values. The diagonal approach is used by the SCRIMP [16] algorithm. By processing diagonal fragments of the distance matrix, the calculated distances of each fragment are spread over many different pairs of subsequences. This can be utilised by some consumers, such as the Matrix Profile, to provide approximate intermediate results when processing all data takes a long time, making it well suited for interactive use cases.

Figure 3.1 shows a schematic visualization of the Matrix Profile calculation fitted into the SDM framework.

By separating the distance calculation and processing, we can easily combine generators and consumers to our needs. For example, the techniques described by Akbarinia et al. [13] and Furtado Silva et al. [14] are a combination of the p-norm or ψ-DTW generator with a Matrix Profile consumer. Combinations that have not yet been researched, such as combining a ψ-DTW generator with an MPDist consumer, are - thanks to the SDM framework - just as straightforward. A second benefit is that multiple consumers can be configured for a single generator, instead of having to adjust the algorithms itself, this way reducing calculation overhead. Lastly, by adopting a component based design, each component can be optimized independent of the others. For example, if a faster way is found to calculate the z-normalized Euclidean distance, only one generator has to be updated, instead of every technique using the z-normalized Euclidean distance.

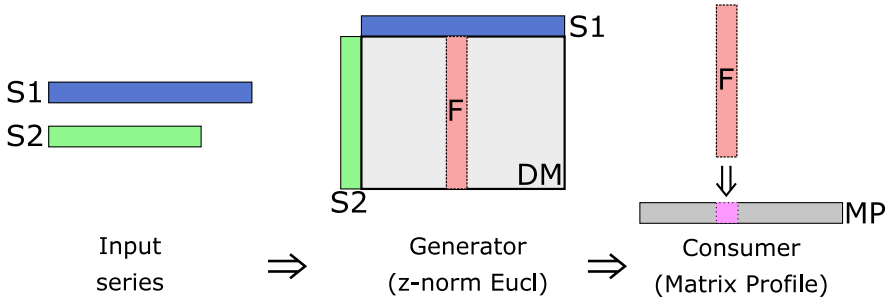


Figure 3.1: The Matrix Profile calculation fitted into the SDM framework. Starting from two input series (S1, S2), the z-normalized Euclidean distance generator iteratively creates fragments, in this case columns (F), of the distance matrix of all subsequences (DM). Each of these fragments are processed by the Matrix Profile consumer, storing the minimum value for each column in the resulting Matrix Profile (MP).

3.3.2 SDM: Python Implementation

As part of this work, we released a Python library[1] under the MIT license implementing our SDM framework and CMP consumer. In addition to the contributions of this work, it contains implementations for the noise-corrected z-normalized Euclidean distance ([5, 6, 16, 15]), Euclidean distance, Matrix Profile [5], Multidimensional Matrix Profile [10], Left- and Right-Matrix Profile [11] and VALMOD [17]. It supports batch operations as well as streaming data. At the time of writing, and to the best of our knowledge, this is the first public Python library integrating this many different Matrix Profile related work as consumers and generators in our generic framework.

3.4 Contextual Matrix Profile

This section covers a new series analysis technique, the CMP, which can easily find repeated patterns in series and shares the benefits of the Matrix Profile: it is deterministic, domain agnostic, exact and is suited for parallelization. The CMP is calculated by the CMP-consumer in the SDM framework. Note that thanks to the SDM framework, we can focus purely on how the calculated distances should be processed, since we can combine the CMP with *any distance measure* that has a corresponding SDM-generator implementation.

As the name implies, the CMP is closely related to the Matrix Profile, and can be best explained in how it differs from it. We make our comparison starting from the distance matrix (the implicit matrix containing the distances of all subsequences from the first input series to all subsequences from the second input series). Where the

[1] https://github.com/IDLabResearch/seriesdistancematrix/

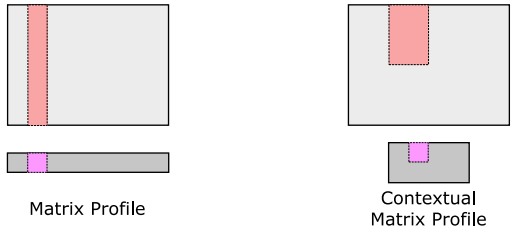


Figure 3.2: Matrix Profile and CMP differ in how they are created using the distance matrix (light gray). The Matrix Profile (dark gray, left) consists of the column-wise minimum of the values in the distance matrix. The Contextual Matrix Profile (dark gray, right) is created by taking the minimum over rectangular areas. Note that these areas may overlap and may or may not cover the entire distance matrix, depending on the user configuration.

Matrix Profile is defined as the column-wise minimum over the entire distance matrix, the CMP is defined as the minimum over rectangular regions of the distance matrix. These rectangles may overlap and may or may not cover the entire distance matrix. Their configuration is up to the user. A visual comparison of the Matrix Profile and the CMP can be seen in Figure 3.2. Note that the CMP-consumer may be configured in such a way that it calculates the Matrix Profile. In this way, the CMP can be seen as a generalization of the Matrix Profile.

Given two input series S_1 and S_2 and subsequence length m, the Matrix Profile looks for the best matching subsequence in S_2 for any subsequence in S_1. The CMP on the other hand looks for the best matching subsequence in ranges over S_1 and S_2. These ranges allow us to group the data in different ways and can reveal new insightful patterns. Specifically, because we aggregate the distances in ranges across both series, the CMP is very good at picking up repeated patterns, even if these patterns are not strictly periodic. We will show two use cases for the CMP, i.e., data visualization and anomaly detection, but first we discuss more thoroughly how the CMP is calculated.

3.4.1 Calculating the CMP

Many specialized algorithms could be conceived for specific region configurations. Here, we provide a general purpose algorithm. In this algorithm, the regions of interest are provided by specifying ranges along the dimensions of the distance matrix. This principle is illustrated in Figure 3.3. One advantage of this approach is that for non-overlapping ranges, the resulting CMP resembles a reduced distance matrix. We will exploit this property in our use cases below.

Our algorithm assumes the distance matrix is provided in a column-wise manner (similar to the STOMP algorithm [6]). A straightforward adaptation for diagonals is also made available in our reference implementation.

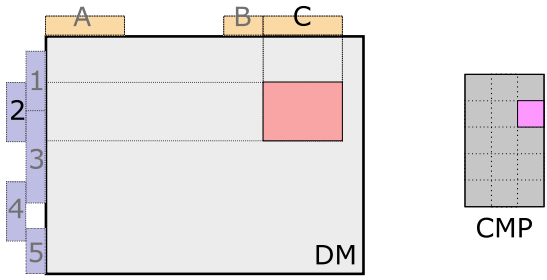


Figure 3.3: Example of region definitions: a user has specified three horizontal ranges (A, B, C) and five vertical ranges (1…5) on the axes of the distance matrix (DM). Any pair of ranges from both axes corresponds to one region of interest in the distance matrix. The minimum value of the region is calculated and stored in the CMP. Note that the ranges may overlap and may or may not fully cover the distance matrix dimensions.

Algorithm 1: CMP-consumer Initialization

Input: $R1$, ranges for the vertical axis of the distance matrix. A range is a pair defining a start (inclusive) and end (exclusive) index.
Input: $R2$, ranges for the horizontal axis of the distance matrix.

1 $v\_ranges \leftarrow R1$;
2 $h\_ranges \leftarrow R2$;
3 $cmp \leftarrow |R1| \times |R2|$ matrix, filled with $+\infty$;
4 $cmp\_index \leftarrow |R1| \times |R2|$ matrix, filled with $(-1, -1)$;

The initialization of the CMP-consumer is outlined in Algorithm 1. We take two lists of ranges as input, each defining the contexts for one of the input series. We store the ranges in line 1 and 2. Next, we prepare containers for the CMP and corresponding indices, similar to the Matrix Profile Index. Note that the CMP indices are two-dimensional since we need to track the exact match index for both input series.

The actual calculation of the CMP is listed in Algorithm 2. In line 1, we iterate over all ranges defined over the horizontal dimension of the distance matrix and skip any that do not contain the column being processed in lines 2-4. Next, we iterate over all ranges for the vertical axis. Since all ranges will have some overlap with the distance matrix column, we do not need to filter. In lines 6 and 7, we determine the minimum value of the distance matrix column that is contained in both ranges. We compare this minimum against the best value so far and update the distance and corresponding index if we find a better match (lines 8-12).

Note that when h_ranges is very long, a linear scan becomes inefficient. De-

Algorithm 2: CMP-consumer Column Processing

Input: The column index col.
Input: A vector d containing all distances on column col.

```
for j, h_range ← enumerate(h_ranges) do
    if col not in h_range then
        continue
    for i, v_range ← enumerate(v_ranges) do
        dists ← d[v_range];
        min_dist ← min(dists);
        if min_dist < cmp[i, j] then
            cmp[i, j] ← min_dist;
            row ← argmin(dists) + v_range[0];
            cmp_index[i, j] ← (row, col);
```

pending on the intended use, optimizations are obvious: tree maps for general cases, hash based lookup for strictly periodic ranges, or storing the search index for non-overlapping ordered ranges. In this section, we did not attempt to list all possibilities and instead presented the approach best suited for understanding the technique.

Lastly, we briefly discuss the complexity of the CMP. Strictly speaking, the space complexity is constant as it is determined by the configuration of the vertical (V) and horizontal (H) ranges: $O(|H||V|)$. When ranges will be defined in function of the length of the input series (n), $O(n^2)$ is more representative. Note that this last form is overly pessimistic as $|H|$ and $|V|$ will typically be much smaller than n. The time complexity for *processing a single column* is $O(|H| + |V| \times S)$, where S represents the average span of a vertical range. In a typical case where ranges will not overlap, this can be simplified to $O(n)$. As such, a full calculation can be done in $O(n^2)$, the same complexity as the calculation of the Matrix Profile using STOMP.

3.5 CMP for Data Visualization and Anomaly Detection

We will demonstrate the value of the CMP using two different use cases: data visualization and anomaly detection. For both cases, we use the public New York Taxi dataset and a dataset delivered to us by Renson (a ventilation manufacturing company) that we share as part of this publication [25]. Additionally, in our most recent paper [20], we combine the CMP with the noise elimination technique [15] to visualize a UCI activity dataset and show potential for activity segmentation as well. Note that it is not our goal to improve upon the state-of-the-art anomaly detection techniques in this section, but rather to show the potential of the CMP.

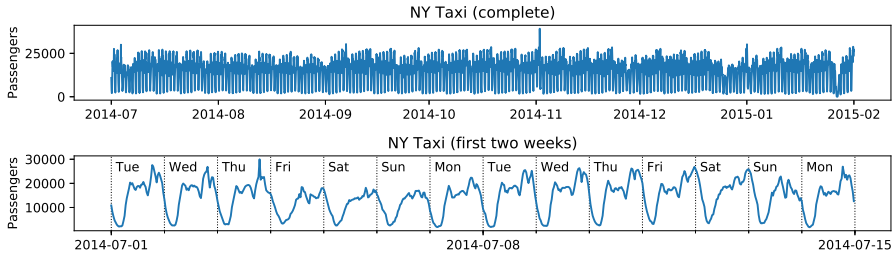


Figure 3.4: The New York Taxi dataset from the Numenta Anomaly Benchmark. It lists the summed number of taxi passengers in New York at 30 minute intervals. Top: Complete dataset. Bottom: The first two weeks of the dataset, where we see a clear periodic pattern. Note how the pattern for the first Friday, Independence Day, resembles the pattern for a weekend day.

All figures in this section were created using Python-based Jupyter notebooks, which we have shared online [25]. Besides providing an easy way to reproduce our results, they offer some additional visualizations we omitted due to size constraints.

3.5.1 New York Taxi Dataset: Data Visualization

The first dataset is the New York Taxi public dataset from the Numenta Anomaly Benchmark [26]. It lists the total number of taxi passengers in New York city for a period from July 2014 up to February 2015, bucketed per half hour. An overview and excerpt is shown in Figure 3.4.

We calculated the CMP by self-joining the data using the z-normalized Euclidean distance, using a window length of 44 (22 hours) and a daily context starting at midnight until 02:00 in the morning. Because we are self-joining the data, a constraint prevents any day from matching itself. Simply put, we are asking for the most (shape-wise) similar subsequences between any pair of days, where both subsequences are 22 hours long and can start between midnight and 02:00. These values were based on a quick visual inspection of the data. By choosing a two hour context range and a 22 hour window length, we allow temporal shifts when comparing windows, while always comparing values of the same day. Note that for slightly different values, we obtained similar results. Since the dataset contains 215 days and we define one context per day, the resulting CMP is a 215 by 215 matrix. It is shown in Figure 3.5. Note that the CMP is symmetrical because of the self-join, higher values in the CMP correspond to more dissimilarity.

When visualized, the CMP can be used to gain insight into the dataset it was built on. For example, the pattern of small squares visible in Figure 3.5 indicates that there are typically 5 days displaying similar behavior, followed by 2 days of different behavior. These patterns are of course caused by the cycle of weekdays and weekends. Other artefacts standing out are the wide band around New Year, near the end of November

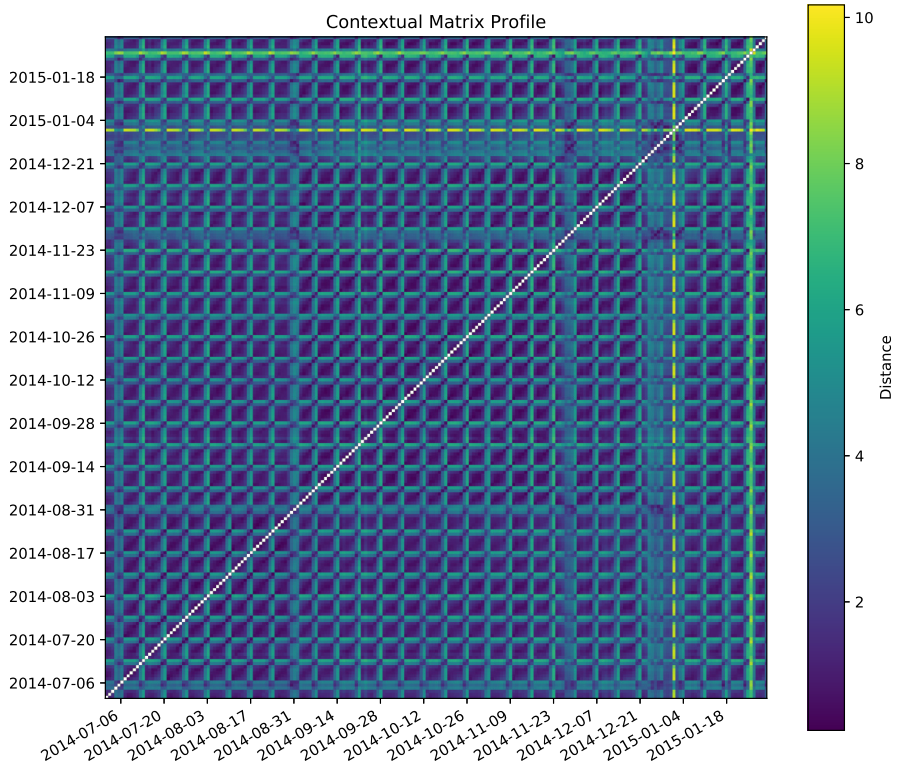


Figure 3.5: The CMP for the New York Taxi dataset. Each point displays the distance between 2 days, defined as the z-normalized Euclidean distance between the best matching 22 hour long subsequences of both days. Lower distances correspond to a better match. We can clearly see a periodic pattern caused by weekdays versus weekends and the changed behavior around Thanksgiving and between Christmas and New Year. The bright line near the end of January is the effect of a blizzard hitting New York.

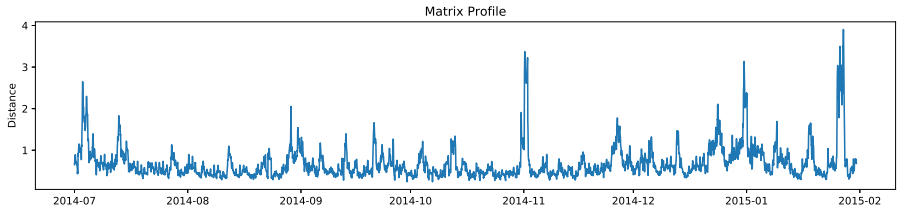


Figure 3.6: The Matrix Profile for the New York Taxi dataset. Each value represents the distance from the subsequence of the series starting at that index to its nearest match, where higher distances mean more unique subsequences. While we see higher values corresponding to some holidays or other events (discussed in Section 3.5.2), the periodic nature of the data is not captured in this visualization.

(Thanksgiving) and the stripe near the end of January (when a blizzard struck New York), all indicating different behavior in the dataset.

Visualizations like these help data scientists explore new datasets. By inspecting the CMP, they can find patterns and deviations from these patterns that might require further investigation (as we will do in our next use case). Another application is the creation of visual thumbnails for series, helping users to navigate large collections of series. Other thumbnail techniques have been presented using SAX [27] and time series snippets [23] but are unable to provide this degree of insight into the underlying patterns.

Of course, the Matrix Profile can also be visualized to gain insight in a series. We calculated the Matrix Profile using the same parameters as the CMP, it is shown in Figure 3.6. As mentioned before, the Matrix Profile is a one dimensional vector where high values correspond to more unique subsequences. Looking at the figure, we gain some insights in where the data displays unique behavior, which is further explored in Section 3.5.2. However, the Matrix Profile is unable to capture the periodic nature of the data since each sequence is compared against all other sequences rather than multiple spans like the CMP does.

As a final demonstration of the possibility to gain insights from visualizing the CMP, we would like to share an unexpected trivia we discovered. Looking carefully, one can see a small difference in the values before and after September 1st (Labor Day). This is more clearly presented in Figure 3.7 (left). We see the days before Labor Day have a worse match with the days after Labor Day and vice versa, indicating the taxi passenger behavior has changed. Indeed, when looking at the daily graphs (Figure 3.7 right), we see a noticeable difference in the behavior around 07:30 in the morning: after Labor Day, the number of taxi passengers is higher. The most likely explanation is the start of the school year, which also falls on September 1, enabling parents to leave earlier for work.

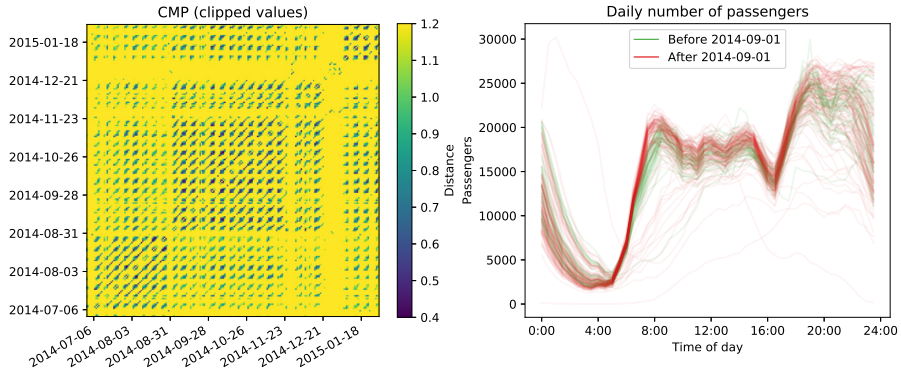


Figure 3.7: Left: The CMP for the New York Taxi dataset, with values restricted to the range [0.4, 1.2], highlighting the change in distance for days before and after September 1st. Right: The origin of the difference in distances. The number of taxi passengers before and after September 1st differs noticeably around 07:30 in the morning.

3.5.2 New York Taxi Dataset: Anomaly Detection

As anomalies are defined as *patterns that do not conform to expected behavior* [28], objectively evaluating them is particularly difficult for realistic datasets. What is interpreted as anomalous for one user, might be normal behavior for another [29]. While the New York Taxi dataset contains a ground truth of 5 anomalies (listed in Table 3.1) that were specified by the dataset provider as "anomalies with known causes"[2], we argue several deviations from expected patterns are present in the data but were not included in the ground truth because of background knowledge not present in the data. As a result, we find the ground truth to be biased towards techniques that find unique behavior, rather than unexpected behavior. Luckily, it is easy to further investigate and validate suspected anomalies, as we will do next.

The visualization of the CMP in Figure 3.5 already gives a good visual indication about anomalies: on some days the *expected* repetitive pattern is not present. Based on the visual pattern, we divided the contexts into three groups and form smaller CMPs: one containing weekdays and two containing only Saturdays and only Sundays respectively. This is visualised in Figure 3.8. These reduced CMPs each represent a collection of days that we expect to behave in a similar manner. Since each value in a column (or row) in the CMPs indicates how much a single day (context) deviates from other days (contexts), we can average each column to obtain a single value indicating how much this day deviates from the other days. We define this value as the anomaly score for that day. Note that we average the values in the reduced CMPs, meaning that, e.g. the anomaly score of any Sunday is based on how much it differs from all other Sundays

[2] https://github.com/numenta/NAB/wiki/FAQ

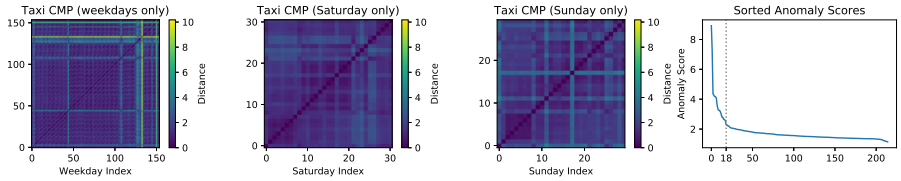


Figure 3.8: Reduced CMPs from Figure 3.5, containing only the entries for weekdays (first), Saturdays (second) or Sundays (third) on both axes. Fourth: The anomaly scores (obtained by averaging each column of all reduced CMPs), ordered from high to low. We determined the number of worthy anomalies to be 18.

in the dataset, irrespective of the differences with Saturdays or weekdays. After calculating the anomaly score for every day, we ordered all anomaly scores and using the Elbow method, we determined a threshold to obtain 18 anomalous days in total (Figure 3.8 right). The anomalies are listed in Table 3.1 and visualized in Figure 3.9.

We compare the anomalies against those found by the Matrix Profile. The Matrix Profile can be used to find series discords, subsequences that maximally differ from any other subsequence, these discords can be interpreted as anomalies [5]. We calculated the anomalies using the Matrix Profile with a window length of 22 hours (similar as the CMP) and not allowing overlapping anomalies. We obtained 16 anomalies using the Elbow method, which are listed in Table 3.1 and visualized in Figure 3.10. Note that the anomalies here have no starting time restriction and can partially cover one or two days.

Of the 25 different anomalies listed in Table 3.1, only nine are flagged as anomalous by both techniques. For each of these nine, a reasonable explanation could be found, falling into the categories of holiday (Independence Day, Thanksgiving, Martin Luther King Day), holiday predecessor (day before Christmas, New Year's Eve) or large scale event (Climate March, Daylight Savings Time and blizzard). The CMP additionally detected Labor Day, and many weekdays in the Christmas and New Years period, typical days when people take time off from work. Note that since the anomalies by the Matrix Profile can span two days, it would not be fair to consider Christmas and New Year to be found exclusively by the CMP. For one CMP anomaly no clear explanation could be found, though we suspect it is an after effect of the Independence Day celebrations. The Matrix Profile on the other hand exclusively found one weather event, one large scale event (the Millions March against police brutality), Halloween (most likely due to the effect of late-night parties) and four days for which no clear-cut explanation could be found. However, two of the unknown anomalies precede Labor Day, so this could again be an effect caused by people heading out of town for celebrations. Perhaps surprisingly, the Matrix Profile cannot detect Labor Day itself, this is because it closely matches Martin Luther King Day and two weekends in the dataset, meaning it will not be flagged as a series discord.

Rather than looking at individual anomalies, we can also look at the broader pic-

Date	Event	Numenta	MP	CMP
Thu 2014-07-03	Evening thunderstorms		5	
Fri 2014-07-04	Independence Day		6	5
Sun 2014-07-06	*Unknown*			15
Sun 2014-07-13	*Unknown*		10	
Fri 2014-08-29	*Unknown*		8	
Sun 2014-08-31	*Unknown*		15	
Mon 2014-09-01	Labor Day			6
Sun 2014-09-21	Climate March		13	17
Fri 2014-10-31	Halloween		9	
Sun 2014-11-02	Daylight Savings Time	x*	3*	9
Thu 2014-11-27	Thanksgiving	x	11*	12
Fri 2014-11-28	Day after Thanksgiving			11
Sat 2014-12-13	Millions March		16	
Wed 2014-12-24	Christmas period		7	3
Thu 2014-12-25	Christmas	x		7
Fri 2014-12-26	Christmas period			10
Mon 2014-12-29	New Year period			14
Tue 2014-12-30	New Year period			18
Wed 2014-12-31	New Year's Eve		4	16
Thu 2015-01-01	New Year	x		1
Fri 2015-01-02	New Year period			13
Fri 2015-01-09	*Unknown*		12	
Mon 2015-01-19	Martin Luther King Day		14*	8
Mon 2015-01-26	Blizzard		2	2
Tue 2015-01-27	Blizzard	x	1	4

Table 3.1: Anomalies as found by the Matrix Profile (MP) and CMP as well as the ground truth for the dataset (Numenta). The numbers in column CMP and MP correspond to the ordering used in Figure 3.9 and 3.10 respectively, where a lower number indicates a higher anomalous behavior.

*: Actually listed on the preceding day, but visual inspection shows the aberrant behavior takes place after midnight.

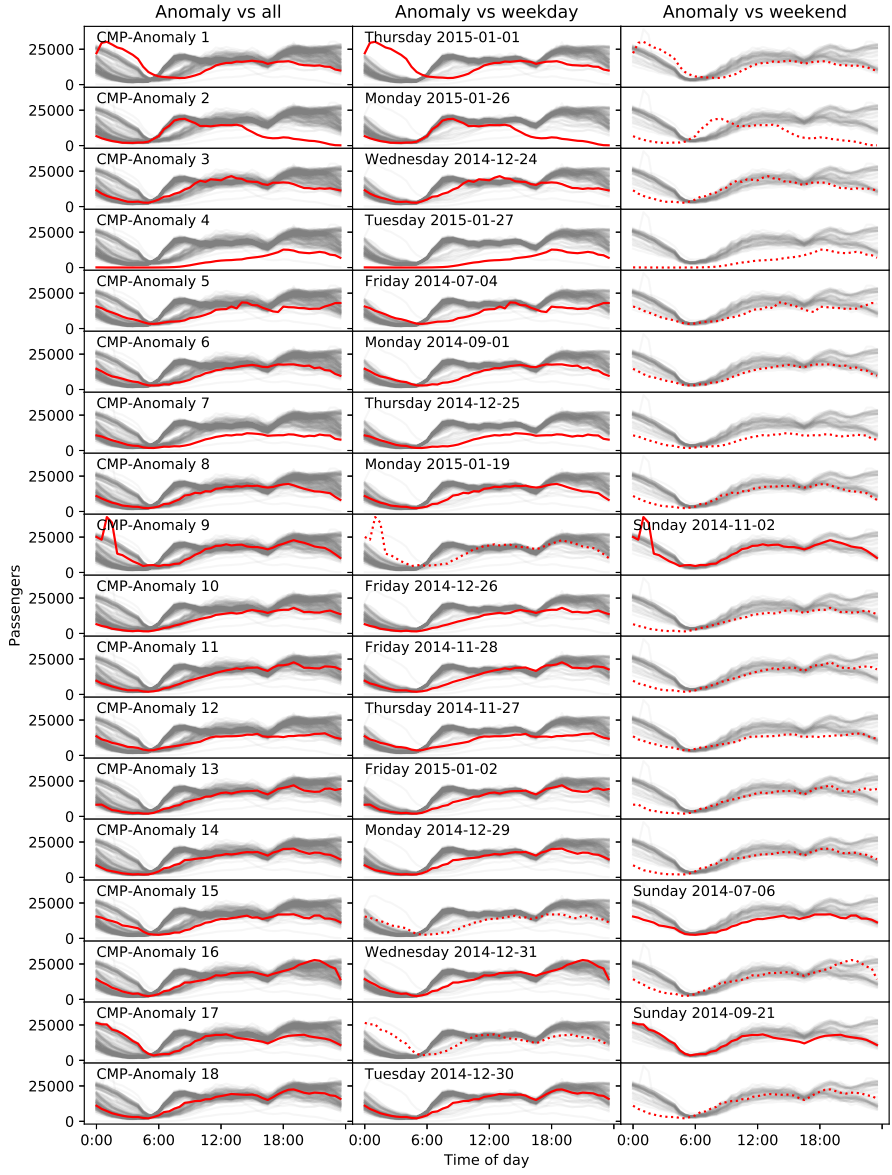


Figure 3.9: The 18 anomalous days found using the CMP, ordered from most anomalous to least anomalous. Each row shows one anomalous day (red) against all other days in the dataset (gray). A dotted red line is used to visualize the anomaly in the column that does not match its own type (weekday/weekend).

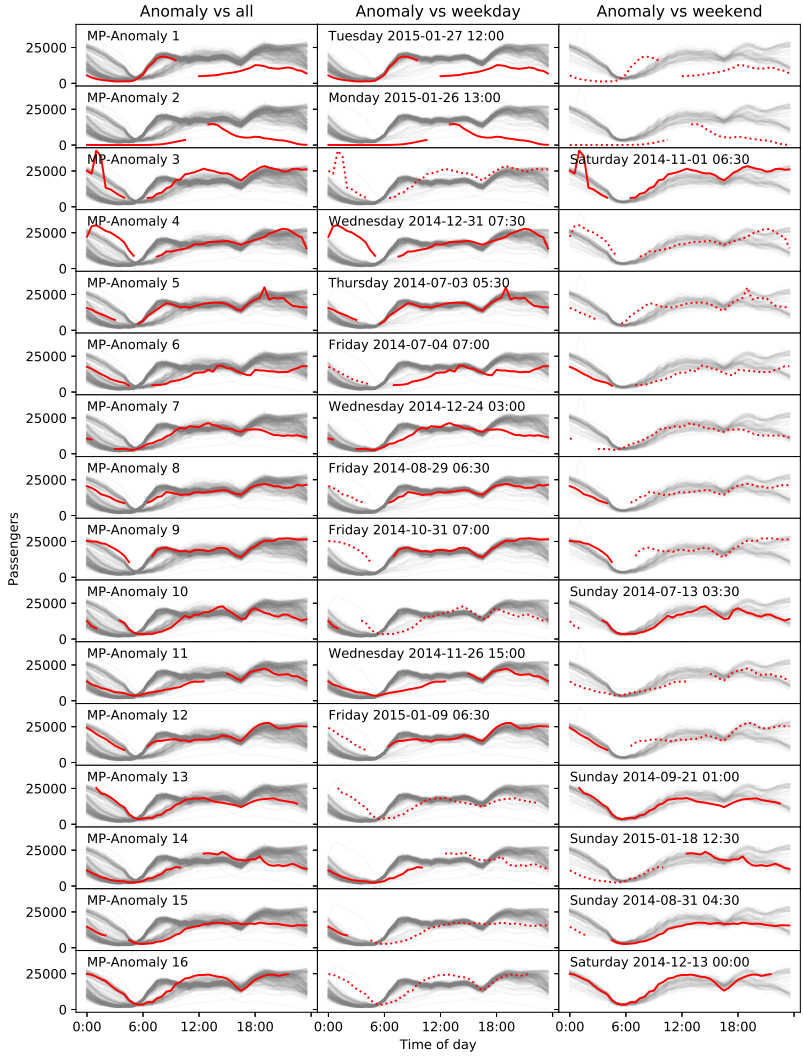


Figure 3.10: The 16 anomalous sequences found using the Matrix Profile, ordered from most anomalous to least anomalous. Each row shows one anomalous sequence of 22 hours (red) against all other days in the dataset (gray). A dotted red line is used to visualize the anomaly in the column that does not match its own type (weekday/weekend).

ture. By comparing each CMP anomaly against other days of the same type (the second or third column in Figure 3.9, whichever contains a solid red line), we see that all anomalous days noticeably differ from the majority of the reference days (gray band in the figure). This is less the case for the anomalies found by the Matrix Profile (Figure 3.10). Here, about half of the anomalies resemble the reference days, but contain some local variation such as a spike, elongated tail or less pronounced bumps.

The question arises: which of these techniques is best suited for anomaly detection? While we suspect most users will find the results of the CMP to be more insightful for this specific dataset, the general answer remains *"it depends"*. Fundamentally, both techniques are searching for different things. While the Matrix Profile is looking for the most unusual patterns (discords) in the series, the CMP based anomaly detection is looking for patterns that differ most from a group of reference contexts. Both approaches will have applications depending on the type of anomalies the user is interested in.

Whereas a simple distance matrix between weekdays and weekends could also have found these anomalies, this assumes knowing the underlying pattern in advance. One benefit of the CMP is that it allows us to discover these patterns in advance when the pattern is *unknown in advance*, which is often the case. So, assuming we did not know the weekday/weekend similarity beforehand, we could have easily deduced it by visualizing the CMP. The CMP has one other major advantage over a basic distance matrix, it allows for a (time) shift when comparing sequences (for which the added value is better demonstrated for the next dataset). A similar approach with typical techniques would result in a high complexity, instead we can rely on the computationally efficient implementations of the distance generators of the SDM framework [6, 16].

3.5.3 Ventilation Dataset: Data Visualization

Our second dataset is a proprietary dataset delivered to us by Renson, a ventilation manufacturing company. It contains measurements of various air quality metrics such as temperature, humidity, carbon dioxide and volative organic compounds, for all rooms within a building that are connected to a ventilation unit, for several anonymized buildings. The users of Renson ventilation products can use this data to observe the functioning of the ventilation system and to estimate the air quality of their home. The metrics are measured at 15 minute intervals and differ per room type. Here, we focus on the CO2 sensor of rooms designated as kitchen. The dataset is shown in Figure 3.11. Unlike the Taxi dataset, each household has a wide range of distinct daily behaviors and no immediate obvious repeating patterns, it is also not possible to verify any root causes of anomalies. This use case represents a typical use case wherein a data scientist has to explore data for which little to nothing is known.

We calculated the CMP using the z-normalized Euclidean distance, using a subsequence length of 3 hours and specifying contexts ranging from 06:00 until (including) 08:00 in the morning. The results are visualized in Figure 3.12. We see that all three units display very different morning behavior. The first unit displays a pattern that

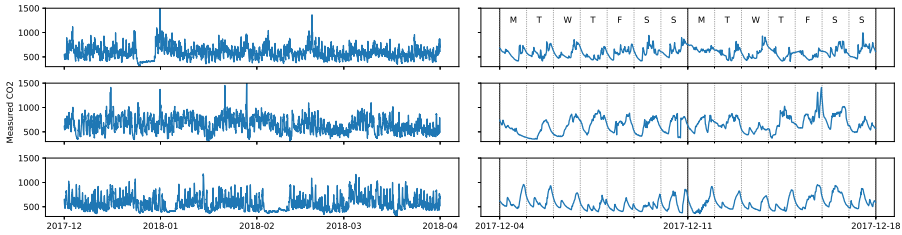


Figure 3.11: Measured CO2 air content in the kitchen for three ventilation units. Left: The complete datasets. Right: Closeup of two weeks for each corresponding dataset. A day/night pattern is somewhat discernible, but unlike the Taxi dataset, a weekday/weekend pattern is much less obvious.

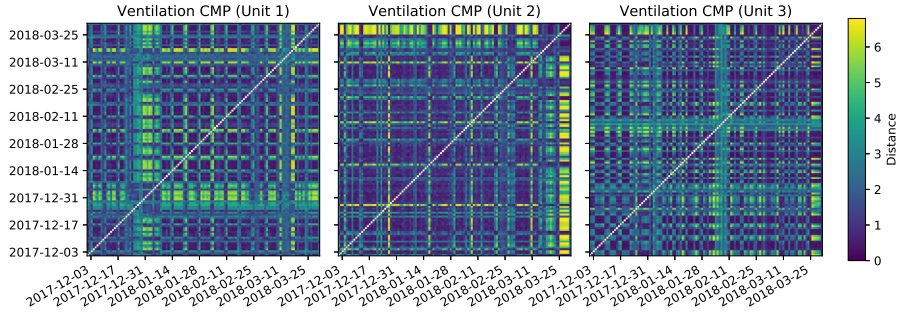


Figure 3.12: CMP calculated on the morning behavior of three kitchens. The first unit displays a weekday/weekend periodic pattern similar to the Taxi dataset, as well as different behavior around the holiday period. The second unit shows no clear pattern, indicating most mornings have a similar regime. The third unit shows a somewhat periodic pattern that does not match with weekdays/weekends.

closely resembles the Taxi dataset, with distinct behavior for weekdays, weekends and holidays. It most likely belongs to a family household with regular school and working hours. The second unit shows no clear patterns, though we can see a change near the end of the dataset. The last unit shows a pattern at the start of the dataset, which changes starting January. While we have no explanation for the behavior in these units, the patterns are still interesting to discover and could prove useful for experts. In parallel, we calculated other CMPs for noon and evening, but do not list them in this work due to size constraints and refer to the accompanying sources for more details [25].

3.5.4 Ventilation Dataset: Anomaly Detection

After exploring the data, we continue here with the dataset for the first unit. We choose this dataset as it shows most similarity to our expectations of a regular household and

should therefore be easier to interpret. Similar to the Taxi dataset, we split the CMP into contexts linked to weekdays and weekends. Since the weekday mornings are very similar, the results are quite similar to those of the Taxi dataset and we refer the reader to the supplementary material for more detailed results. Instead, we will focus on the more challenging weekend behavior in this section.

The weekend measurements do not only have a wider range of behavioral patterns, but the start time of these patterns also varies from day to day. Using the CMP calculated on the morning contexts from the previous section, we created a smaller CMP only containing weekend days. Unlike the Taxi dataset, we did not split up Saturdays and Sundays, since there was no distinctive pattern visible for these days in the CMP data visualization. Using the Elbow method, we determined the presence of six anomalies.

Due to the wide variation of the patterns in both values and time, it becomes harder to visualize the anomalies in an intuitive way. One useful approach is a matching table, of which an extract is shown in Figure 3.13 (the complete figure is available in the source files [25]). Every row of the table corresponds to a single weekend day (one row in the CMP). This day is shown in the first column with the morning context highlighted. The remaining columns show the matches with other weekend days, ordered from best match to worst match. Rather than showing all matches, we simply select the matches on all three quartiles, as well as the best and worst match. Note that each match corresponds to one single value listed in the CMP.

When inspecting the contents of the matching table, we see that the mornings classified as normal have many good matches, only showing minor differences in the third quartile match. The matches for the anomalous mornings already show this level of difference in the first quartile, showing that they are in fact uncommon behavior for a weekend morning. This is quantified in the distances listed in Figure 3.13: the distances of the first quartile match of anomalies are already higher than those of the third quartile of the normal days. Going further into detail, we see that the normal mornings share a common pattern of a plateau followed by a smooth bump and a second, higher plateau. We suspect this pattern is caused by someone waking up, having breakfast in the kitchen and going to an adjacent room. The mornings marked as anomalous show subtly different patterns. The first lacks the second plateau, the second has an earlier start (causing the first plateau to fall outside the context) and also lacks the higher plateau, the third anomaly lacks the distinct high bump at the start. Note that the second normal morning should probably be classified as anomalous. But even though the first spike occurs before the context, the z-normalisation enables a good match between the subtle second bump with the bumps of other days. This again demonstrates the need to finetune the anomaly detection algorithm to the needs of the user.

When looking at the matches in detail, we see how the blue subsequences are not exactly the same for each match. Indeed, the contexts used to produce the CMP allow a time shift: the three hour long subsequence should start between 06:00 and 08:00. As we can see, this flexibility allows us to recognize similar behavioral patterns, despite them not being aligned in time. This flexibility comes at the cost of the user having to

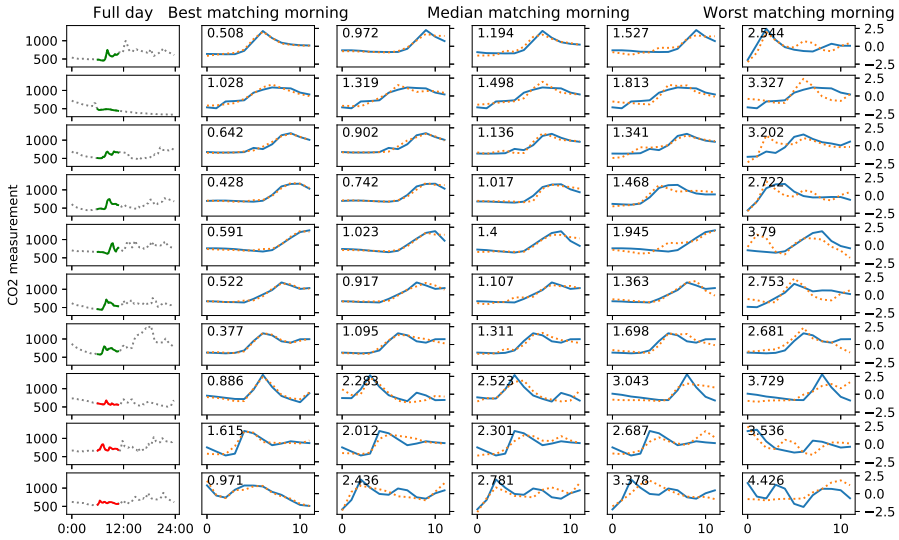


Figure 3.13: Matching table for subset of weekend days for ventilation unit 1. Each row corresponds to one weekend day, which is displayed in the first column with the morning context (including the window length) highlighted. The first seven rows display days classified as regular (green), the last three show anomalous days (red). The columns show the matching of the morning context (blue) with other morning contexts (dotted orange, one per column). Note that the matching uses subsequences of the context: each blue fragment is a three hour subsequence of the five hour long green/red fragment. For each match, the z-normalized Euclidean distance is displayed in the top left.

define the contexts, often having to rely on expert knowledge of the underlying process. In this case, we relied on our personal experience about kitchen usage patterns to define the contexts.

3.5.5 Summary

We conclude this section by reiterating our claim that anomaly detection is an inherent subjective topic and difficult to validate. Only when knowing what a user defines as anomalous, can the proper technique be chosen and tried. In this section, we defined normal behavior as behavior that closely matches the majority of the data, and found the CMP to be a suitable technique to detect outliers. We found 18 anomalies for the Taxi dataset, which is more than the five listed as ground truth, and could provide a straightforward explanation for all but one. In the ventilation dataset, we found six anomalies but had no way to validate them independent of the data.

One advantage of the CMP over the Matrix Profile for anomaly detection is that the CMP does not depend on the uniqueness of anomalies (it does not simply find

discords), but rather on the *the expectations of the user regarding normal behavior*. These expectations correspond to the CMP contexts and can be based on the insights retrieved using the CMP for data visualization. As part of the SDM framework, the CMP can be calculated using any distance measure and calculated in parallel with other techniques such as the Matrix Profile.

3.6 Conclusion

In this chapter we introduced the Series Distance Matrix framework (SDM), a generalisation of the original approach used to calculate the Matrix Profile. The SDM framework splits the generation and consumption of the all-pair subsequence distances, putting the focus on the distance matrix itself. This allows for easier and more flexible experiments by freely combining components and eliminates the need to re-implement algorithms to combine techniques in an efficient way. The extensions of the Matrix Profile can be fitted in this framework as (part of) a SDM-generator or SDM-consumer. Furthermore, we suspect new techniques will be discovered by further studying the properties of the distance matrix in future work.

We introduced one additional SDM-consumer, namely the Contextual Matrix Profile (CMP). The CMP processes rectangular areas of the distance matrix, compared to the Matrix Profile processing columns. As a result, the CMP is able to compare a range of subsequences against many other ranges, rather than only tracking the best match.

We proved the utility of the CMP for two use cases. When used for data visualization, the CMP was able to reveal repetitive and deviating patterns in the data, making it an ideal first step for data exploration, especially for data containing repetitive patterns. When used for anomaly detection, we defined contexts based on our expectations of the data and were able to find anomalies in the contexts not matching those expectations. Unlike the Matrix Profile, the CMP is able to detect anomalies that are not discords. Both cases were demonstrated on the New York Taxi dataset and a proprietary ventilation metric dataset. In the former, we were able to reasonably explain all patterns and anomalies. In the latter, we showed the visual difference between different ventilation units and relied on the time shift capability of the CMP to discover anomalous mornings.

As part of this publication, we have released a Python implementation of the SDM framework, already comprising implementations for a substantial set of related work. Furthermore, the source code for all use case related processing has been made available online [25].

References

[1] IoT Analytics. *State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating*. https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/. Aug. 2018.

[2] Eamonn Keogh, Li Wei, Xiaopeng Xi, Sang-hee Lee, and Michail Vlachos. "LB _ Keogh Supports Exact Indexing of Shapes under Rotation Invariance with Arbitrary Representations and Distance Measures". In: *Proc. of the 32nd Int. Conf. on Very Large Data Bases*. Seoul, Korea: VLDB Endowment, 2006, pp. 882–893. ISBN: 1595933859. URL: http://dl.acm.org/citation.cfm?id=1182635.1164203.

[3] Yaguo Lei, Naipeng Li, Liang Guo, Ningbo Li, Tao Yan, and Jing Lin. "Machinery health prognostics: A systematic review from data acquisition to RUL prediction". In: *Mechanical Systems and Signal Processing* 104 (2018), pp. 799–834. ISSN: 10961216. DOI: 10.1016/j.ymssp.2017.11.016. URL: https://doi.org/10.1016/j.ymssp.2017.11.016.

[4] D Vries, B. van den Akker, E. Vonk, W. de Jong, and J. van Summeren. "Application of machine learning techniques to predict anomalies in water supply networks". In: *Water Science and Technology: Water Supply* 16.6 (Dec. 2016), pp. 1528–1535. ISSN: 1606-9749. DOI: 10.2166/ws.2016.062. URL: http://ws.iwaponline.com/cgi/doi/10.2166/ws.2016.062.

[5] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. "Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets". In: *2016 IEEE 16th Int. Conf. on Data Mining (ICDM)*. IEEE, Dec. 2016, pp. 1317–1322. ISBN: 978-1-5090-5473-2. DOI: 10.1109/ICDM.2016.0179. URL: http://ieeexplore.ieee.org/document/7837992/.

[6] Yan Zhu, Zachary Zimmerman, Nader Shakibay Senobari, Chin-Chia Michael Yeh, Gareth Funning, Philip Brisk, and Eamonn Keogh. "Matrix Profile II : Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins". In: *2016 IEEE 16th Int. Conf. on Data Mining (ICDM)* (2016), pp. 739–748. DOI: 10.1109/ICDM.2016.126.

[7] Chin-Chia Michael Yeh, Helga Van Herle, and Eamonn Keogh. "Matrix profile III: The matrix profile allows visualization of salient subsequences in massive time series". In: *Proceedings - IEEE Int. Conf. on Data Mining, ICDM* (2017), pp. 579–588. ISSN: 2374-8486. DOI: 10.1109/ICDM.2016.0069.

[8] Chin-Chia Michael Yeh, Nickolas Kavantzas, and Eamonn Keogh. "Matrix profile IV: Using Weakly Labeled Time Series to Predict Outcomes". In: *Proc. of the VLDB Endowment* 10.12 (Aug. 2017), pp. 1802–1812. ISSN: 21508097. DOI: 10.14778/3137765.3137784. URL: http://www.vldb.org/pvldb/vol10/p1802-yeh.pdf%20http://dl.acm.org/citation.cfm?doid=3137765.3137784.

[9] Hoang Anh Dau and Eamonn Keogh. "Matrix Profile V: A Generic Technique to Incorporate Domain Knowledge into Motif Discovery". In: *Proc. of the 23rd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining - KDD '17*. New York, New York, USA: ACM Press, 2017, pp. 125–134. ISBN: 9781450348874. DOI: 10.1145/3097983.3097993. URL: http://dl.acm.org/citation.cfm?doid=3097983.3097993.

[10] Chin-Chia Michael Yeh, Nickolas Kavantzas, and Eamonn Keogh. "Matrix Profile VI: Meaningful Multidimensional Motif Discovery". In: *2017 IEEE Int. Conf. on Data Mining (ICDM)*. IEEE, Nov. 2017, pp. 565–574. ISBN: 978-1-5386-3835-4. DOI: 10.1109/ICDM.2017.66. URL: http://ieeexplore.ieee.org/document/8215529/.

[11] Yan Zhu, Makoto Imamura, Daniel Nikovski, and Eamonn Keogh. "Matrix Profile VII: Time Series Chains: A New Primitive for Time Series Data Mining". In: *2017 IEEE Int. Conf. on Data Mining (ICDM)*. IEEE, Nov. 2017, pp. 695–704. ISBN: 978-1-5386-3835-4. DOI: 10.1109/ICDM.2017.79. URL: http://ieeexplore.ieee.org/document/8215542/.

[12] Shaghayegh Gharghabi, Yifei Ding, Chin-Chia Michael Yeh, Kaveh Kamgar, Liudmila Ulanova, and Eamonn Keogh. "Matrix Profile VIII: Domain Agnostic Online Semantic Segmentation at Superhuman Performance Levels". In: *2017 IEEE Int. Conf. on Data Mining (ICDM)*. IEEE, Nov. 2017, pp. 117–126. ISBN: 978-1-5386-3835-4. DOI: 10.1109/ICDM.2017.21. URL: http://ieeexplore.ieee.org/document/8215484/.

[13] Reza Akbarinia and Bertrand Cloez. "Efficient Matrix Profile Computation Using Different Distance Functions". 2019. URL: https://arxiv.org/abs/1901.05708.

[14] Diego Furtado Silva and Gustavo E. A. P. A. Batista. "Elastic Time Series Motifs and Discords". In: *2018 17th IEEE Int. Conf. on Machine Learning and Applications (ICMLA)*. IEEE, Dec. 2018, pp. 237–242. ISBN: 978-1-5386-6805-4. DOI: 10.1109/ICMLA.2018.00042. URL: https://ieeexplore.ieee.org/document/8614067/.

[15] Dieter De Paepe, Olivier Janssens, and Sofie Van Hoecke. "Eliminating Noise in the Matrix Profile". In: *Proceedings of the 8th Int. Conf. on Pattern Recognition Applications and Methods*. SCITEPRESS - Science and Technology Publications, 2019, pp. 83–93. ISBN: 978-989-758-351-3. DOI: 10.

5220 / 0007314100830093. URL: http : / / www . scitepress . org / DigitalLibrary/Link.aspx?doi=10.5220/0007314100830093.

[16] Yan Zhu, Chin-Chia Michael Yeh, Zachary Zimmerman, Kaveh Kamgar, and Eamonn Keogh. "Matrix Profile XI: SCRIMP++: Time Series Motif Discovery at Interactive Speeds". In: *2018 IEEE Int. Conf. on Data Mining (ICDM)*. IEEE, Nov. 2018, pp. 837–846. ISBN: 978-1-5386-9159-5. DOI: 10 . 1109 / ICDM . 2018 . 00099. URL: https : //ieeexplore . ieee . org/document / 8594908/.

[17] Michele Linardi, Yan Zhu, Themis Palpanas, and Eamonn Keogh. "Matrix Profile X". In: *Proc. of the 2018 Int. Conf. on Management of Data - SIGMOD '18*. ACM Press, 2018, pp. 1053–1066. ISBN: 9781450347037. DOI: 10.1145/3183713 . 3183744. URL: http : //dl . acm . org/citation . cfm?doid= 3183713.3183744.

[18] Eamonn Keogh and Shruti Kasetty. "On the need for time series data mining benchmarks". In: *Proc. of the 8th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining - KDD '02* (2002), p. 102. ISSN: 13845810. DOI: 10 . 1145/775047 . 775062. URL: http://portal . acm . org/citation . cfm? doid=775047.775062.

[19] Abdullah Mueen, Yan Zhu, Michael Yeh, Kaveh Kamgar, Krishnamurthy Viswanathan, Chetan Gupta, and Eamonn Keogh. *The Fastest Similarity Search Algorithm for Time Series Subsequences under Euclidean Distance*. http:// www.cs.unm.edu/~mueen/FastestSimilaritySearch.html. Aug. 2017.

[20] Dieter De Paepe, Diego Nieves Avendano, and Sofie Van Hoecke. "Implications of Z-Normalization in the Matrix Profile". In: *Pattern Recognition Applications and Methods*. Ed. by Maria De Marsico, Gabriella Sanniti di Baja, and Ana Fred. Cham: Springer International Publishing, 2020, pp. 95–118. ISBN: 978-3-030-40014-9. DOI: https://doi.org/10.1007/978-3-030-40014-9_5.

[21] Yan Zhu, Abdullah Mueen, and Eamonn Keogh. "Admissible Time Series Motif Discovery with Missing Data". In: (2018). arXiv: 1802.05472. URL: %5Curl% 7Bhttps://arxiv.org/pdf/1802.05472.pdf%7D.

[22] Diego F Silva, Chin-chia M Yeh, Yan Zhu, Gustavo E A P A Batista, and Eamonn Keogh. "Fast Similarity Matrix Profile for Music Analysis and Exploration". In: *IEEE Transactions on Multimedia* 21.1 (Jan. 2019), pp. 29–38. ISSN: 1520-9210. DOI: 10 . 1109 / TMM . 2018 . 2849563. URL: https : // ieeexplore.ieee.org/document/8392419/.

[23] Shima Imani, Frank Madrid, Wei Ding, Scott Crouter, and Eamonn Keogh. "Matrix Profile XIII: Time Series Snippets: A New Primitive for Time Series Data Mining". In: *2018 IEEE Int. Conf. on Big Knowledge (ICBK)*. IEEE, Nov. 2018, pp. 382–389. ISBN: 978-1-5386-9125-0. DOI: 10 . 1109 / ICBK . 2018 . 00058. URL: https://ieeexplore.ieee.org/document/8588817/.

[24] Shaghayegh Gharghabi, Shima Imani, Anthony Bagnall, Amirali Darvishzadeh, and Eamonn Keogh. "Matrix Profile XII: MPdist: A Novel Time Series Distance Measure to Allow Data Mining in More Challenging Scenarios". In: *2018 IEEE Int. Conf. on Data Mining (ICDM)*. IEEE, Nov. 2018, pp. 965–970. ISBN: 978-1-5386-9159-5. DOI: 10.1109/ICDM.2018.00119. URL: https://ieeexplore.ieee.org/document/8594928/.

[25] *Source code for our experiments*. https://sites.google.com/view/generalizing-matrix-profile. 2019.

[26] Alexander Lavin and Subutai Ahmad. "Evaluating Real-Time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark". In: *2015 IEEE 14th Int. Conf. on Machine Learning and Applications*. IEEE, Dec. 2015, pp. 38–44. ISBN: 978-1-5090-0287-0. DOI: 10.1109/ICMLA.2015.141. arXiv: 1510.03336. URL: http://ieeexplore.ieee.org/document/7424283/.

[27] Nitin Kumar, Venkata Nishanth Lolla, Eamonn Keogh, Stefano Lonardi, Chotirat Ann Ratanamahatana, and Li Wei. "Time-series Bitmaps: a Practical Visualization Tool for Working with Large Time Series Databases". In: *Proc. of the 2005 SIAM Int. Conf. on Data Mining*. Society for Industrial and Applied Mathematics, Apr. 2005, pp. 531–535. ISBN: 9781491917022. DOI: 10.1137/1.9781611972757.55. URL: http://epubs.siam.org/doi/abs/10.1137/1.9781611972757.55.

[28] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.

[29] Haemwaan Sivaraks and Chotirat Ann Ratanamahatana. "Robust and accurate anomaly detection in ECG artifacts using time series motif discovery". In: *Computational and Mathematical Methods in Medicine* 2015 (2015). ISSN: 17486718. DOI: 10.1155/2015/453214.

Chapter 4

Implications of Z-Normalization in the Matrix Profile

Many data analytics methods for time series deal in some way with the concept of similarity between subsequences, and consequently, with distance measures that determine this similarity. Chapter 1 gave an overview of several commonly used measures, each having distinctive properties. The z-normalized Euclidean distance, which was the original distance measure used for the matrix profile, is a measure that mainly captures similarities in the pattern (or shape) of subsequences. It has been applied to a wide variety of use cases, and is intuitive for users due to the visual similarity of sequences with distinct patterns. However, when sequences lack these distinct patterns, similarity becomes counter-intuitive because the considered shape originates from the noise present in the signal. This in turn negatively affects all data analytics techniques where the analyzed signal contains parts lacking distinct patterns. This chapter discusses a method to restore the applicability of these techniques for such cases. Furthermore, it studies other properties of this measure as well, such as the close link with the Pearson correlation coefficient.

My contributions can be summarized as follows:

1. Discussion of known properties of the Pearson correlation and z-normalized Euclidean distance measure in the context of pattern matching. One direct application is a method to normalize distances, irrespective of sequence length.

2. Analytical derivation of a method to estimate the effect of noise on the z-normalized Euclidean distance, and providing a method to correct for this effect.

3. Demonstration of the benefits for time series anomaly detection, segmentation and visualization.

Implications of Z-Normalization in the Matrix Profile

D. De Paepe, D. Nieves, and S. Van Hoecke

Published in "Pattern Recognition Applications and Methods"

Abstract Companies are increasingly measuring their products and services, resulting in a rising amount of available time series data, making techniques to extract usable information needed. One state-of-the-art technique for time series is the Matrix Profile, which has been used for various applications including motif/discord discovery, visualizations and semantic segmentation. Internally, the Matrix Profile utilizes the z-normalized Euclidean distance to compare the shape of subsequences between two series. However, when comparing subsequences that are relatively flat and contain noise, the resulting distance is high despite the visual similarity of these subsequences. This property violates some of the assumptions made by Matrix Profile based techniques, resulting in worse performance when series contain flat and noisy subsequences. By studying the properties of the z-normalized Euclidean distance, we derived a method to eliminate this effect requiring only an estimate of the standard deviation of the noise. In this chapter we describe various practical properties of the z-normalized Euclidean distance and show how these can be used to correct the performance of Matrix Profile related techniques. We demonstrate our techniques using anomaly detection using a Yahoo! Webscope anomaly dataset, semantic segmentation on the PAMAP2 activity dataset and for data visualization on a UCI activity dataset, all containing real-world data, and obtain overall better results after applying our technique. Our technique is a straightforward extension of the distance calculation in the Matrix Profile and will benefit any derived technique dealing with time series containing flat and noisy subsequences.

4.1 Introduction

With the lower cost of sensors and according rise of IoT and Industrial IoT, the amount of data available as time series is rapidly increasing due to rising interest of companies to gain new insights about their products or services, for example to do pattern discovery [1], user load prediction or anomaly detection [2].

The Matrix profile is state-of-the-art technique for time series data that is calculated using two time series and a provided subsequence length. It is a one-dimensional series where each data point at a given index represents the Euclidean distance between the z-normalized (zero mean and unit variance) subsequence starting at that index in the first time series and the best matching (lowest distance) z-normalized subsequence in the second time series. Both inputs can be the same, meaning matches are searched for in the same time series. The Matrix Profile Index, which is calculated alongside the Matrix Profile, contains the location of the best match (in the second series) for each subsequence.

The Matrix Profile can be used to find the best matching subsequence in a series, i.e. motif discovery, or to find the subsequence with the largest distance to its nearest match, i.e. discord discovery. It also serves as a building block for other techniques such as segmentation [3], visualizing time series using Multidimensional Scaling [4] or finding gradually changing patterns in time series [5].

The usage of the z-normalized Euclidean distance can be explained by two factors. First, the MASS algorithm [6] was a known method to calculate the z-normalized distance between a sequence of length m and all subsequences obtained by sliding a window of length m over a longer sequence of length n. MASS was a vital part of the original method to calculate the Matrix Profile in reasonable time. Secondly, the z-normalized Euclidean distance can be seen as a two-step process to compare the *shape* of two sequences: the z-normalization transforms each sequence to their normal form, which captures their shape, after which the Euclidean distance compares both shapes. This makes the Matrix Profile well suited for finding patterns in data where a wandering baseline is present, as often occurs in signals coming from natural sources or due to uncalibrated sensors, or where patterns manifest with different amplitudes, which can occur by subtle changes in the underlying system or when comparing signals from different sources.

Although z-normalization is important when comparing time series [7], it has one major downside: when dealing with flat sequences, any fluctuations (such as noise) are enhanced, resulting in high values in the Matrix Profile. This behavior conflicts with our human intuition of similarity and can have an adverse effect on techniques based on the Matrix Profile. A preliminary example of this can be seen in Figure 4.2, where a discord that is easily detectable using the Matrix Profile becomes hidden once noise is added to the signal. Previous literature has mainly avoided cases with series containing flat and noisy regions, most likely due to this effect.

This chapter is an extended version of our previous work [8]. In this version, we elaborate less on the many merits of the Matrix Profile and instead discuss several properties of the z-normalized distance relevant for the Matrix Profile. Furthermore, we have used a new dataset from Yahoo! Webscope in our anomaly detection use case and introduced a new visualization use case.

This chapter is structured as follows: Section 4.2 lists literature related to the Matrix Profile. Section 4.3 discusses several properties of the z-normalized Euclidean distance that are either directly relevant when using the Matrix Profile or are used in our main contribution. Section 4.4 provides detail on the effect of flat, noisy subsequences in the Matrix Profile, as well as our solution to compensate for this effect. We demonstrate our technique for anomaly detection in Section 4.5, for semantic segmentation in Section 4.6, on data visualization in Section 4.7 and conclude our work in Section 4.8.

4.2 Related Work

In this section, we focus on works related to the Matrix Profile, introduced by Yeh et al. [9] as a new time-series analysis building block, together with the STAMP and STAMPI algorithm to calculate the Matrix Profile in batch or incremental steps respectively.

Internally, STAMP uses the z-normalized Euclidean distance metric to compare subsequences. Originally, all subsequences were compared using the MASS algorithm [6] allowing the Matrix Profile to be calculated in $O(n^2 \log n)$, with n being the length of the series. The later introduced STOMP and SCRIMP algorithms [10, 11] reduced the runtime to $O(n^2)$ for both batch and incremental calculation respectively by applying dynamic programming techniques.

Various variations or enhancements of the Matrix Profile have been published. When users want to track the best earlier and later match of each subsequence, rather than the best global match, the left and right Matrix Profile can be calculated instead [5]. The Multidimensional Matrix Profile tracks the best matches between time-series containing multiple channels [12]. Zhu et al. have suggested a way to calculate the Matrix Profile when the data contains missing values, using knowledge about the range of the data [13]. Lastly, we presented the Contextual Matrix Profile [14] as a generalization of the Matrix Profile that is capable of tracking multiple matches over configurable time spans.

Different distance measures have also been proposed for the Matrix Profile. The Euclidean distance or more general p-norm, might be useful in areas such as finances, engineering, physics or statistics [15]. A distance measure that performs a non-linear transformation along the time axis and can ignore the prefix or suffix of sequences being matched, based on Dynamic Time Warping, has been suggested by Furtado Silva et al. [16]. Recently, we suggested the Series Distance Matrix framework [14] as a way to easily combine different distance measures with the techniques processing these distances in a plug-and-play way.

Once the Matrix Profile and corresponding Matrix Profile index have been calculated, they can be used for motif or discord discovery. In case the user wants to focus on specific parts of the signal, for example based on time regions or high-variance periods in the signal, they can shift the Matrix Profile using the Annotation Vector [17], allowing them to find different sets of discords or motifs.

The Matrix Profile can also be used as a building block for other techniques. Time Series Chains are slowly changing patterns that occur throughout a time series and can be found by analyzing the left and right Matrix Profile [5]. Time series segmentation involves detecting changes in the underlying behavior of a time series and is possible using the offline FLUSS or online FLOSS algorithm [3], which investigate the number of arcs defined by the Matrix Profile index to detect likely transitions. Classification of time series is possible through a dictionary of identifying patterns discovered through the Matrix Profile [18]. The Matrix Profile has also been shown useful for MDS, a data exploration technique that does not work well when visualizing all subsequences in a

series, by selecting representative subsequences of series [4]. Lastly, MPDist [19], a distance measure that treats sequences similar if they share many similar subsequences, is calculated using the Matrix Profile and has been used to summarize large datasets for visualisation and exploration [20].

The Matrix Profile has been used in various techniques across many domains. However, series where flat and noisy regions are present have been mostly avoided in related literature, most likely due to the issue mentioned in Section 4.1. We suspect this issue affects any Matrix Profile based technique using the z-normalized Euclidean distance, and will especially have a negative impact on techniques dealing with discord discovery (such as anomaly detection) or techniques involving matches made on flat sequences (such as the assumption of motifs being present in homogeneous regions when performing time series segmentation). To the best of our knowledge, this issue has not yet been discussed or solved prior to our work. We show how to solve this issue in Section 4.4, after first discussing several relevant properties of the z-normalized distance measure in Section 4.3.

4.3 Properties of the Z-normalized Euclidean Distance

This section gathers aspects of the z-normalized Euclidean distance that are relevant for the remainder of this chapter or when working with the Matrix Profile in general. Some properties listed here are obtainable through straightforward mathematical derivation of previously published properties, but have not yet been mentioned in Matrix Profile related literature, despite their high relevance.

4.3.1 Definition

The z-normalized Euclidean distance D_{ze} is defined as the Euclidean distance D_e between the *z-normalized* or *normal form* of two sequences, where the z-normalized form $\hat{X}$ is obtained by transforming a sequence X of length m so it has mean $\mu = 0$ and standard deviation $\sigma = 1$.

$$\hat{X} = \frac{X - \mu_X}{\sigma_X}$$

$$D_{ze}(X, Y) = D_e(\hat{X}, \hat{Y}) = \sqrt{(\hat{x}_1 - \hat{y}_1)^2 + \ldots + (\hat{x}_m - \hat{y}_m)^2}$$

4.3.2 Link with Pearson Correlation Coefficient

The z-normalized Euclidean distance between two sequences of length m is in fact a function of the correlation between the two sequences, as originally mentioned by Rafiei D. [21], though without the derivation we provide below.

$$D_{ze}(X, Y) = \sqrt{2m(1 - corr(X, Y))}$$

To derive this property, we first highlight the following property of the inner product of a z-normalized sequence with itself:

$$\sigma_X^2 = \frac{\sum_i^m (x_i - \mu_X)^2}{m}$$
$$m = \sum_i^m \left(\frac{x_i - \mu_X}{\sigma_X} \right)^2$$

Using this, we can derive the equality as follows:

$$\begin{aligned} D_{ze}(X,Y)^2 &= \sum_i^m \left(\frac{x_i - \mu_X}{\sigma_X} - \frac{y_i - \mu_Y}{\sigma_Y} \right)^2 \\ &= \sum_i^m \left(\frac{x - \mu_X}{\sigma_X} \right)^2 + \sum_i^m \left(\frac{y - \mu_Y}{\sigma_Y} \right)^2 - 2 \sum_i^m \left(\frac{x - \mu_X}{\sigma_X} \right) \left(\frac{y - \mu_Y}{\sigma_Y} \right) \\ &= 2m \left(1 - \frac{1}{m} \sum_i^m \left(\frac{x - \mu_X}{\sigma_X} \right) \left(\frac{y - \mu_Y}{\sigma_Y} \right) \right) \\ &= 2m(1 - corr(X,Y)) \end{aligned}$$

4.3.3 Distance Bounds

Since the correlation is limited to the range $[-1, 1]$, the D_{ze} between two sequences of length m will fall in the range $[0, 2\sqrt{m}]$, where zero indicates a perfect match and $2\sqrt{m}$ corresponds to the worst possible match.

As a result, the upper bound of $2\sqrt{m}$ can be used to *normalize distances* to the range $[0, 1]$, allowing us to compare matches of different lengths and enabling us to define and reuse thresholds to define degrees of similarity when using D_{ze}. This way, we can define a more uniform similarity threshold (e.g.: 0.3) for sequences of any length rather than specifying a threshold that is dependent on m (e.g.: a threshold of 3 for sequences of length 25, 6 for sequences of length 100 and so on). Note that Linardi et al. [22] had already pragmatically found the normalization factor $\sqrt{m}$ to compare matches of different lengths, though without making the connection to the underlying mathematics.

4.3.4 Best and Worst Matches

The distance bounds of D_{ze} of 0 and $2\sqrt{m}$ correspond to correlation coefficients of 1 and -1 respectively. This means that for any sequence X of length m with $\sigma_X \neq 0$, $D_{ze}(X, Y) = 0$ and $D_{ze}(X, Z) = 2\sqrt{m}$ if:

$$Y = aX + b$$
$$Z = -aX + b$$

for any values of a and b, where $a > 0$.

4.3.5 Effects of Noise on Self-Similarity

If we have a base sequence $S \in \mathbb{R}^m$ and two noise sequences $N \in \mathbb{R}^m$ and $N' \in \mathbb{R}^m$ sampled from a normal distribution $\mathcal{N}\left(0,\, \sigma_N^2\right)$, then the expected distance between the two sequences obtained by adding the noise to the base sequence can be expressed as follows:

$$X = S + N$$
$$Y = S + N'$$

$$\mathbb{E}\left[D_{ze}\left(X, Y\right)^2\right] = (2m + 2)\,\frac{\sigma_N^2}{\sigma_S^2 + \sigma_N^2} \tag{4.1}$$

Note that in (4.1), σ_N^2 is the variance of the noise and $\sigma_S^2 + \sigma_N^2$ is the expected variance of either noisy sequence. We apply the derivation below, originally published in our previous work [8]. For the remainder of this section, we treat the sequences as random variables.

$$\begin{aligned}
\mathbb{E}\left[D_{ze}(X, Y)^2\right] &= \mathbb{E}\left[(\hat{x}_1 - \hat{y}_1)^2 + \ldots + (\hat{x}_m - \hat{y}_m)^2\right] \\
&= m \cdot \mathbb{E}\left[(\hat{x} - \hat{y})^2\right] \\
&= m \cdot \mathbb{E}\left[\left(\frac{x - \mu_X}{\sigma_X} - \frac{y - \mu_Y}{\sigma_Y}\right)^2\right]
\end{aligned} \tag{4.2}$$

Since X and Y are the sum of the same two uncorrelated variables, they both have the same variance.

$$\sigma_X^2 = \sigma_Y^2 = \sigma_S^2 + \sigma_N^2 \tag{4.3}$$

Next, we decompose μ_X and μ_Y in the component from the original sequence μ_S and the influence of the noise. Here we use n as a random variable sampled from the noise distribution. Note that μ_S can be seen as a constant as it refers to the mean of the base sequence.

$$\begin{aligned}
\mu_X = \mu_Y &= \mu_S + \frac{n_1 + \ldots + n_m}{m} \\
&= \mu_S + \mu_N \\
\mu_N &\sim \mathcal{N}\left(0,\, \frac{\sigma_N^2}{m}\right)
\end{aligned} \tag{4.4}$$

We perform the same decomposition for x and y, where s is an unknown constant originating from the base sequence:

$$\begin{aligned} x = y &= s + n \\ n &\sim \mathcal{N}\left(0,\, \sigma_N^2\right) \end{aligned} \tag{4.5}$$

Using (4.3), (4.4) and (4.5) in (4.2), canceling out constant terms and merging the distributions results in:

$$\begin{aligned} \mathbb{E}\left[D_{ze}(X,Y)^2\right] &= m \cdot \mathbb{E}\left[\left(\frac{n_x - n_y - \mu_{N_x} + \mu_{N_y}}{\sqrt{\sigma_S^2 + \sigma_N^2}}\right)^2\right] \\ &= m \cdot \mathbb{E}\left[(\nu)^2\right] \\ \nu &\sim \mathcal{N}\left(0,\, \frac{2+2m}{m} \cdot \frac{\sigma_N^2}{\sigma_S^2 + \sigma_N^2}\right) \end{aligned} \tag{4.6}$$

To finish, we apply the theorem $\mathbb{E}[X^2] = \text{var}(X) + \mathbb{E}[X]^2$:

$$\mathbb{E}\left[D_{ze}(X,Y)^2\right] = (2m+2) \cdot \frac{\sigma_N^2}{\sigma_S^2 + \sigma_N^2} \tag{4.7}$$

4.4 Flat Subsequences in the Matrix Profile

While the utility of the z-normalized Euclidean distance as a shape-comparator has been proven by the many Matrix Profile related publications [4, 5, 22], results become counter-intuitive for sequences that contain subsequences that are flat, with a small amount of noise. While humans would consider such sequences as similar, the z-normalized Euclidean distance will be very high.

We can explain this effect in two ways and demonstrate this in Figure 4.1, where we visualize three pairs of noisy sequences that only differ by their slope. First, considering the Euclidean distance on z-normalized sequences, we can see in Figure 4.1 (middle) how the effect of noise becomes more outspoken for flatter sequences due to the normalization, resulting in a high Euclidean distance. Alternatively, we can consider the correlation of both sequences, as mentioned in Section 4.3.2. Looking at both sequences as a collection of points, shown in Figure 4.1 (bottom), we can see that flatter sequences more closely resemble the random distribution of the underlying noise and are therefor less correlated. Since a correlation of zero corresponds to a z-normalized Euclidean distance of $\sqrt{2m}$, or $\frac{1}{\sqrt{2}} \approx 0.707$ if we rescale this value using the distance bounds mentioned in Section 4.3.3, we can see that uncorrelated sequences will have a high distance.

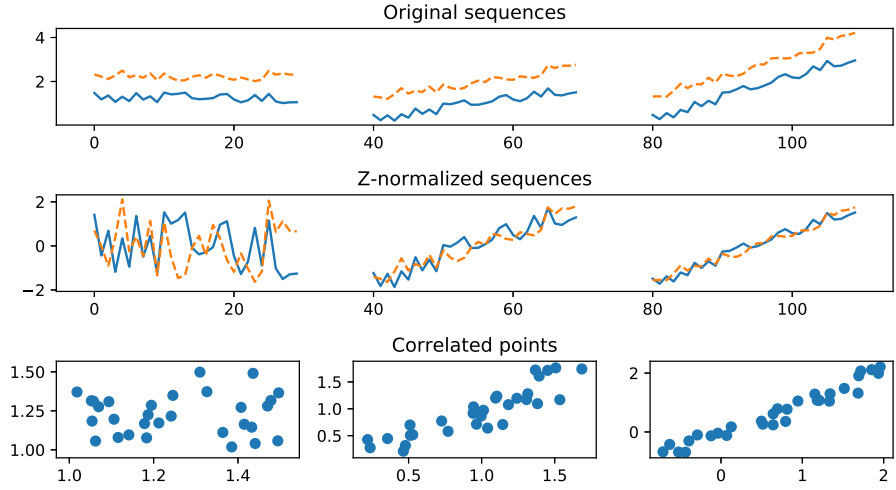


Figure 4.1: Three pairs of sequences with varying slopes, each pair has the same noise profile. By looking at the effect of the noise in the z-normalized sequences, we see why the Euclidean distance will return much larger distances for flat sequences. At the bottom we see a visualization of the correlation between both sequences, where we see that the slope of the signal has a major influence on the corresponding correlation.

The effect of flat, noisy subsequences will have a negative effect on some use cases of the Matrix Profile. Since the flat sequences result in high Matrix Profile values where we would intuitively expect low values, we can estimate which use cases will suffer and which will not. For example, anomaly detection or discord discovery using the Matrix Profile involves finding the highest values in the Matrix Profile. When flat, noisy sequences are present, true discords may be hidden by this effect. Another example is the semantic segmentation technique using the Matrix Profile [3], this technique detects transitions in a signal by analyzing the matches of each subsequence, assuming homogeneous regions will contain many good matches. In this case, homogeneous regions containing flat and noisy sequences will result in poor matches, violating the base principle of the segmentation technique. Notably, motif detection will not suffer from this issue, assuming the user is not interested in flat motifs.

Next, we will discuss seemingly useful resolutions that do not actually manage to solve this effect before presenting our own solution. First however, we introduce a synthetic dataset that will serve as our running example in this section.

4.4.1 Running Example

We generated a sinusoid signal of 2000 samples and introduced an anomaly in one of the slopes by increasing the value of 10 consecutive values by 0.5 and create a noisy

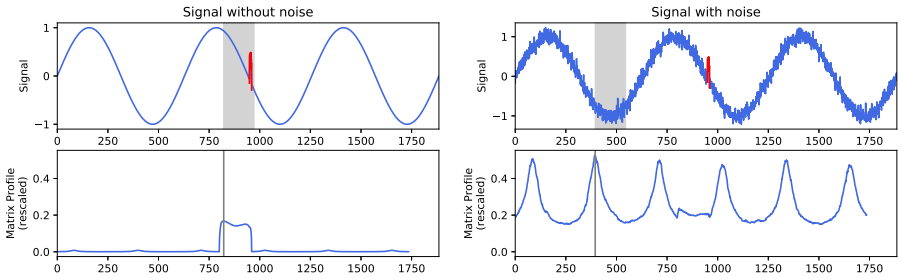


Figure 4.2: Top: Sinusoid signal without (left) and with (right) added Gaussian noise. An anomaly of length 10 (red) was introduced at index 950. Bottom: Corresponding Matrix Profile for both signals, rescaled using the method of Section 4.3.3. The top discord is marked in gray. As can be seen, the presence of noise increases the Matrix Profile values of the flat regions to the degree that they now hide the true anomaly. This figure is modified from our previous work [8].

copy by adding Gaussian noise sampled from $\mathcal{N}(0, 0.01)$. The Matrix Profile for both signals was calculated using a subsequence length m of 100 and a trivial match buffer of $\frac{m}{2}$, as recommended in [9]. The signal and corresponding Matrix Profile are displayed in Figure 4.2. For the noise-free signal, we see exact matches (distance equal to zero) everywhere except in the region containing the anomaly. For the noisy signal, we see how the Matrix Profile has shifted upwards, as would be expected since exact matches are no longer possible. However, we also see previously non-existing peaks in the Matrix Profile where the signal was more flat, because of this the anomaly is no longer trivial to locate automatically.

Let us briefly further investigate how the properties of the noise affect the Matrix Profile in this example. Figure 4.3 displays our starting sinusoidal signal with anomaly, to which Gaussian noise sampled from different distributions was added. As expected, we see that as the variation of the noise increases, the Matrix Profile becomes more deformed. The anomaly is no longer visually obvious in the Matrix Profile for noise with standard deviation of 0.05 or more. Somewhat surprising is how quickly this effect becomes apparent: when the noise has a standard deviation of around 0.02 (at this point the signal-to-noise ratio is 1250 or 31 dB), the anomaly is already occasionally overtaken as the top discord by the flat subsequences (depending on the sampling of the noise).

Before coming to our solution, we will discuss why some simple, seemingly useful methods to circumvent this problem do not work.

- **Changing the subsequence length m**: as m becomes smaller, the effect of any anomaly on the Matrix Profile will indeed increase. However, as the subsequences become shorter and relatively flatter as a result, the effect of the noise also becomes bigger, resulting in a more eratic Matrix Profile. Increasing m will have a beneficial effect, but this is simply because the longer subsequences will

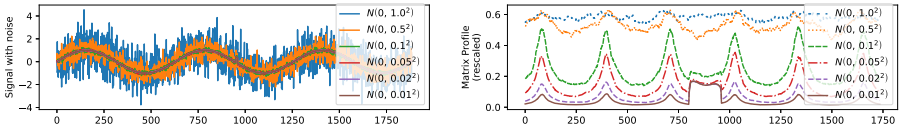


Figure 4.3: A demonstration of the effect of varying degrees of noise on the Matrix Profile. Left: the sinusoidal signal with noise sampled from various Gaussian distributions. Right: The corresponding rescaled Matrix Profile. This figure is modified from our previous work [8].

become less flat in this specific example, so this is not a general solution. This approach is demonstrated in Figure 4.4 (top left).

- **Ignoring flat sections**: ignoring subsequences whose variance is below a certain value would result in the removal of the peaks in the Matrix Profile. A first problem with this is that finding the correct cutoff value is not trivial. Secondly, this approach will not be applicable in datasets where the flat subsequences are regions of interest, either as anomalies or for finding similar subsequences, as is demonstrated in the time series segmentation of Section 4.6. This approach is visualized in Figure 4.4 (middle left).

- **Smoothing or filtering**: by preprocessing the noisy signal, one could hope to remove the noise altogether. Unfortunately, unless the specifics of the noise are well known and the noise can be *completely* separated from the signal, there will always remain an amount of noise. As was shown in Figure 4.3, even a small amount of noise can have a large effect on the Matrix Profile. This approach is demonstrated in Figure 4.4 (bottom left).

4.4.2 Eliminating the Effect of Noise

Ideally, we want flat subsequences to have good matches with other flat subsequences. This would be the case if those flat subsequences were stretched and/or shifted versions of one another, as mentioned in Section 4.3.4. Unfortunately, this is not the case due to the effects of noise. We can however, still consider them to be identical, in which case we can use our derivation from Section 4.3.5, which estimates the effect of the noise on the z-normalized Euclidean distance. By subtracting this estimate during the calculation of the Matrix Profile, we are actively negating the effects of the noise. The only requirement is that we known the standard deviation of the noise that is present in the signal. This may be either known in advance or can be easily estimated by analyzing a flat part of the signal. Note that we also need the standard deviation of the subsequences being compared, but as these are already needed for the distance calculation [10, 11], these are precalculated and available as part of the Matrix Profile calculation.

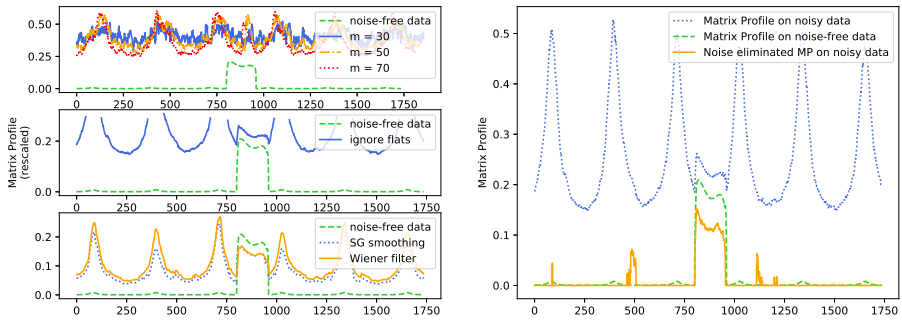


Figure 4.4: Left: The effects of several seemingly useful methods to combat the effect of flat, noisy subsequences that in fact do not work. From top to bottom: reducing the subsequence length, ignoring flat sections and smoothing/filtering. None of these methods approach the noise-free Matrix Profile. Right: The effect of our noise elimination technique on the Matrix Profile. We see how the corrected Matrix Profile closely resembles the Matrix Profile of the noise-free signal. This figure is modified from our previous work [8].

The algorithm is straightforward, after calculating the squared distance between a pair of subsequences using any of the existing algorithms, we subtract the squared estimate of the noise influence. We do this *before* the element-wise minimum is calculated and stored in the Matrix Profile, because this correction might influence which subsequence gets chosen as the best match. Pseudo code is listed in Algorithm 3 and can run in $O(1)$ runtime.

Algorithm 3: Algorithm for Eliminating the Effects of Noise

Input: d: distance between subsequence X and Y
Input: m: subsequence length
Input: σ_X, σ_Y: standard deviation of subsequence X and Y
Input: σ_n: standard deviation the noise
Output: corrDist: corrected distance between subsequence X and Y

1 $corrDist = \sqrt{d^2 - (2+m)\frac{\sigma_n^2}{max(\sigma_X, \sigma_Y)^2}}$

The only difference between this code and the formula from Section 4.3.5 is that we use the maximum standard deviation of both subsequences. When processing two fundamentally different subsequences, this choice effectively minimizes the effect of the noise elimination technique.

We demonstrate our technique on the running example in Figure 4.4 (right). We see that unlike the previously methods, we can closely match the Matrix Profile of the noise-free signal. We do see some small residual spikes, which appear depending on the sampling of the noise, they are caused by local higher-than-expected noise values in that part of the signal.

After demonstrating our noise elimination technique on a limited synthetic dataset,

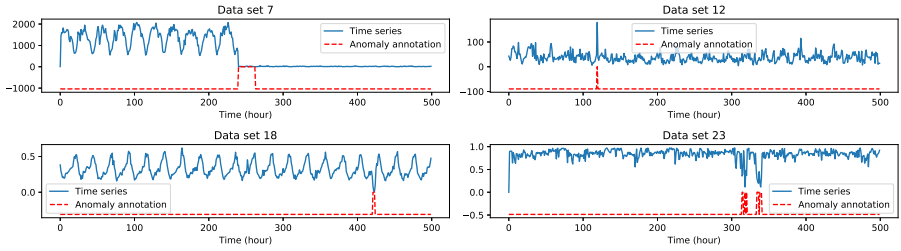


Figure 4.5: Extracts of four series from the Yahoo! Webscope anomaly dataset.

we will use the remainder of this chapter to prove the merit of our noise elimination method for several use cases using real-world datasets.

4.5 Use Case: Anomaly Detection

One of the original applications for the Matrix Profile is the discovery of discords, where a discord is the subsequence in a series that differs most from any other subsequence. Discords in fact correspond to the subsequences starting at the indices where the Matrix Profile is highest. When interested in the top-k discords, one can take the top-k values of the Matrix Profile where each value should be at least m index positions away from all previous discord locations. This requirement ensures we cannot select overlapping subsequences as discords, as these basically represent the same anomaly [23].

In this section we demonstrate the benefit of our noise elimination technique when performing anomaly detection utilizing real-world data from Yahoo. In our previous work [8], we performed a similar experiment using the "realAWSCloudwatch" collection from the Numenta Anomaly Benchmark [24].

We use the **Labeled Anomaly Detection Dataset of Yahoo! Webscope**, consisting of both real and synthetic time-series. For this work we focus on the "A1Benchmark" dataset, which contains real traffic metrics from Yahoo! services, reported at hourly intervals. The benchmark consists of 67 time series with labeled anomalies, ranging from 741 to 1461 data points. The time-series vary considerable, containing diverse ranges, seasonality, trends, variance, among other properties. Figure 4.5 shows some examples.

Rather than classifying each point in the time series as anomalous or normal, which would involve optimizing a classification threshold, we instead score performance by counting the number of attempts needed before all anomalies in a series are reported, or until 10 incorrect guesses are made, as was done in our previous work for the Numenta benchmark [8]. This way of scoring resembles a user being alerted with suspected anomalies, measuring the capability of the algorithm to present relevant anomalies.

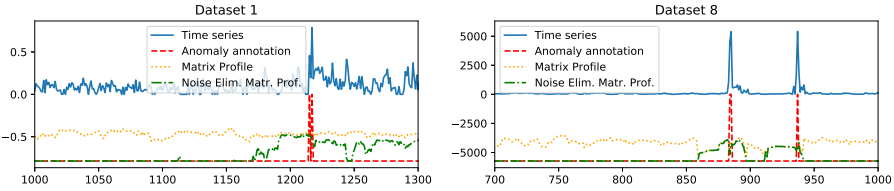


Figure 4.6: Two examples displaying the beneficial effect of our noise elimination on anomaly detection. The anomalies are not noticeable in the regular Matrix Profile, but are obvious after applying noise elimination.

We perform anomaly detection by self-joining each series with a subsequence length of 24 (one day) and using the left Matrix Profile [5] for anomaly detection. The left Matrix Profile only tracks matches preceding each subsequence, similar to how streaming data is processed, and increases the chance to treat sudden changes as discords.

Since we do not know the characteristics of the noise, we would need to estimate the standard deviation using the signal. However, this is not a trivial task since we do not known in advance which signals contain noise and which do not. Making an inaccurate estimate by assuming a non-noisy signal is in fact noisy would result in poor predictions.

To detect the presence of noise, we devised a heuristic where we evaluate the Matrix Profile values of the first three days of data. If the average rescaled Matrix Profile value is above a certain threshold, we assume the start of the data is noisy and we take the median standard deviation of all subsequences in the first three days as noise parameter. Based on the first 10 datasets and leaving all other datasets as test set, we manually determined a threshold of 0.2.

This heuristic marked 34 out of 67 datasets as noisy. We compared the anomaly detection results for these 34 datasets with and without our noise elimination technique. The results are displayed in Table 4.1, they show that our noise elimination technique performed better for 32 out of 34 datasets. On average, the regular Matrix Profile found 36 out of a total of 84 anomalies using 291 incorrect guesses, after applying our technique this improved to 79 found anomalies using only 80 incorrect guesses. This means that on average, one in two suspected anomalies turned out to be correct!

Figure 4.6 shows two close-ups demonstrating the effect of the noise elimination technique. It shows the Matrix Profile having a somewhat consistent high value, whereas the noise eliminated version only increases near the actual anomalies.

Our method was unable to find all anomalies for four of the datasets. Two of these are shown in Figure 4.7. In dataset 53 the first two anomalies were not detected due to their similarity with other flat series. In dataset 61 the algorithm behaves similarly to the original matrix profile due to the sudden increase in the noise level and becomes unable to differentiate anomalies from noise. In this case the first anomaly is found but

Table 4.1: Results of anomaly detection using the Matrix Profile with and without noise elimination on the Yahoo! Webscope anomaly dataset. For each dataset, we kept guessing until all anomalies were found or 10 incorrect guesses were made. When using noise elimination we were able to find most anomalies with few attempts.

		Without Noise Eliminaton		With Noise Elimination	
Dataset	# Anomalies	Found Anomalies	Wrong Guesses	Found Anomalies	Wrong Guesses
1	2	0	10	2	0
2	2	1	10	2	3
4	3	0	10	3	0
5	1	0	10	1	0
6	1	0	10	1	0
8	3	0	10	3	5
10	1	0	10	1	0
11	1	0	10	1	0
12	2	1	10	2	1
14	1	1	8	1	0
17	3	1	10	3	2
19	3	0	10	3	0
21	2	1	10	2	1
22	1	1	8	1	0
23	12	5	10	12	3
24	3	3	3	2	10
25	1	1	2	1	0
31	2	0	10	2	1
32	2	2	5	2	1
33	1	0	10	1	1
40	2	1	10	2	9
41	3	1	10	3	3
42	3	0	10	3	7
43	3	2	10	3	1
45	1	0	10	1	0
48	1	0	10	0	10
50	1	1	2	1	0
53	4	4	3	2	10
58	1	0	10	1	0
61	2	0	10	1	10
62	4*	0	10	4	1
63	1	0	10	1	0
66	6	5	10	6	1
67	5	5	0	5	0
Sum	84	36	291	79	80

* Dataset 62 actually contains five anomalies, but because the first anomaly occurs within the first three days which are used to estimate the noise level, we do not consider it in the results.

the second one is missed.

4.6 Use Case: Semantic Segmentation for Time Series

Semantic segmentation of time series involves splitting a time series into regions where each region displays homogeneous behavior, these regions typically correspond to a particular state in the underlying source of the signal. Applications of segmentation may include medical monitoring, computer-assisted data annotation or data analysis in general. In this section, we perform semantic segmentation on the PAMAP2 activity dataset using the Corrected Arc Curve (CAC). The CAC is calculated by the FLUSS algorithm for batch data or the FLOSS algorithm for streaming data, using the Matrix Profile index [3].

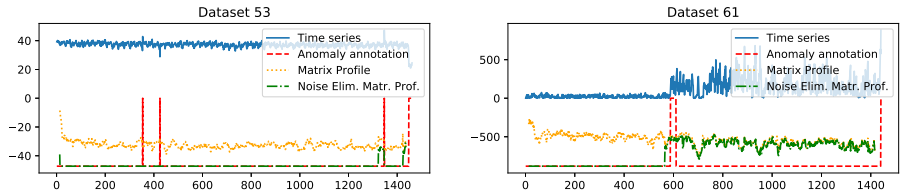


Figure 4.7: Left: Dataset containing two anomalies which are not detected as they closely resemble the estimated noise. The noise eliminated Matrix Profile is zero for this segment. Right: Dataset where the original amount of noise is small but increases at one point, causing the Noise Elimination to lose its effect.

The CAC was introduced as a domain agnostic technique to perform time series segmentation on realistic datasets, with support for streaming data while requiring only a single intuitive parameter (the subsequence length to consider). During evaluation the CAC was found to perform better than most humans on dozens of datasets, allowing the authors to claim "super-human performance" [3].

The CAC is a vector of the same length as the Matrix Profile, and is constructed by analyzing the Matrix Profile Index. They consider arcs running from each subsequence to the location of its nearest match. To calculate the CAC, they compare the number of arcs running over each location against the number of arcs expected if all match locations would be determined by uniform sampling over the entire series. This ratio is defined as the CAC, its values are strictly positive without an upper bound, but can be safely restricted to the range $[0, 1]$. Assuming homogeneous segments will display similar behavior while heterogeneous segments will not, a low CAC value is seen as evidence of a change point, though a high CAC value should not be seen as evidence of the absence of one.

We use the **PAMAP2 Activity Dataset** [25], which contains sensor measurements of 9 subjects performing a subset of 18 activities like sitting, standing, walking, and ironing. Each subject was equipped with a heart rate monitor and 3 inertial measurement units (IMU) placed on the chest, dominant wrist and dominant ankle. Each IMU measured 3D acceleration data, 3D gyroscope data and 3D magnetometer data at 100 Hz. The time series are annotated with the activity being performed by the subject and transition regions in between activities. The duration of each activity varies greatly, but most activities last between 3 to 5 minutes.

The PAMAP2 dataset has already been used in the context of segmentation [12], where the authors used the Matrix Profile to classify the activities in passive and active activities. At one point they note that the motif pairs in the passive actions (such as lying, sitting or standing) are less similar and therefore less useful for segmentation. The underlying problem here is the inability to detect motifs in the passive activities, because they mainly consist of flat, noisy signals. By using our method to compensate for this effect, we will be able to find the needed motifs, resulting in better segmentation

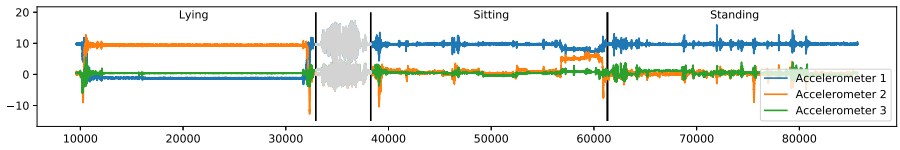


Figure 4.8: Three accelerometer channels of subject 6 from the PAMAP2 dataset. We see three activities and one long transition period. No clear patterns are discernible and many flat and noisy subsequences are present. Reproduced from our previous work [8].

results.

We applied time series segmentation on the passive activities present in the PAMAP2 dataset, focusing on the "lying", "sitting" and "standing" activities. We picked these activities since their measurements display very few patterns in the data and they are performed consecutively for all subjects, meaning we did not have to introduce time-jumps in our experiments.

We considered subjects 1 to 8 of the dataset (subject 9 had no recordings of the relevant activities) and tested both the transition from "lying" to "sitting" as well as the transition from "sitting" to "standing". For each subject, we used the 3 accelerometer signals from the IMU placed on the chest of the subject, any missing data points were filled in using linear interpolation. We calculated the CAC by self-joining each sensor channel with and without noise elimination using a subsequence length of 1000 (10 seconds). The standard deviation of the noise was estimated (without optimizing) by taking the 5th percentile of the standard deviations of all subsequences.

An example of the signals spanning over the 3 passive activities can be seen in Figure 4.8. We emphasize it is not our goal to build the optimal segmentation tool for this specific task, but to simply evaluate the effect of our noise elimination technique on the CAC for sensor signals containing flat and noisy subsequences.

There is one unexpected side effect of the noise cancellation technique that needs to be corrected before calculating the CAC. Because most of the flat subsequences will have an exact match (distance equal to zero) to other flat subsequence, there will be many locations to represent the best match. However, since the Matrix Profile Index only stores one value, the selected best match will become determined by the first or last match, depending on the implementation of the Matrix Profile. This creates a pattern in the Matrix Profile Index, that actually violates the CAC's assumption of matches being spread out over a homogeneous region. Note that this effect is in fact also present in the normal Matrix Profile, but typically has goes unnoticed because multiple exact matches are extremely rare.

To prevent this effect, we need to randomly pick one of the best matches and store its location in the Matrix Profile Index. This is straightforward when using the STOMP algorithm [10], as every step in STOMP calculates all matches for one particular subsequence. If we want to calculate the Matrix Profile in an online fashion using the

SCRIMP algorithm [11], where the matches for one specific subsequence are spread over many iterations, we need to utilize reservoir sampling [26] in the construction of the Matrix Profile Index. Reservoir sampling allows uniform sampling without replacement from a stream without knowing the size of the stream in advance. We use it to sample the stream of best matches. Implementing reservoir sampling requires us to store an additional vector of the same length as the Matrix Profile, to keep track of the number of exact matches that was encountered so far for each subsequence. Pseudo code to update the Matrix Profile and its indices is listed in Algorithm 4.

Algorithm 4: SCRIMP Matrix Profile Update using Reservoir Sampling

Input: dists: distances on diagonal calculated by SCRIMP
Input: indices: corresponding indices of dists
Input: numMatches: number of exact matches per subsequence
Input: mp: part of Matrix Profile vector being updated
Input: mpi: part of Matrix Profile Index being updated

```
   /* Handle new better matches                                    */
1  better = dists < mp
2  mp[better] = dists[better]
3  mpi[better] = indices[better]
4  numMatches[better] = 1
   /* Handle matches equal to current best match                   */
5  equal = dists == mp ∧ finite(dists)
6  numMatches[equal] = numMatches[equal] + 1
7  for i in equal do
8      if random() < 1/numMatches[i] then
9          mpi[i] = indices[i]
```

The code is straightforward. In lines 1 to 3 we update the Matrix Profile and Index if a better match was found and in line four we reset the tracked number of exact matches. In line five, we gather any matches equally good as the match being tracked in the Matrix Profile. Line six increases the counter of any equally good matches found and line seven to nine perform the reservoir sampling to update the Matrix Profile Index for each newly found equal match.

For both experiments, the CAC was calculated using the Matrix Profile with randomly sampled indices. The activity-transition point was taken where the CAC was minimal, ignoring any values in the first and last 50 seconds (5 times the subsequence length), as suggested by the original paper [3]. We considered 4 segmentations per subject: one CAC for each of the three sensor channels and one obtained by averaging the individual CACs.

To evaluate the ability to predict the transition period, we define the score as the normalized distance between the predicted transition and the ground truth transition.

Note that some ground truth transitions are instantaneous, while others consist of a transition period, as can be seen in Figure 4.8. We also added an additional buffer period equal to the subsequence length m (10 seconds) before and after the transition period that we still consider as correct to consider the detection interval of the Matrix Profile. Pseudo code for our scoring function is listed in Algorithm 5, a score will range from 0 to 100, where lower is better.

Algorithm 5: Scoring Function for Semantic Segmentation

Input: estimate: estimated transition
Input: trueStart, trueEnd: ground truth start and end of transition
Input: n: length of series (both activities and transition period)
Input: m: subsequence length / transition buffer
Output: score

if $estimate < trueStart - m$ **then**
  $score = ((trueStart - m) - estimate)/n * 100$
else if $estimate > trueEnd + m$ **then**
  $score = (estimate - (trueEnd + b)/n * 100$
else
  $score = 0$

Table 4.2: Scores for the segmentation of the transition from "lying" to "sitting" using the 3 chest accelerometers from the PAMAP2 dataset for subjects 1 through 8, with and without noise elimination applied. Segmentation is performed using the CAC from a single sensor (C1, C2 and C3) and using the average of the 3 CACs (combined). Similar or better performance are achieved when applying noise elimination for all subjects except subject 1. Results are reproduced from our previous work [8].

	Without Noise Elimination				With Noise Elimination			
Subject	C1	C2	C3	Combined	C1	C2	C3	Combined
1	**5.9**	31.3	**31.9**	**31.7**	41.3	31.8	41.8	36.7
2	32.9	1.4	1.4	1.4	28.8	1.4	1.7	1.4
3	35.9	2.8	31.1	33.8	**2.4**	2.3	**2.3**	**2.3**
4	0.0	2.8	5.9	0.0	0.0	1.5	6.6	0.8
5	1.1	7.6	5.1	3.9	1.6	**1.7**	4.9	1.6
6	2.5	1.9	2.3	2.3	2.4	1.9	2.0	2.4
7	0.1	1.8	11.1	2.0	2.1	1.8	**1.9**	1.9
8	0.0	1.4	5.5	1.7	0.0	1.4	1.4	1.4
Average			9.32	9.61			7.71	6.07

Table 4.3: Scores for the segmentation of the transition from "sitting" to "standing" using the 3 chest accelerometers from the PAMAP2 dataset for subjects 1 through 8, with and without noise elimination applied. Segmentation is performed using the CAC from a single sensor (C1, C2 and C3) and using the average of the 3 CACs (combined). Overall, we see similar or better performance when applying noise elimination, except for the segmentation using the first channel for subject 1, 3 and 8. Results are reproduced from our previous work [8].

	Without Noise Elimination				With Noise Elimination			
Subject	C1	C2	C3	Combined	C1	C2	C3	Combined
1	**32.5**	0.0	3.6	2.2	38.7	0.0	3.7	2.2
2	36.5	37.2	36.4	37.0	**7.1**	**30.0**	32.7	**29.2**
3	**10.0**	30.2	43.1	**30.2**	43.2	**14.0**	43.7	43.2
4	7.8	1.9	1.1	1.2	**0.7**	2.0	1.3	1.3
5	13.1	0.0	28.5	10.6	13.3	1.0	**1.2**	**1.0**
6	36.1	36.6	26.9	36.6	**23.3**	**3.4**	26.5	**3.2**
7	43.1	38.0	16.5	16.5	43.4	**1.6**	**0.0**	**1.6**
8	**2.3**	1.0	24.8	1.0	21.1	0.0	**16.5**	1.0
Average			21.12	16.9			**15.35**	**10.3**

Table 4.2 lists the results for the segmentation when transitioning from "lying" to "sitting". For all subjects expect subject one, the results show similar improved scores for segmentation using individual sensors as well as the combined approach when using the noise elimination technique. The average score for the individual sensors improves from 9.32 to 7.71, a modest improvement corresponding to a gain of about 8 seconds. The segmentation based on the combined CACs improves from 9.61 to 6.07, a gain of about 18.5 seconds. Note that most scores without noise elimination were already very good, leaving little room for improvement. The bad results for subject one can be explained by an incorrect early estimate which is caused by movement of the subject near the start of the "lying" activity. Note that subject one has bad scores for both techniques.

Table 4.3 lists the results for the transition from "sitting" to "standing". Like the previous experiment, we see similar or improved results when applying noise elimination, except for subjects one, three and eight using the first sensor series and for the combined approach for subject three. While the overall scores are worse, the gain by enabling noise elimination is more significant. The average result for a single sensor improves from 21.12 to 15.35, corresponding to a gain of about 27 seconds. When using the combined approach the average score improves from 16.9 to 10.3, a gain of around 31 seconds.

Though limited in scope, these result indicate that the noise elimination technique improves the ability of the CAC to detect transitions in a series containing flat and noisy subsequences.

4.7 Use Case: Data Visualization

Data visualization is a great tool for exploring newly acquired data or for finding similar data in a larger data collection. Unfortunately, time series data for realistic use cases is typically very lengthy, making trivial visualizations useless as these will fail to simultaneously capture the overall and minute details of a series. More advanced techniques summarize the series in a way that is still useful to gain insight into the data. The visualization technique used in this section is the Contextual Matrix Profile (CMP), recently introduced by the authors [14].

The CMP is a generalization of the Matrix Profile that tracks the best match between predefined ranges whereas the Matrix Profile tracks the best match for every possible sliding window location. The distinction between the two can also be made in terms of the implicit distance matrix, defined by the pairwise distance between all subsequences of two series. Where the Matrix Profile equals the column-wise minimum of the distance matrix, the CMP consists of the minimum over rectangular areas of the distance matrix.

For this use case, we use the **Chest-Mounted Accelerometer Dataset** [27], an activity recognition dataset publicly available at the UCI repository[1]. The dataset contains data of 15 subjects performing seven different activities, measured using a chest-mounted accelerometer sampling at 52Hz. The data is labeled with the corresponding activity, though visual inspection reveals the labels seem misaligned for some subjects. The activities performed are: Working at a computer; standing up, walking, going up-/down stairs; standing; walking; going up/down stairs; walking while talking; talking while standing. We selected this dataset for this use case as it contains both activities with a periodic nature as well as passive activities where the accelerometer signal consists of mainly noise.

For the remainder of this section, we focus on the first subject of the dataset due to space constraints, though similar results were obtained for all subjects. This series comprises 52 minutes of data containing nine regions of activity, it is visualized in Figure 4.9. In the top of the figure we see the complete dataset with annotations indicating the activity regions. At the bottom of the figure, close-ups of the signal for four different activities are displayed. While the "working at computer" consists mainly of a flat signal, a periodic pattern is visible in the channels of the other activities, with the "walking" activity having the clearest pattern. Note that the three data channels are uncalibrated, which is not an issue since the z-normalization focuses on the shape of subsequences rather than the absolute values.

The CMP for each data channel is calculated by self-joining the data, using a subsequence length of 52 (1 sec) and specifying contexts of length 469 at 520 (10 secs) intervals. The context length is chosen so that matches can never overlap. Calculating the CMP comes down to dividing the series in non-overlapping, contiguous 10 sec win-

[1] https://archive.ics.uci.edu/ml/datasets/Activity+Recognition+from+Single+Chest-Mounted+Accelerometer

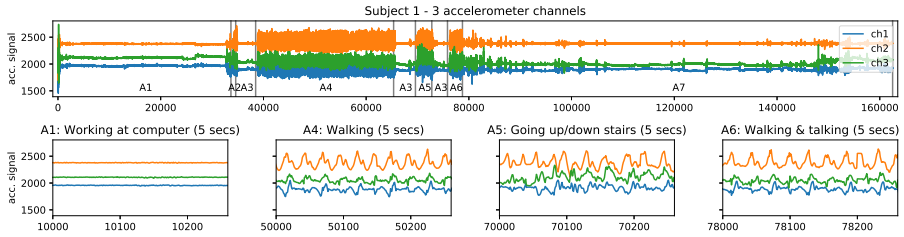


Figure 4.9: Top: Uncalibrated accelerometer data (3 channels) for subject 1, sampled at 52Hz, for a total of 52 mins. The annotations A1 to A7 indicate the corresponding activity labels. The activities performed are computer work, standing up/walking/stairs, standing, walking, standing, stairs, standing, walking while talking, and talking while standing. Bottom: four extracts of 4 different activities, each 5 secs long.

dows and finding the best one second match for each pair of windows. The resulting CMP is a 312 by 312 matrix, each value representing the distance of the best match of the intervals defined by the row and column. The resulting CMPs for each data channel can be seen in Figure 4.10 (left). To demonstrate the value of the visualization, we also added grayscale band to the CMP outline that corresponds to the activity labels present in the data.

To interpret a CMP visualization, one should consider both axes represent the flow of time, starting at the origin. Each value in the CMP indicates how well one region of time matched another region, low (dark) values represent good matches while high (light) values represent bad matches. Looking at Figure 4.10 (left), we can observe a number of things. Most obvious is the symmetry of each CMP, this is because we performed a self-join. We also see the dark square centered at index 100 in all three channels, this indicates a period containing a repetitive pattern across all channels. This region corresponds to the "walking" activity in the dataset, as can be seen by referencing Figure 4.9. Next, for channel two we see similar dark regions for the "stairs" and "walking while talking" activities, meaning there is a similarity between all three activities based on channel two. Though these activities also appear in channel one and three, they are less visually noticeable, especially the "stairs" activity in channel one could be easily missed. The same can be said for the very short "standing up/walking/stairs" activity that precedes the walking activity. One final observation is the presence of high value bands occurring across all channels to some degree between indices 0 and 60 and between 200 and 250. While the first band corresponds to the "computer work" activity, there is no clear-cut corresponding activity for the second band. Most likely, these artefacts are caused by regions with very flat signals, where the noise has a large effect on the distance calculation.

Next, we calculated the CMPs while using the noise elimination technique, keeping all other parameters equal. We estimated the noise parameter for each channel by sliding a one second window over the entire series, calculating the standard deviation

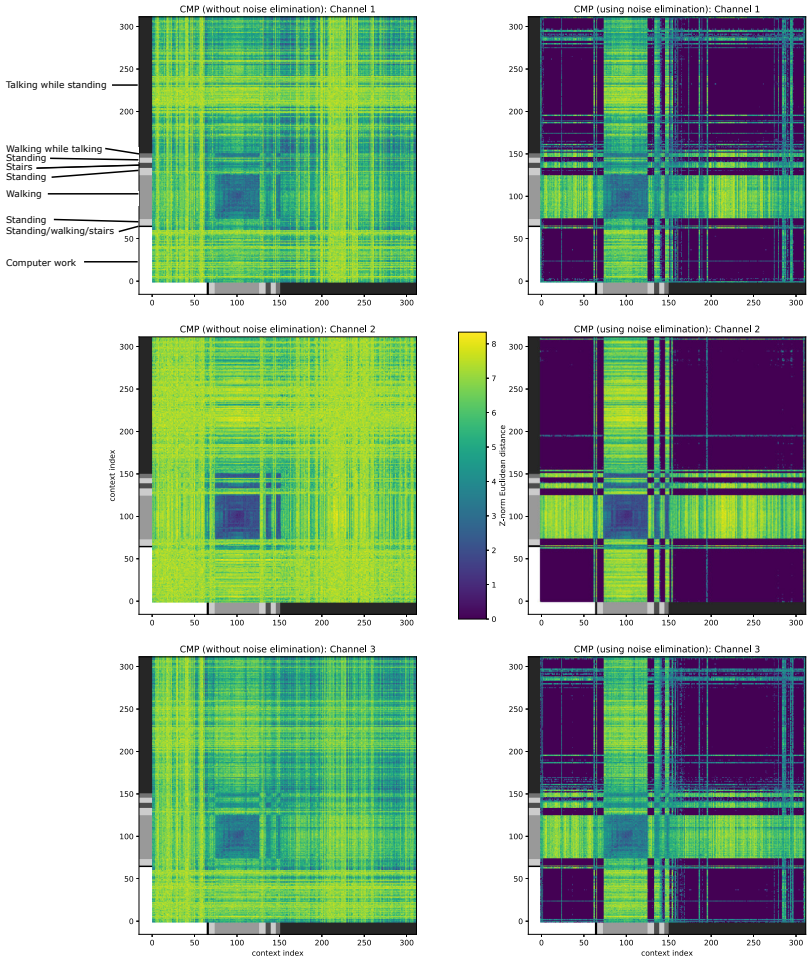


Figure 4.10: CMPs produced for each channel of the dataset using z-normalized Euclidean distance without (left) and with (right) compensating for the noise. The left and bottom grayscale bar for each CMP corresponds to the different activity labels for each of the time windows. We see how the noise elimination results in large areas of exact matches for the passive activities, because of this, the different transitions between active and passive activities is clearly visible. In case the activity labels were unknown, the CMP would have given a good indication of the different regimes present in the signal.

for each location and taking the value corresponding to the fifth percentile, similar as we did in Section 4.6. For reference, these values were: 2.6, 2.3 and 2.7 respectively. The resulting CMPs are visualized in Figure 4.9 (right).

We see the CMPs generated using noise elimination now have additional large, rectangular regions with low values. As can be seen from the activity markings, these regions correspond to the passive activities (computer work, standing and talking while standing), where the series is flat and lacks distinctive patterns. Looking in detail, we see how the activity markings almost perfectly line up with the transitions in the CMP, which is major difference with the CMPs without noise elimination. We do see some line artefacts in the final activity of the dataset for all channels, near index 190 and 280, which correspond to increases in the signal on all three channels. The questions whether or not there is an activity change occurring there is in a way debatable and may simply be a question of tweaking the noise parameter or refining the activity labels.

Of course, the CMP visualization can provide more insights than simply the difference between passive and active activities. It can also be used to differentiate between different activities, provided these activities have different underlying patterns. As an example, we can see that for the CMP of the first channel, the activities "walking" and "walking while talking" have better matches than "walking" and "stairs". This is not the case for channels two and three. When looking at the closeup data of Figure 4.9, we can in fact see a more distinct pattern for channel one for the "stairs" activity. We do not quantify the ability to discern various activities as it is not in scope of this work, our goal was simply to demonstrate the added benefit of noise elimination when visualizing data with the CMP.

To conclude, we demonstrated the effect of the noise elimination technique for data visualization using the CMP on accelerometer data for an activity dataset. Flat signals, such as those from recording passive activities, will result in high values in the CMP and might make it difficult to see the patterns of the underlying data. If we apply the noise elimination technique while calculating the CMP, the passive activities become easily discernible as regions of low values, giving the user better insight in the underlying structure of the data and allowing the user to focus more on the more salient parts. Importantly, the regions with active activities are unaffected by the noise elimination, meaning we can apply noise elimination without risk.

4.8 Conclusion

In this chapter we explained the unintuitive behavior of the z-normalized Euclidean distance when comparing sequences that are flat and noisy, and demonstrated how this negatively affects the Matrix Profile and techniques using the Matrix Profile as building block. We discussed several properties of the z-normalized Euclidean distance, including an estimation of the effect of noise, which we use to eliminate this effect altogether.

We applied our noise elimination technique on three different use cases involving real-world data from open data sets. For anomaly detection on the Yahoo! Webscope anomaly dataset, we were able to automatically guess twice as many anomalies while utilizing less than one third of attempts when using our technique. When used for semantic time series segmentation, we showed an improved accuracy for detecting the transition between two passive activities. Finally, in our visualization use case, we showed a major change in the visualization of activity data using the Contextual Matrix Profile, allowing us to separate the underlying activities that were previously indistinguishable.

Since our technique is conceptually simple, users should be able to reason whether or not their use case will benefit from our technique. Our technique is straightforward to implement and incurs only a constant factor overhead, so it can be used by everyone using Matrix Profile related techniques working with data containing flat and noisy subsequences.

Future remains on more robust noise estimations and dealing with series where noise characteristics change over time.

References

[1] Spiros Papadimitriou and Christos Faloutsos. “Streaming Pattern Discovery in Multiple Time-Series”. In: *International Conference on Very Large Data Bases (VLDB)* (2005), pp. 697–708.

[2] Xing Wang, Jessica Lin, Nital Patel, and Martin Braun. “A Self-Learning and Online Algorithm for Time Series Anomaly Detection, with Application in CPU Manufacturing”. In: *Proc. 25th ACM International on Conference on Information and Knowledge Management - CIKM ’16*. New York, New York, USA: ACM Press, 2016, pp. 1823–1832. ISBN: 9781450340731.

[3] Shaghayegh Gharghabi, Yifei Ding, Chin-Chia Michael Yeh, Kaveh Kamgar, Liudmila Ulanova, and Eamonn Keogh. “Matrix Profile VIII: Domain Agnostic Online Semantic Segmentation at Superhuman Performance Levels”. In: *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, Nov. 2017, pp. 117–126. ISBN: 978-1-5386-3835-4.

[4] Chin-Chia Michael Yeh, Helga Van Herle, and Eamonn Keogh. “The matrix profile allows visualization of salient subsequences in massive time series”. In: *Proc. IEEE Int. Conf. on Data Mining, ICDM* (2017), pp. 579–588. ISSN: 2374-8486.

[5] Yan Zhu, Makoto Imamura, Daniel Nikovski, and Eamonn Keogh. “Time Series Chains: A New Primitive for Time Series Data Mining”. In: *IEEE Int. Conf. on Data Mining (ICDM)*. 2017, pp. 695–704. ISBN: 978-1-5386-3835-4.

[6] Abdullah Mueen, Krishnamurthy Viswanathan, CK Gupta, and Eamonn Keogh. "The fastest similarity search algorithm for time series subsequences under Euclidean distance". In: (2015).

[7] Eamonn Keogh and Shruti Kasetty. "On the need for time series data mining benchmarks". In: *Proc. 8th ACM SIGKDD international conference on Knowledge discovery and data mining* (2002), p. 102. ISSN: 13845810.

[8] Dieter De Paepe, Olivier Janssens, and Sofie Van Hoecke. "Eliminating Noise in the Matrix Profile". In: *Proc. 8th International Conference on Pattern Recognition Applications and Methods*. Feb. 2019, pp. 83–93. ISBN: 978-989-758-351-3.

[9] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. "All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets". In: *IEEE 16th Int. Conf. on Data Mining (ICDM)*. 2016, pp. 1317–1322. ISBN: 978-1-5090-5473-2.

[10] Yan Zhu, Zachary Zimmerman, Nader Shakibay Senobari, Chin-Chia Michael Yeh, Gareth Funning, Philip Brisk, and Eamonn Keogh. "Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins". In: *2016 IEEE 16th Int. Conf. on Data Mining (ICDM)* (2016), pp. 739–748.

[11] Yan Zhu, Chin-Chia Michael Yeh, Zachary Zimmerman, Kaveh Kamgar, and Eamonn Keogh. "SCRIMP++: Time Series Motif Discovery at Interactive Speeds". In: *IEEE Int. Conf. on Data Mining (ICDM)*. 2018, pp. 837–846. ISBN: 978-1-5386-9159-5.

[12] Chin-Chia Michael Yeh, Nickolas Kavantzas, and Eamonn Keogh. "Meaningful Multidimensional Motif Discovery". In: *IEEE Int. Conf. on Data Mining (ICDM)*. IEEE, 2017, pp. 565–574. ISBN: 978-1-5386-3835-4.

[13] Yan Zhu, Abdullah Mueen, and Eamonn Keogh. "Admissible Time Series Motif Discovery with Missing Data". In: (2018). arXiv: 1802.05472. URL: %5Curl%7Bhttps://arxiv.org/pdf/1802.05472.pdf%7D.

[14] Dieter De Paepe, Sander Vanden Hautte, Bram Steenwinckel, Filip De Turck, Femke Ongenae, Olivier Janssens, and Sofie Van Hoecke. "Generalizing the Matrix Profile". In: *Engineering Applications of Artificial Intelligence* (). Submitted 2019-06-06.

[15] Reza Akbarinia and Bertrand Cloez. "Efficient Matrix Profile Computation Using Different Distance Functions". 2019. URL: https://arxiv.org/abs/1901.05708.

[16] Diego Furtado Silva and Gustavo E. Batista. "Elastic Time Series Motifs and Discords". In: *17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2018, pp. 237–242. ISBN: 978-1-5386-6805-4.

[17] Hoang Anh Dau and Eamonn Keogh. "Matrix Profile V: A Generic Technique to Incorporate Domain Knowledge into Motif Discovery". In: *Proc. 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017, pp. 125–134. ISBN: 9781450348874.

[18] Chin-Chia Michael Yeh, Nickolas Kavantzas, and Eamonn Keogh. "Using Weakly Labeled Time Series to Predict Outcomes". In: *Proc. of the VLDB Endowment* 10.12 (2017), pp. 1802–1812. ISSN: 21508097.

[19] Shaghayegh Gharghabi, Shima Imani, Anthony Bagnall, Amirali Darvishzadeh, and Eamonn Keogh. "MPdist: A Novel Time Series Distance Measure to Allow Data Mining in More Challenging Scenarios". In: *IEEE Int. Conf. on Data Mining (ICDM)*. 2018, pp. 965–970. ISBN: 978-1-5386-9159-5.

[20] Shima Imani, Frank Madrid, Wei Ding, Scott Crouter, and Eamonn Keogh. "Time Series Snippets: A New Primitive for Time Series Data Mining". In: *IEEE Int. Conf. on Big Knowledge (ICBK)*. IEEE, 2018, pp. 382–389. ISBN: 978-1-5386-9125-0.

[21] D. Rafiei. "On similarity-based queries for time series data". In: *Proceedings 15th International Conference on Data Engineering*. IEEE, 1999, pp. 410–417. ISBN: 0-7695-0071-4.

[22] Michele Linardi, Yan Zhu, Themis Palpanas, and Eamonn Keogh. "Matrix Profile X". In: *Proc. 2018 Int. Conf. on Management of Data (SIGMOD)*. 2018, pp. 1053–1066. ISBN: 9781450347037.

[23] Abdullah Mueen, Eamonn Keogh, Qiang Zhu, Sydney Cash, and Brandon Westover. "Exact Discovery of Time Series Motifs". In: *Proc. SIAM International Conference on Data Mining*. 2009. ISBN: 9781615671090.

[24] Alexander Lavin and Subutai Ahmad. "Evaluating Real-Time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark". In: *IEEE 14th Int. Conf. on Machine Learning and Applications (ICMLA)*. Dec. 2015, pp. 38–44. ISBN: 978-1-5090-0287-0. arXiv: 1510.03336.

[25] Attila Reiss and Didier Stricker. "Introducing a new benchmarked dataset for activity monitoring". In: *International Symposium on Wearable Computers*. 2012, pp. 108–109. ISBN: 9780769546971.

[26] Jeffrey S Vitter. "Random sampling with a reservoir". In: *ACM Transactions on Mathematical Software (TOMS)* 11.1 (1985), pp. 37–57.

[27] Pierluigi Casale, Oriol Pujol, and Petia Radeva. "Personalization and user verification in wearable systems using biometric walking patterns". In: *Personal and Ubiquitous Computing* 16 (June 2012), pp. 1–18.

Chapter 5

Mining Recurring Patterns in Real-Valued Time Series using the Radius Profile

Motifs are those subsequences that are most similar to other subsequences in a time series. They can reveal insights into the data or serve as prerequisite for more advanced analytics. Motifs are most often defined in the context of a single best match, but we can also define them in the context of multiple matches. One recent work introduced Ostinato, a method for finding the top-k consensus motifs, i.e. the best preserved subsequences that are repeated in a collection of series. This chapter further refines consensus motif discovery by introducing the Radius Profile. Similar to the matrix profile, the Radius Profile is a derived time series that can be used to extract consensus motifs but can be directly used for analytics. By exploiting the anytime techniques available to the matrix profile, we can obtain a representative Radius Profile in a fraction of the time needed for Ostinato to find a single consensus motif. Additionally, we tackle the even harder question of how to find repeated sequences in a collection of series, irrespective of the position of their occurrence. In analogy to the consensus motifs, we name these repeating patterns "common motifs" and define the common-k Radius Profile, which can be used as a new insightful primitive, or to locate these common motifs.

My contributions can be summarized as follows:

1. Introducing the Radius Profile as a new time series primitive that can be used to find consensus motifs. Our method for calculating it can be used as an anytime algorithm and supports variants such as the k-of-n variant that considers only a subset of the series.

2. Introducing the common-k Radius Profile as a new series primitive that can be used to find common motifs.

Mining Recurring Patterns in Real-Valued Time Series using the Radius Profile

D. De Paepe and S. Van Hoecke

This chapter is an extended version of the paper published in "Proceedings of the 20th International Conference on Data Mining (ICDM)"

Abstract Time series analysis is becoming more popular in both research and industry. One recent innovation is the Ostinato algorithm, which finds the best preserved patterns that are repeated in a collection of series, i.e. consensus motifs and corresponding radii. However, Ostinato only works as a batch algorithm, can only find the top-k patterns and only finds patterns that are repeated in multiple series. Furthermore, the runtime of Ostinato can vary widely depending on the input series and setup parameters. To tackle these limitations, we present two algorithms in this chapter that can answer broader questions. First, we created an anytime version of Ostinato, called Anytime Ostinato, that finds the exact consensus radius for each subsequence, i.e. the radius profile, or can estimate these radii in a fraction of the time. Second, we designed a batch algorithm, called Single Series Ostinato, that finds the radius profile for a single series allowing us to detect repeating patterns in a single series, which is not possible for Ostinato. In this chapter we explain both algorithms and apply them to the REFIT and PAMAP2 datasets respectively. The Anytime Ostinato algorithm gives an estimated but representative radius profile in less time than it takes for Ostinato to find the top-1 consensus motif and radius. The radius profile found by the Single Series Ostinato algorithm clearly marks regions with repeating patterns and allows us to extract representative subsequences. We believe the radius profile can also be useful beyond finding repetitions and act as a building block, similar as to how the Matrix Profile captures motif information but has since been used in many other techniques.

5.1 Introduction

With the rise of IoT and continuous monitoring, time series data is becoming more and more prevalent in various domains, and proper analysis of this data can lead to valuable monetary or societal advantages. For example, manufacturers can monitor machine behavior to find outliers in their products or production performance, traffic analysis might lead to better city planning, or medical studies might benefit from activity recognition using non-intrusive sensors. In the relatively new time series analysis domain, several topics are being researched today. These include broad topics like classification or anomaly detection, but also specialized techniques such as discord or motif discovery, that can act as building blocks for other techniques.

Whereas many publications deal with finding the best matching subsequence (motif discovery) in time series, only one (recent) publication tackles finding the most similar subsequence in a collection of series, the so called *consensus motif*. Consen-

sus motifs can be directly used to find repeating patterns (e.g., daily occurrences), to extract atomic patterns from a signal stream (e.g., isolating letters from eye tracking movement signals [1]), and may have applications for time series classification using shapelets [2].

Given a collection of series and a subsequence length, the Ostinato [1] algorithm finds the subsequence s and corresponding minimal distance r (the *radius*), so that for each series the distance from s to the best matching subsequence of that series is r or less. Since we can consider a single, partitioned series as a collection of individual series, Ostinato is also applicable to partitioned series. Furthermore, a straightforward adjustment to Ostinato allows us to find the top-k consensus motifs instead of only the top-1.

However, the Ostinato algorithm has two important restrictions. First of all, it assumes a good match is to be found in every series. If this is not the case, the algorithm will take a substantial longer time and may produce unintuitive results. A variant that drops this assumption exists but requires insight in the data and a longer runtime. Second, Ostinato assumes we only care about the single best match in every series. This means that multiple occurrences of a pattern within a single series are disregarded. As a result, we cannot use Ostinato to find the most common pattern in a (collection of) series where we do not know in advance where the pattern occurs.

To solve these shortcomings we present two innovations. First, we present an anytime version of the Ostinato algorithm that finds the consensus radius for *all* subsequences, i.e. the *radius profile*, rather than the top-k consensus motifs and corresponding radii. Here, the runtime is independent of the presence of good matches. Furthermore, due to the anytime property, we can shorten the runtime to obtain approximated results. We show that we get representative results for all subsequences in less time than it takes Ostinato to find the top-1 result and that in some cases the complete, exact calculation is faster than the top-1 Ostinato calculation. Secondly, we present an algorithm that finds the radius profile for a single series, allowing us to find repetitions in a single, non-partitioned series or in multiple series while ignoring where these repetitions occur.

The remainder of this chapter is structured as follows. Section 5.2 introduces notation and definitions. Related work on finding repeated patterns is presented in Section 5.3. Section 5.4 covers the inner workings and runtime insights of the Ostinato algorithm and introduces a visual representation which we will reuse in further sections. In Section 5.5, we introduce our Anytime Ostinato algorithm to find the radius profile for all subsequences, and demonstrate its performance using the REFIT power consumption dataset [3]. In Section 5.6, a second algorithm is introduced, i.e. Single Series Ostinato, which calculates the radius profile for a single series. Results of this algorithm are demonstrated on the PAMAP2 activity dataset [4]. We conclude the chapter in Section 5.7.

The source code for our algorithms and the experiments listed in this chapter have been made available online, so others can verify or extend our work [5].

5.2 Notation & Definitions

We start by defining the common concepts of *series* and *subsequences*.

Definition 5.1 (Series) A series $S \in \mathbb{R}^n$ is an ordered collection of n real values $(s_1, s_2 \dots s_n)$.

Definition 5.2 (Subsequence) A subsequence $S_{i,m}$ is the continuous subsequence of S starting at index i of length m: $(s_i, s_{i+1} \dots s_{i+m-1})$. It has to fall completely within S: $(1 \leq i \leq n - m + 1)$.

In the context of pattern matching, the distance between two subsequences is often defined as the z-normalized Euclidean distance. Other measures such as Euclidean distance or cosine distance can be chosen as well, as long as a lower distance value indicates a better match.

Definition 5.3 (Z-normalized Euclidean distance) The z-normalized Euclidean distance $D(A, B)$ between 2 series of equal length $A \in \mathbb{R}^m$ and $B \in \mathbb{R}^m$ is defined as follows, where μ and σ represent the mean and standard deviation respectively:

$$D(A,B) = \sqrt{\left(\frac{a_1 - \mu_A}{\sigma_A} - \frac{b_1 - \mu_B}{\sigma_B}\right)^2 + \ldots + \left(\frac{a_m - \mu_A}{\sigma_A} - \frac{b_m - \mu_B}{\sigma_B}\right)^2}$$

Definition 5.4 (Radius) The radius r of a subsequence $Q \in \mathbb{R}^m$ with respect to a collection of series $\Sigma = (S^1, \dots, S^k)$, is the maximum of the distances between Q and the best matching subsequence from each series. In other words, it is the minimal distance needed to reach at least one subsequence in each series, starting from Q.

$$\min_r \forall S \in \Sigma, \exists i : D(Q, S_{i,m}) \leq r$$

Definition 5.5 (Consensus motif) Given a collection of series $(S^1, \dots, S^k)$, the consensus motif of a given length m is the subsequence $S^j_{i,m}$ taken from one of those series, such that the radius is minimal.

5.3 Related Work

A substantial amount of research exists on finding repeating patterns in symbolic series, mainly driven by the temporal rule mining community. Just as frequent pattern mining is commonly used in rule mining, frequent episode mining is used in temporal rule mining, where an episode consists of an ordered list of event types. Han et al. made an extensive survey on temporal mining related techniques [6]. Similarly, in the biomedical domain, common patterns in DNA are called consensus motifs and

various techniques focus on discovering these [7]. Other techniques originate from the music domain, where repeating patterns can be used for audio segmentation, beat detection or summarising music. For example, Hsu et al. present a technique, where they use a correlative matrix to find non-trivial repetitions in music, representing notes as symbols [8]. However, methods for symbolic series are not directly applicable to real-valued time series.

To the best of our knowledge, little to no work can be found regarding repeating patterns in real-valued series. Motif discovery techniques, such as Matrix Profile [9], aim to find the best matching subsequence pair rather than the most common one. However, the best match is not guaranteed to also be part of the most common pattern-group. An adaptation of the Matrix Profile technique has been used to find time series chains, which are repeating patterns that slowly change throughout time [10]. Again, there is no guarantee that these chains will overlap with the best repeating pattern, because time series chains assume each pattern is most similar to the immediately preceding and later occurrence. Another variation is the Time Series Snippets algorithm [11], which looks for representative subsequences to summarize a series. However, this algorithm only considers a subset of all subsequences and will evaluate similarity against the entire dataset, making it unsuited for cases where we want to find well-preserved patterns with few repetitions. From the music domain, REPET is a Fourier based technique to separate foreground audio from repeating background music [12]. The technique assumes the pattern is repeated continuously, as is often the case in background music, making it unsuited to find patterns that are spread throughout a series. To conclude, it seems only the Ostinato algorithm, described in the recent work by Kamgar et al. deals with finding repeated patterns in multiple real-valued series [1].

5.4 The Ostinato Algorithm

In this section, we provide an overview on how Ostinato works and highlight some of its properties. However, we first introduce the distance matrix as a visual aid for this and later sections.

Given a number of time series $S_1, \ldots, S_n$ and a subsequence length m, we can represent the pairwise distance of all subsequences as a distance matrix D, as shown in Figure 5.1. Here, both axes represent all possible subsequences. That is, element $D[i, j]$ contains the (z-normalized Euclidean [9]) distance between the i-th and j-th subsequence. Note that we display the series along both axes for visual clarity, but in reality the axes are shorter than the series as there are only $l - m + 1$ subsequences in a series of length l.

Figure 5.1 also shows the relation between the distance matrix and a consensus motif. For any subsequence c, we can calculate the distance between c and the best matching subsequence of all series (i.e. d_1, d_2, d_3 and d_4 in Figure 5.1). The maximum

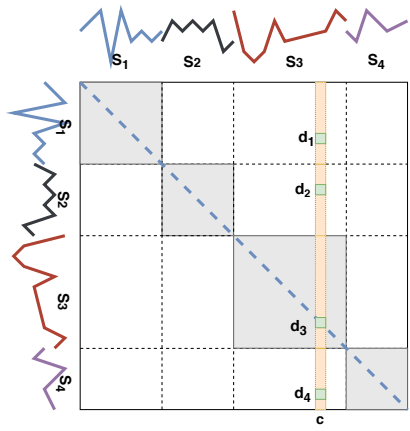


Figure 5.1: Distance matrix for four series $S_1, \ldots, S_4$. Each row i and column j represents a single subsequence of one series and the element at index $[i, j]$ corresponds to the distance between those subsequences. As a side effect of this, the main diagonal (dashed blue) consists of all zeros. The consensus radius of a specific subsequence c is calculated by finding the best matching subsequence for all series and taking the maximum of these distances $(d_1, \ldots, d_4)$. The top-k consensus motifs are the k subsequences with the minimum radius.

value of these distances is defined as the radius r. The top-k consensus motifs are the k subsequences for which the radius is minimal.

Ostinato finds the top-1 consensus motif and corresponding radius using a straightforward greedy branch and bound approach, as visualized in Figure 5.2. For series S_1, it calculates all subsequence distances between S_1 and S_2, tracking the best distance for each subsequence of S_1 (see Figure 5.2, step 1). Next, the subsequence with the lowest distance is considered the candidate consensus motif and its radius is calculated by determining the distance with all other series (step 2). At this point, Ostinato has an upper bound and repeats the process for all other subsequences of S_1, aborting the search if at any time the upper bound is passed (step 3). After all subsequences of S_1 have been considered, the search is repeated for the next series (steps 4 and further). Pseudocode is listed in Algorithm 6 and is a composition of the original pseudocode and the accompanying reference implementation [1].

Due to the branch and bound approach, the runtime of Ostinato is dependent on details of the time series. In a very bad case, Ostinato has to calculate a large portion of the distance matrix (excluding the self-join of each series). In the very best case, it has to calculate n full-series calculations (i.e. steps 1, 4, 6 and 8 in Figure 5.2) and *one* single column (step 2). In more realistic cases, the number of columns calculated will depend on the relation between the found upper bound and any distances that remain to be calculated. As an unfortunate side effect of this, the performance of Ostinato

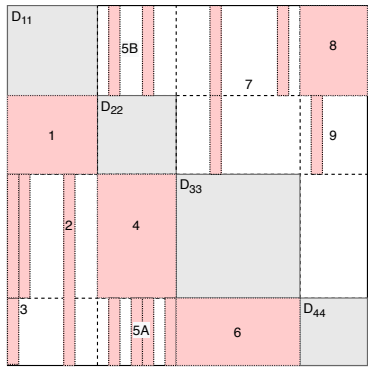


Figure 5.2: Visualization of the distance calculations made in different steps by the Ostinato algorithm. For each series, a full pairwise distance calculation is performed to the next series (steps 1, 4, 6 and 8), after which distances to all remaining series are calculated (steps 2, 3, 5, 7 and 9). By using a best-so-far threshold found in step 2 and possibly updated in later steps, many of the distance calculations can be skipped (remaining white area in the distance matrix). Note that there is no need to calculate the distances in $D_{11}, \ldots, D_{44}$, as the best matching distance is known to be zero.

degrades when a specific pattern is repeated in all series but one, for example due to a labeling error. Not only will this cause a poor result, but the calculation will also take longer to finish due to the higher upper bound.

We demonstrate this is in Figure 5.3, where we compare the timings of four different collections of time series. The first collection consists of 10 identical series, and represents the optimal case for Ostinato. The second collection consists of 10 series of random noise, where we expect a large upper bound due to the lack of patterns. The third collection represents real-world data and was extracted from the REFIT power consumption dataset (aggregate readings from the first house) [3], with each series corresponding to a different time range. As shown later on, this dataset contains multiple repeating patterns. The final collection was the same as the third, but with one series replaced by randomly sampled noise, as to mimic an incorrectly labeled time series. The length of each series was chosen as a multiple of two, and the subsequence length was 1024, as was used in the benchmarks of the original paper [1].

As expected, the timings in Figure 5.3 show a similar trend, but vary greatly between the different types of time series. It is somewhat surprising how the longest runtime does not belong to the random dataset, but rather to the realistic dataset where one series was replaced with noise. In this case, the runtime is about 16 times longer than the optimal case. We suspect this is due to the large number of good matches, all

Algorithm 6: Ostinato algorithm for finding the top-1 consensus motif

Input: series: an array of n series
Input: m: subsequence length
Output: tsIndex: series index of consensus motif
Output: ssIndex: subsequence index of consensus motif
Output: radius: radius of consensus motif

$tsIndex, ssIndex, radius = (-1, -1, +\infty)$
for *s in [0 ... n-1]* **do**
  $sNext = (s + 1) \mod n$
  $MP = MatrixProfile(series[s], series[sNext])$
  $candidateSsIdxs = argsort(MP)$
  for *j in candidateSsIdxs* **do**
    $candRadius = MP[j]$
    if $candRadius \geq radius$ **then**
      break loop
    $otherSeries = [sNext \dots n[\cup [0 \dots s[$
    for *sOther in otherSeries* **do**
      $distances = dist(series[s][j : j + m], series[sOther])$
      $candRadius = max(candRadius, min(distances))$
      if *candRadius* $\geq$ *radius* **then**
        break loop
    if *candRadius* $<$ *radius* **then**
      $tsIndex, ssIndex, radius = (s, j, candRadius)$

of whose distances will be obtained through column-calculations, only to end up with a poor match in the random series, whereas the pure random data will have few good matches overall.

Each of the n full-joins are calculated in $O(l^2)$ using the STOMP [13] algorithm, and each column is calculated using the MASS algorithm in $O(l \log l)$ [14], with n representing the number of series and l the average series length. Overall, we can say the memory complexity is $O(l)$ and the time complexity is $O(n^2 l^2 \log l)$. Note that the subsequence length m does not affect the runtime.

Two variants of Ostinato are mentioned in the original paper [1]. The first variant lets Ostinato find the top-k consensus motifs by tracking the k best results instead of the single best one. Overall, this will result in a higher upper bound and more column-wise calculations in the distance matrix. The second variant can be used to find the top-1 k of n consensus motif, namely the best consensus motif that can be found in a subset of k series. To do this, a full join is calculated and best matches are tracked for each series to $n - k + 1$ other series, rather than one.

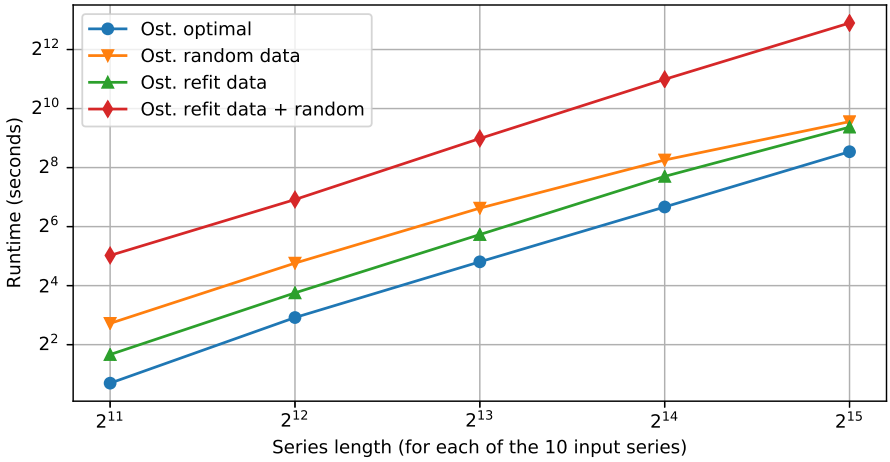


Figure 5.3: Timings for Ostinato on four different types of time series data for subsequence length 1024. Each time, the input consisted of 10 series of the same length. We see that the type of data has a major influence on the runtime, due to the branch-and-bound approach.

As we have shown, Ostinato is an exact and fast algorithm for finding the top-1 (or top-k) consensus motifs. However, there are still some limitations. While it could return intermediate solutions, Ostinato remains a batch algorithm [1], which might be unsuited for applications with strict time constraints. Secondly, the upper bound calculation differs for the top-k and k of n variants, meaning that the user needs to have an idea of what he is looking for before starting a calculation. In the next section, we present an anytime version of Ostinato to calculate not only the top-k motifs, but *all* consensus motifs, including *all* k of n variants, in a single run.

5.5 Finding All Consensus Motif Radii

Finding all consensus motifs rather than the top-k might give further insights into the properties of the series. This is similar as to how the Matrix Profile [9] was originally introduced as a way to find all motifs rather than only the top motif, but has since seen numerous other applications including series segmentation, visualization or rule mining. In a way, it does not make sense to talk about finding all *consensus motifs*, since every subsequence is one. Instead, in this section we will discuss how to find the corresponding radius for each subsequence, i.e. the *radius profile*.

Looking back at Figure 5.1, we could simply calculate the full distance matrix (excluding the diagonal blocks), and calculate the radius for each subsequence one after the other. This approach works and closely resembles the STAMP variant of

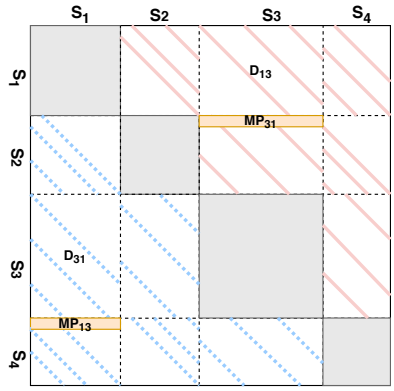


Figure 5.4: Visual representation of the distance calculation for the Anytime Ostinato algorithm. Diagonals are calculated for each pair of series in the upper triangle of the distance matrix (red solid lines), and the distances are reused in the lower triangle (blue dotted lines). Whenever a diagonal is calculated for a pair of series (e.g. D_{13}), the two corresponding matrix profiles are updated (e.g. MP_{13} and MP_{31}). Note that we do not visualize all tracked matrix profiles.

Ostinato used by Kamgar et al. [1] for benchmarking, but is considerably slower than Ostinato. However, there is a better way to approach this problem.

5.5.1 Algorithm

Our solution consists of three parts. First, we can exploit the fact that the distance matrix is symmetric, i.e. the distance at index $[i, j]$ is the same as the distance at index $[j, i]$. This allows us to avoid half of all distance calculations. Secondly, for each subsequence, we track the distance to the best matching subsequence of the other series, i.e. the Matrix Profile for each pair of series. Finally, instead of calculating the distances column-wise (using STAMP or STOMP), we calculate diagonals using the SCRIMP [15] algorithm. A visual representation of our approach is shown in Figure 5.4.

By calculating diagonals, we effectively obtain an anytime calculation for the consensus radius of each subsequence. And, as we will demonstrate below, the anytime estimate on a small subset of the data still gives a representative radius estimate for each subsequence, from which we can also distill the top-k motifs.

Storing the matrix profile for each pair of series (excluding self-joins) provides all information to know the radius of each subsequence for both the top-k consensus motifs as well as the k of n variant. Furthermore, it allows us to resume refining the matrix profiles after looking at intermediate results, which could prove useful for data

exploration purposes. One downside of this approach is the $O(n^2 l)$ memory usage, though this can be reduced to $O(nl)$ if the resumability property is dropped and we do not need the k of n variant, by processing the distance matrix in series-based batches.

As our algorithm is stateful, it consists of an initialization, a calculation, and a result extraction step.

The initialization step is listed in Algorithm 7. We iterate over all pairs of series from the upper triangle of the distance matrix (lines 3 and 4), create a stateful class to calculate both matrix profiles (line 5) and finally store the calculator and output variables (lines 6 till 8). At this point, all matrix profiles contain only infinite distances.

The calculation step is straightforward as shown in Algorithm 8: for each of the SCRIMP calculators (for which we refer to the SCRIMP paper [15]), we calculate diagonals until the desired fraction of all distances has been processed. For each diagonal calculated, the distances are used to update both corresponding matrix profiles.

At any point we can get the all-subsequence consensus radius using Algorithm 9 or the all-subsequence k of n consensus radius using Algorithm 10. The former collects the maximum distance of each subsequence, the latter first sorts all distances per subsequence and extracts the $k-1$ lowest distance.

Algorithm 7: Initialization of the Anytime Ostinato algorithm

```
   Input: series: an array of n series
   Input: m: subsequence length
1  mpCalculators = []
2  mps = array of n empty arrays
3  for s1 in [0 ... n-1] do
4      for s2 in [s1 + 1 ... n-1] do
           /* SCRIMP based matrix profile calculator                    */
5          c = Calculator(series[s1], series[s2], m)
6          mpCalculators.append(c)
           /* Distances of s1 to s2                                     */
7          mps[s1].append(c.mp1)
           /* Distances of s2 to s1                                     */
8          mps[s2].append(c.mp2)
```

Algorithm 8: Calculation step of the Anytime Ostinato algorithm

```
   Input: fraction: fraction of distances to calculate [0 ... 1]
1  for c in mpCalculators do
2      c.updateMPs(fraction)
```

Algorithm 9: Radius calculation for all subsequences in the Anytime Ostinato algorithm

Input: seriesIndex: index of series for which to collect radii
Output: radii: radius for each subsequence of the series

$radii =$ array of $|mps[seriesIndex]|$ zeros
for *mp in mps[seriesIndex]* **do**
    $radii = max(radii, mp)$

Algorithm 10: K of n radius calculation for all subsequences in the Anytime Ostinato algorithm

Input: seriesIndex: index of series for which to collect radii
Input: k: number of series to use for radius, $[2, n[$
Output: radii: radius for each subsequence of the series

```
/* Creates 2D array of shape (n-1, numSubseq)                    */
```

$dists = concat(mps[seriesIndex])$
$sort(dists, alongAxis = 0)$
$radii = dists[k - 1, :]$

5.5.2 Results on the REFIT Dataset

To demonstrate our technique, we use the REFIT dataset [3], which was also used in the original Ostinato paper. We extracted seven time series of the aggregated power consumption of the first house, each a day long (about 12843 data points), from the first week of December 2014. Using Ostinato, we calculated the top-1 consensus motif using a subsequence length of 800 (one and a half hours), which took 94 seconds. Next, we calculated the radius for *all* subsequences using our Anytime Ostinato algorithm for 5, 10, 25, 50 and 100 percent of the data. This took respectively 20, 36, 86, 170 and 328 seconds.

The results are shown in Figure 5.5. At the top, the radius profile for a single day is shown. For visual clarity, we have normalized the distances to a range of zero to one [16] and only show results for the smallest (five percent) and complete calculation. Even though the complete calculation used twenty times as many distance calculations, we see that the results are very similar. At the middle and bottom of Figure 5.5, we see the top-5 consensus motifs for both calculations. Again, we can see that all resulting consensus motifs are very similar. In fact, the top-1 motif found using five percent of all calculations is simply a shifted version of the true top-1 motif. This means that we were able to obtain a representative estimate of the radii for *all* subsequences in less than a fourth of the time it took to obtain the assured top-1 motif using Ostinato.

Similarly we see how the top-2, top-3 and top-5 consensus motif calculated on 5% of the data closely resemble the true top-2, top-3 and top-4 motifs. In all cases, a better

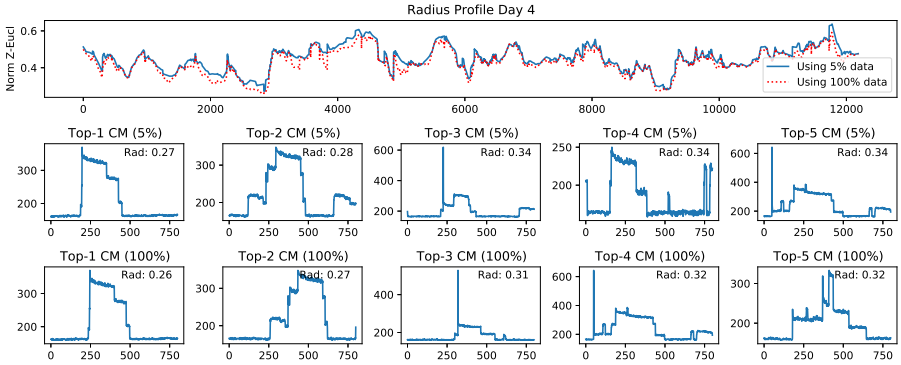


Figure 5.5: Top: Radius profile for the fourth day (which contains the top-1 motif), using either five percent or using all distance calculations. We see that even when calculating only a small fraction, the profile matches very well with the exact profile. Middle & bottom: the top-5 consensus motifs found in the fourth day of data. Each motif corresponds to a local minimum in the radius profile. The top-1 100% motif shown is the true top-1 motif over all considered data. Even using only five percent of all calculations, we found a slightly shifted pattern of this motif. Note that the listed radii for the top-5 5% motifs are upper bounds of the true radius for each pattern.

radius was found by examining the remaining 95% the data. The top-4 5% motif is no longer in the true top-5. Note that none of the 5% motifs are not wrong in the way that they never overestimate the radius. Instead, as more data is processed the radii for those patterns could still decrease or other patterns could be found to change the top-k listing.

Next, let us look deeper at the runtime of our Anytime Ostinato technique. In Figure 5.6, we show the timings for various percentages of calculated distances, and overlay these with the Ostinato timings. Note that unlike Ostinato, the runtime for our algorithm is not affected by the type of data since we do not use a branch and bound method. As we can see, our 100% calculation is still faster than the worst result for Ostinato. For the most representative timing, we should look at the non-modified results for the REFIT dataset. In this case, we can calculate between 10 and 25 percent to get a similar runtime as Ostinato. However, our anytime algorithm returns a result for *all subsequences*, while Ostinato only finds the top-1 consensus motif.

5.6 Finding the Most Common Patterns

Both Ostinato and our improved anytime version are meant to find the patterns that are best conserved over multiple series, i.e. consensus motifs. Similarly, the Matrix Profile can be used to find the best preserved pattern within a single series, i.e. motifs. However, neither technique can be efficiently used to find the top-k best conserved

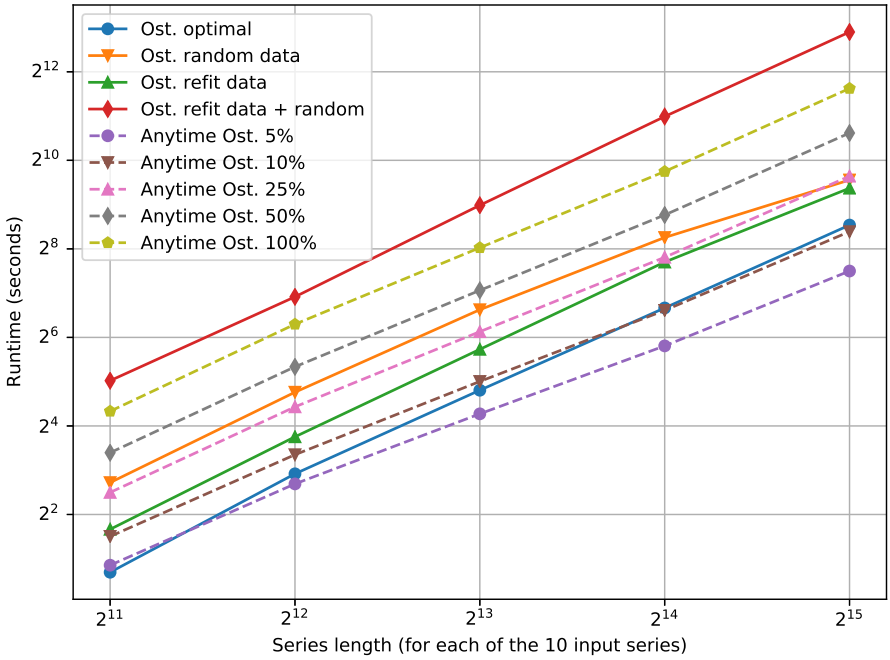


Figure 5.6: Timings for Anytime Ostinato for various calculation percentages, overlayed with the Ostinato results from Figure 5.3. Note that the anytime algorithm is not affected by the type of data. Again, each run was done with 10 series of the same length and subsequence length of 1024. We see how the optimal case of Ostinato takes equally long as an anytime run with 10%, how the normal REFIT data takes about equally long as a 25% run, and how a complete 100% run is still twice as fast as the worst case runtime of Ostinato.

patterns occurring multiple times in a single series. We could call these the intra-series consensus motifs, but to avoid confusion with the multi-series consensus motif, we opt to call these patterns *common motifs* instead.

At this point, there are two ways to formalize these common motifs. One way is to define a distance threshold and find the subsequences that have most matches under this threshold. The second way is to define a number of required matches and determine the subsequence that has the least distance from these matches. We opted for the second interpretation as it allows us to reuse the notion of a motif radius. In other words, for any subsequence s, we find the best k matches within the series and define the radius as the maximum distance of these matches. Then, the top-1 common-k motif is the subsequence for which this radius is minimal. To avoid the issue of trivial matches [9], where nearby sequences have similar distances, we require matches to be at least p indices apart from each other and disregard the vicinity of the original subsequence

s as well. Similar as to how motifs are determined using the Matrix Profile [9], we opted for an exclusion range of $m/2$.

Definition 5.6 (Common-k radius) Given a trivial match exclusion distance p, the common-k radius r of a subsequence $S_{i(0),m}$ with respect to a series S is the distance to the k-th best matching subsequence $S_{i(k),m}$ from S, where each of these best matches $S_{i(1),m}, \ldots, S_{i(k),m}$ and the original subsequence $S_{i(0),m}$ are all at least p apart.

$$\forall a, b \in (0, \ldots, k)^2 : a \neq b \Rightarrow |i(a) - i(b)| \geq p$$

Definition 5.7 (Common-k consensus motif) Given a a series S, the common-k consensus motif of a given length m is the subsequence $S_{i,m}$, such that the common-k radius is minimal.

While we cannot use (Anytime) Ostinato or the Matrix Profile to find common motifs, there is a connection between them. A self-joined Matrix Profile tracks the distance for each subsequence to the best matching subsequence within the same series, this corresponds to the radius of the common-1 motifs, where we only consider a single match. On the other hand, using Ostinato we could find the radius of the top-1 common-k motif if we were to split the series into $k + 1$ subseries where each subseries contained the original subsequence or one of the k matches. Of course, this would requires us to know the top-1 pattern in advance.

In the next subsection, we present our technique to find the radii for all subsequences.

5.6.1 Single Series Radius Profile Algorithm

We show the relation between the distance matrix and the common motif radii in Figure 5.7. Again, the distance matrix is a square where each column or row represents a single subsequence from a series S. Note that we can also apply our technique to find the most common pattern in a set of series by simply appending the series, with a nan marker in between.

For a specific subsequence, we can calculate the distances to each subsequence in S, as shown by the orange column in Figure 5.7. From this distance vector, we iteratively look for the minimum value (= the best match), each time ignoring the subsequences within p positions of any minimum found so far, e.g. $d_0, d_1, d_2, \ldots$ If we ignore the distance from our original subsequence d_0, the other values are the the common-1, common-2, ... radii for this subsequence.

The actual algorithm is straightforward and is shown in Algorithm 11. We iterate over all subsequences (line 4) and extract the subsequence (line 5). Next, we calculate the distance to all subsequences (line 6) using the STOMP algorithm [13]. The STOMP algorithm calculates columns in the distance matrix similar to STAMP, but is faster when calculating neighbouring columns (i.e. complexity is reduced from $O(l^2 log(l))$ to $O(l^2)$). Next, we iterate over all distances from lowest to largest (line 10). We count

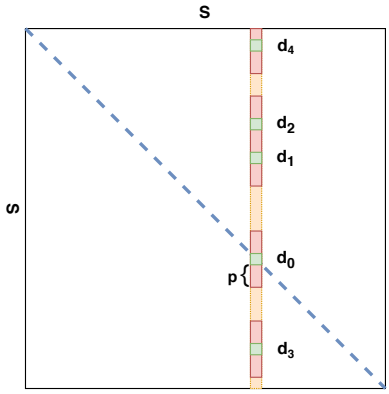


Figure 5.7: Visualization of the link between the distance matrix and common motif radii. We calculate the distance for a single subsequence to the entire series (orange column). From this vector, we iteratively find the lowest distance (= best match), each time ignoring the p subsequences near any previous match (e.g.: $d_0, d_1, d_2, \ldots$). Since we ignore the orignal sequence (d_0), this means the common-1 radius equals d_1, the common-2 radius is d_2 and so on.

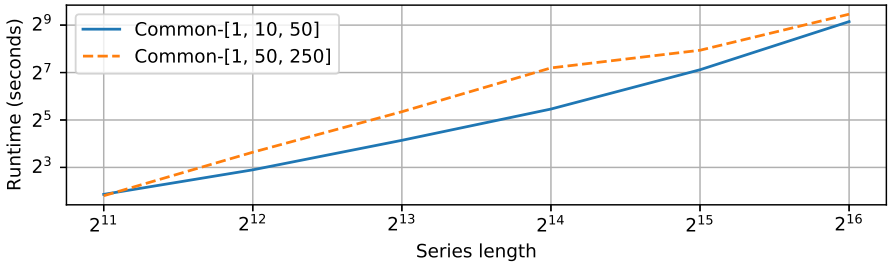


Figure 5.8: Runtimes for the Single Series Ostinato algorithm for two different collections of radius profiles on the PAMAP2 dataset. Note that runtimes can vary slightly for different series due to the inner loop of the algorithm.

the number of non-excluded matches (line 18), mark the exclusion zone in line 19 and skip excluded distances in line 11. Finally, line 13 to 17 record the needed radii and skip to the next subsequence once all radii for the current subsequence are found.

The memory complexity is determined by the size of the output: $O(hn)$, where h is the number of radius profiles to track. The runtime complexity is $O(n^2 l^2 \log l)$ due to the sorting operation that is needed to find the best matches. In Figure 5.8, we show timing benchmarks in function the series length l.

Algorithm 11: Single Series Ostinato Algorithm

Input: series: a 1-D vector
Input: m: subsequence length
Input: kList: k for which to find common-k radii
Input: p: exclusion zone (e.g. m/2)
Output: radii: radius values, each column is one subsequence, each row is one value of k

$numSubseq = |series| - m + 1$
$kList = sort(kList)$
$radii = +\infty$ array of size $(|kList|, numSubseq)$
for *c in [0 ... numSubseq[* **do**
    $subseq = series[c : c + m]$
    $dists = distSTOMP(subseq, series)$
    $matchId = 0$
    $j = 0$
    `/* Indices to sort dists ascendingly */`
    $idxs = argsort(dists)$
    for *i, idx in enumerate(idxs)* **do**
        if *dists[idx]* == $+\infty$ **then**
            continue
        if *matchId* == *kList[j]* **then**
            $radii[j, c] = dists[idx]$
            $j = j + 1$
            if $j > |kList|$ **then**
                break
        $matchId = matchId + 1$
        $dists[max(0, idx - p) : min(numSubseq, idx + p + 1)] = +\infty$

5.6.2 Results on the PAMAP2 Dataset

We demonstrate our technique on the PAMAP2 physical activity dataset [4], which contains accelerometer recordings of participants performing various activities. For demonstration purposes, we limit ourselves to an extract of the y-accelerometer of the first subject performing four different activities, as shown in Figure 5.9.

We calculated the common-k radius profile for values 1, 50 and 250 for subsequence length 100 (i.e. 1 second of recording). These profiles are shown in Figure 5.10. Again, the distances have been normalized.

We see how the walking, nordic walking and running activity have low radii, meaning that the corresponding signals are repetitive, as can be verified in Figure 5.9. One exception here is the high common-250 radius for the running activity, meaning that

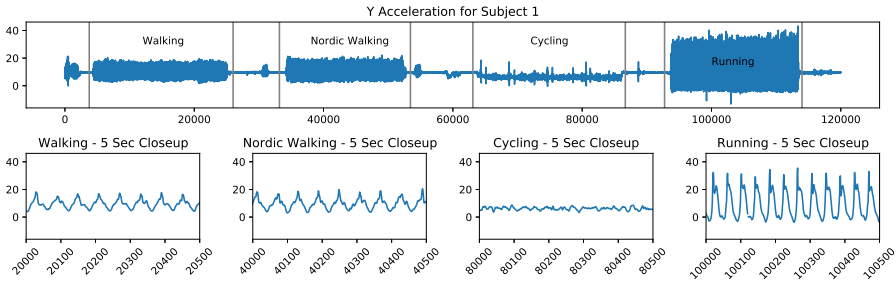


Figure 5.9: Data on which we demonstrate the single series radius profile. This 20 minute extract of the PAMAP2 dataset consists of four activities, separated by transition periods.

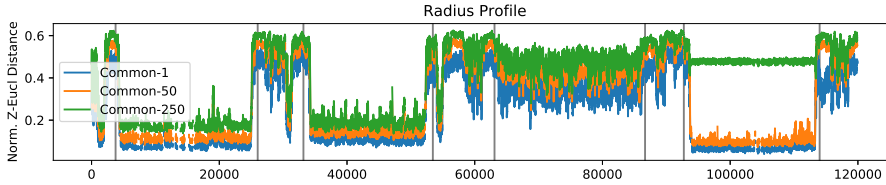


Figure 5.10: Radius profile for the data shown in Figure 5.9 using a subsequence length of 100 (1 second). We see how the radius differs depending on the activity being performed. The more repetitive the signal, the lower the radius. The radii are lowest for the walking and running activities, this means that the patterns are most similar through the data. The cycling activity has a high radius, meaning the subsequences are not repetitive. Gaps in the profile are due to missing data points.

there are less than 250 good matches to each subsequence, which is most likely a result of the repetition interval and the exclusion zone for trivial matches. While the radius profile shows how repetitive each part of the signal is, it gives no direct insight as to which parts are similar to each other. Fortunately, we can use the radius profile to mine for common patterns with some additional work.

5.6.3 Finding the Top-k Common Motifs

The top-1 common-k motif[1] is easily determined: it is the subsequence for which the common-k radius profile is minimal. However, finding the top-k patterns is a little more challenging. Consider we have found the top-1 common-k pattern p_1 and also know its k matching subsequences $m_1, \ldots, m_k$. For the top-2 pattern, we obviously exclude p_1. As $m_1, \ldots, m_k$ are very similar to p_1, it makes sense to also exclude these as well. However, there might be other sequences than $m_1, \ldots, m_k$ that are similar to

[1] To remind the reader, this is the pattern for which the maximal distance to nearest k matches is minimal.

p_1, perhaps having some of their k best matches overlap with $m_1, \ldots, m_k$. Ideally, we want to exclude *all* matches to p_1 that we consider as too similar, though we do not know in advance where exactly this threshold is.

Our approach for finding the top patterns is outlined in Algorithm 12. We start by finding the top-1 pattern (line 2). Next, we calculate the distances between this pattern and all subsequences in the series using MASS (line 5). Whenever a subsequence is a good match (and thus is below a predefined similarity threshold), we exclude both the subsequence as well as the neighbouring subsequences, by changing the radius to infinity (lines 8, 9). This process is repeated until no more motifs can be found.

Algorithm 12: Algorithm to find common-k motifs using the Common-k Radius Profile

Input: series: a 1-D vector
Input: m: subsequence length
Input: radius: radius profile for series
Input: p: exclusion zone (e.g. m/2)
Input: simThresh: similarity threshold
Output: motifIdxs: subsequence indices for all motifs

```
1  motifIdxs = []
2  minIdx = argmin(radius)
3  while radius[minIdx] != +∞ do
4      motifIdxs.append(minIdx)
5      motif = series[minIdx : minIdx + m]
6      dists = dists(motif, series)
7      for i in [0 ... |radius|[ do
8          if radius[i] <= simThresh then
9              radius[max(0, i − p) : min(|radius|, i + p + 1)] = +∞
10     minIdx = argmin(radius)
```

5.6.4 Common Motifs in the PAMAP2 Dataset

Figure 5.11 shows the top-10 common-100 motifs for the PAMAP2 dataset, along with the location of each of the 100 matches in the data. We see that the top-1 and top-5 motif are very similar and correspond to the running activity. The other eight motifs are shared between the walking and nordic walking activity. While the occurrences show how the patterns are distinguishable between these activities, there are overlaps, most noticeable for pattern six. We used a distance threshold of 0.14, which was determined manually after inspecting the first motif. A more lenient threshold of 0.2 resulted in less motifs with a small radius and a less clear distinction between the two walking classes, though results were still representative.

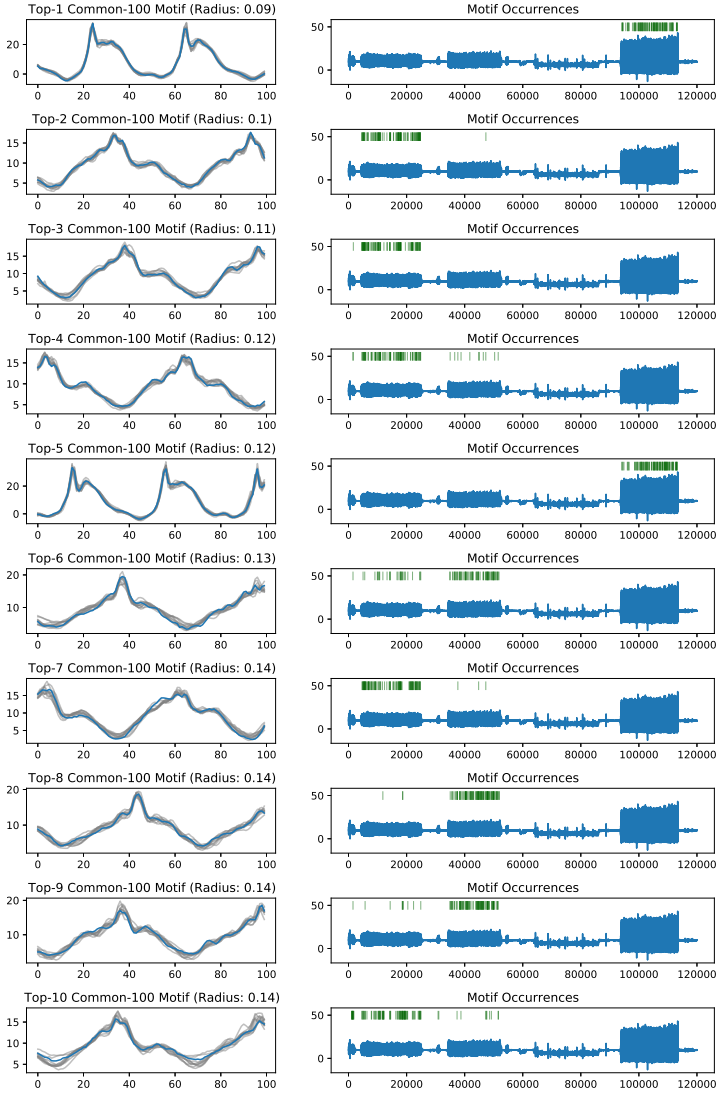


Figure 5.11: The top-10 common-100 motifs for the PAMAP2 dataset. On the left we show the motif (blue) overlayed with the 10 best matches (light gray). On the right, we show the locations of the 100 matches that fall within the radius of the motif. We see that the common motifs originate from the three activities with repetitive behavior. Some motifs appear similar to each other, which can be explained due to slight variations of the activity speed, but also due to an imperfect similarity threshold.

5.7 Conclusion

In this chapter we tackled the question on how to find repeating subsequences in one or more time series containing real values. For this, we have introduced a new time series primitive based on the notion of the consensus motif radius, i.e. the radius profile. This radius profile contains the radius for each subsequence in a series, where the radius is the maximum distance needed to reach a predefined number of best matches.

A first case concerns finding repetition across multiple series. For this we have extended the Ostinato algorithm, which calculates the top-1 radius, to an anytime version that calculates all radii. Our Anytime Ostinato algorithm can calculate an exact profile or make an estimate in a fraction of the time. Even an exact calculation can still be faster than Ostinato for some types of input data. Using the consensus radius profile, one can easily find the k patterns that are most similar over (a subset of) the series collection, i.e. the consensus motifs.

A second case concerns finding well-preserved repetitions in one or more series, irrespective of where they are repeated. Here, we have introduced a new algorithm to find the common-k radius profile, where k represents a predefined number of matches. The profile can be used to visualize the repetitive nature of data and to find the patterns that are repeated in a series, i.e. the common motifs.

Future work remains on finding ways to speed up the calculation of the common-k radius profile. Currently the algorithm depends on column-wise calculations in order to ignore trivial matches, which makes an anytime variant non-trivial. Additionally, future work could look into finding repeating patterns where the length varies between series, e.g. stretched occurrences. Finally, We also foresee further refinements of our techniques applied to specific use cases. For example, time series classification using shapelets could benefit from our techniques to find discriminative patterns.

References

[1] Kaveh Kamgar, Shaghayegh Gharghabi, and Eamonn Keogh. "Matrix profile XV: Exploiting time series consensus motifs to find structure in time series sets". In: *Proceedings - IEEE International Conference on Data Mining, ICDM* 2019-November.Icdm (2019), pp. 1156–1161. ISSN: 15504786. DOI: 10.1109/ICDM.2019.00140.

[2] Lexiang Ye and Eamonn Keogh. "Time Series Shapelets: A New Primitive for Data Mining". In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '09. Paris, France: Association for Computing Machinery, 2009, pp. 947–956. ISBN: 9781605584959. DOI: 10.1145/1557019.1557122. URL: https://doi.org/10.1145/1557019.1557122.

[3] David Murray, Lina Stankovic, and Vladimir Stankovic. "An electrical load measurements dataset of United Kingdom households from a two-year longitudinal study". In: *Scientific data* 4.1 (2017), pp. 1–12.

[4] Attila Reiss and Didier Stricker. "Introducing a new benchmarked dataset for activity monitoring". In: *Proceedings - International Symposium on Wearable Computers, ISWC*. 2012, pp. 108–109. ISBN: 9780769546971. DOI: 10.1109/ISWC.2012.13.

[5] URL: https://sites.google.com/view/mining-patterns-radius-profile.

[6] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. "Frequent pattern mining: current status and future directions". In: *Data Mining and Knowledge Discovery* 15.1 (July 2007), pp. 55–86. ISSN: 1384-5810. DOI: 10.1007/s10618-006-0059-1. URL: http://link.springer.com/10.1007/s10618-006-0059-1.

[7] Timothy L. Bailey, Mikael Boden, Fabian A. Buske, Martin Frith, Charles E. Grant, Luca Clementi, Jingyuan Ren, Wilfred W. Li, and William S. Noble. "MEME Suite: tools for motif discovery and searching". In: *Nucleic Acids Research* 37.suppl 2 (May 2009), W202–W208. ISSN: 0305-1048. DOI: 10.1093/nar/gkp335. URL: https://doi.org/10.1093/nar/gkp335.

[8] Jia-Lien Hsu, Chih-Chin Liu, and A.L.P. Chen. "Discovering nontrivial repeating patterns in music data". In: *IEEE Transactions on Multimedia* 3.3 (2001), pp. 311–325. ISSN: 15209210. DOI: 10.1109/6046.944475. URL: http://ieeexplore.ieee.org/document/944475/.

[9] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. "Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets". In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, Dec. 2016, pp. 1317–1322. ISBN: 978-1-5090-5473-2. DOI: 10.1109/ICDM.2016.0179. URL: http://ieeexplore.ieee.org/document/7837992/.

[10] Yan Zhu, Makoto Imamura, Daniel Nikovski, and Eamonn Keogh. "Matrix Profile VII: Time Series Chains: A New Primitive for Time Series Data Mining". In: *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, Nov. 2017, pp. 695–704. ISBN: 978-1-5386-3835-4. DOI: 10.1109/ICDM.2017.79. URL: http://ieeexplore.ieee.org/document/8215542/.

[11] Shima Imani, Frank Madrid, Wei Ding, Scott Crouter, and Eamonn Keogh. "Matrix Profile XIII: Time Series Snippets: A New Primitive for Time Series Data Mining". In: *2018 IEEE International Conference on Big Knowledge (ICBK)*. IEEE, Nov. 2018, pp. 382–389. ISBN: 978-1-5386-9125-0. DOI: 10.1109/ICBK.2018.00058. URL: https://ieeexplore.ieee.org/document/8588817/.

[12] Zafar Rafii and Bryan Pardo. "REpeating pattern extraction technique (REPET): A simple method for music/voice separation". In: *IEEE Transactions on Audio, Speech and Language Processing* 21.1 (2013), pp. 71–82. ISSN: 15587916. DOI: 10.1109/TASL.2012.2213249.

[13] Yan Zhu, Zachary Zimmerman, Nader Shakibay Senobari, Chin-Chia Michael Yeh, Gareth Funning, Philip Brisk, and Eamonn Keogh. "Matrix Profile II : Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins". In: *2016 {IEEE} 16th International Conference on Data Mining (ICDM)* (2016), pp. 739–748. DOI: 10.1109/ICDM.2016.126.

[14] Abdullah Mueen, Yan Zhu, Michael Yeh, Kaveh Kamgar, Krishnamurthy Viswanathan, Chetan Gupta, and Eamonn Keogh. *The Fastest Similarity Search Algorithm for Time Series Subsequences under Euclidean Distance*. http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html. Aug. 2017.

[15] Yan Zhu, Chin-Chia Michael Yeh, Zachary Zimmerman, Kaveh Kamgar, and Eamonn Keogh. "Matrix Profile XI: SCRIMP++: Time Series Motif Discovery at Interactive Speeds". In: *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, Nov. 2018, pp. 837–846. ISBN: 978-1-5386-9159-5. DOI: 10.1109/ICDM.2018.00099. URL: https://ieeexplore.ieee.org/document/8594908/.

[16] Dieter De Paepe, Diego Nieves Avendano, and Sofie Van Hoecke. "Implications of Z-normalization in the matrix profile". eng. In: *Pattern recognition applications and methods, 8th International Conference, ICPRAM 2019, Prague, Czech Republic, February 19-21, 2019, Revised Selected Papers*. Ed. by Maria De Marsico, Gabriella Sanniti di Baja, and Ana Fred. Vol. 11996. Prague, Czech Republic: Springer, 2020, pp. 95–118. ISBN: 9783030400132. URL: http://dx.doi.org/10.1007/978-3-030-40014-9%5C_5.

Chapter 6

A Complete Software Stack for IoT Time Series Analysis that Combines Semantics and Machine Learning – Lessons Learned from the Dyversify Project

The techniques that have been presented so far have all been demonstrated on well-isolated use cases, as the result of proper scientific methodology. However, many of the challenges faced by companies applying those techniques are related to aspects beyond the low-level technicalities that researchers tend to focus on. Aspects like data management and system integration are examples of such high-level challenges that need to be solved before the solutions of the previous challenges can be put into practice.

One common application of insight mining is anomaly detection on streaming data, as a type of automated system monitoring. Typical challenges here include the setup of scalable and secure systems to ingest and store large data streams, detection of anomalies without labeled data, and the fact that not all users are data scientists. Independently, all of these challenges have several solutions that have been well researched. Data processing can be handled by streaming frameworks or cloud solutions, unsupervised or rule-based techniques can function without anomaly labels, and anomalies can be handled using automated systems or user-friendly dashboards. However, most companies lack the experience in one or more aspects and face difficulties in efficiently combining these systems. Furthermore, companies tend to privatize their experiences and knowledge, meaning that every company is fated to endure the same learn-by-doing process. This means there is also merit in system research, focusing on how different state-of-the-art techniques can be combined into a single system.

Therefor, this chapter describes a system that combines various state-of-the-art techniques and practices, and the experiences and lessons gained from it. This prototype combines semantic knowledge-driven rule-based anomaly detection with unsupervised and supervised pattern case-based event detection techniques, uses a semantic dynamic dashboard to ease user interaction and gather event feedback, and uses a highly scalable deployment system to solve today's shortcomings on system integration research.

My contributions can be summarized as follows:

1. Design of streaming matrix profile based techniques for unsupervised anomaly detection and supervised event detection using feedback from a dynamic dashboard, with support for failure recovery.

2. Contributions to the design of the event messaging format and complete architecture.

3. Contributions to the system integration design, project management and problem resolution.

A Complete Software Stack for IoT Time Series Analysis that Combines Semantics and Machine Learning – Lessons Learned from the Dyversify Project

D. De Paepe, S. Vanden Hautte, B. Steenwinckel, P. Moens, J. Vaneessen, S. Vandekerckhove, B. Volckaert, F. Ongenae, and S. Van Hoecke

Submitted to "Journal of Systems and Software"

Abstract Companies are increasingly gathering and analysing time series data, driven by the rising number of IoT devices and increased bandwidth capabilities. Many works in literature describe analysis systems built using either data-driven or semantic techniques. Little to no works have combined these two branches of research in a complete architecture. Dyversify, a collaborative project between industry and academia, researched several open research questions including the combination of data-driven (non-semantic) and knowledge-driven (semantic) anomaly and event detection, semantification of streaming data, and dynamic dashboarding using semantics. All research was combined in a working prototype as a full, scalable microservice architecture comprising time series ingestion, long term storage, visualisation, user feedback, data semantification and inter-service communication in both semantic and non-semantic formats. This stack was integrated with data sources of two industry partners: Renson, a manufacturer of ventilation systems, and Televic Rail, a train manufacturer. In this chapter, we describe the Dyversify software stack applied to the Renson use case, where we detect events in ventilation metrics in soft-real time. We discuss the system architecture, provide a high-level description for all individual components and relate design choices to usage requirements. We discuss our experiences building and testing the system, including a number of lessons learned. With this work, we shed light on the challenges involved in a full-stack time series processing system that combines machine learning and semantic methods. We think this work may act as a practical reference for parties aiming to set up a similar hybrid system.

6.1 Introduction

Historically, data analytics has always been a core part of research enterprises and large scale tech companies. With the uprise of Internet of Things (IoT), however, more and more enterprises start to gather data, resulting in analytics of these data becoming relevant for a growing range of domains. IoT is expected to reach a market share of over 1000 billion USD and have over 24 billion connections by 2025 [1]. IoT involves a wide range of applications including home automation, traffic control or climate research. But also in factories, machineries are equiped with IoT sensors. Here, data analytics is done to enable predictive maintenance. Predictive maintenance tries to optimize the timing of maintenance operations: early enough to prevent breakdowns but not too early as to minimize maintenance downtime and the cost of replacing healthy compo-

nents. Predictive maintenance use cases are not limited to factories, they apply to any product or service provider aiming for a better customer experience. The market value of predictive maintenance is expected to grow exponentially the coming years, to over 25 billion USD by 2025 [2].

While most research publications solely focus on specific *analysis algorithms*, many other components are needed to create a complete data analysis stack. When a measurement is made, it needs to be *transferred* to an analysis algorithm. In general, data is collected on dedicated machines and processed in bulk, though for some time-critical applications, trivial on-machine calculations may suffice. In either case, *storage* of measurements is useful for later investigations or when new analysis algorithms are retrospectively introduced. Proper *documentation* of all metrics is also vital at this step. When detecting anomalies, the user needs to be *informed* on the details in a timely manner. User *feedback* can also be valuable to optimize the algorithms or visualisations, for example by labeling anomalies for future reference. Finally, these systems should be able to *scale* to allow a rising number of monitored devices and be *resilient* so no relevant events go undetected when the system experiences (partial) failure.

In the imec.icon Dyversify project[1], academia and industry partners collaborated to investigate how machine learning and semantic techniques can be combined for improved anomaly and event detection on time series. We created a working, full-stack prototype combining all of the capabilities listed above. The resulting stack uses a microservice architecture, where different microservice components were developed by different teams. We validated our technology using two use cases with real world data provided by our industrial partners: Renson, Televic Rail and Cumul.io. Televic applied the full-stack solution in the railway sector for monitoring train bogies, Renson applied it to monitor indoor ventilation systems, and Cumul.io validated integration possibilities in their dashboarding platform.

In this article, we discuss the Dyversify full-stack architecture, high level descriptions of the individual components, design choices made and discussion of problems encountered. We focus on the Renson use case, where air quality metrics from consumer owned ventilation units are used for event and anomaly detection. While we do discuss the underlying data analysis techniques, they are not the primary focus of this article. Instead, they serve to give the reader a complete view of the system. As such, we believe this article will be most interesting for parties who seek to build any type of (semantic) data analysis pipeline, from sensor to dashboard, but lack relevant experience in full-stack development.

The remainder of this work is structured as follows. We start by giving background information about the Dyversify project and the Renson use case in Section 6.2, as well as a brief introduction in the Semantic Web domain. An overview of related literature follows in Section 6.3. We discuss the architecture and individual components in Sec-

[1] https://www.imec-int.com/en/what-we-offer/research-portfolio/dyversify

tion 6.4. We finish by discussing our experiences and lessons learned throughout the project in Section 6.5 and conclude in Section 6.6.

6.2 Background information

6.2.1 The Dyversify Project

Dyversify is an imec.icon project, where multi-disciplinary teams from academia and industry work together to do demand-driven research. During the two-year project, a total of seven different teams worked together, four different research teams from IDLab, Ghent University and three industry partners, i.e. Renson (ventilation and shading products), Televic Rail (train components and systems) and Cumul.io (dashboarding platform).

As a research project, each group worked on specific topics including dynamic dashboarding using semantic technologies, adaptive anomaly and event detection for time series using both machine learning and knowledge-driven techniques, and scalable services. All research components were brought together in a working prototype for a full-stack data analysis pipeline that processed real-world data coming from Renson and Televic. This stack includes time series data ingestion and persistence, time series anomaly and event detection, data semantification, visualisation in a dashboard using dynamically configured widget and a user feedback mechanism. The prototype was built as a scalable and resilient microservice architecture.

This article does not cover all research topics of Dyversify in detail, but focuses on the resulting system as a whole. We discuss the stack architecture, give a high level insight into individual components and describe the experiences we learned along the way. To streamline our story, we will everything using the Renson use case.

6.2.2 Renson Use Case

Renson is a Belgian company that produces and sells shading and facade cladding elements, as well as ventilation products. One of their recent products for ventilation is the Healthbox 3.0, a mechanical extraction ventilation unit. The Healthbox regulates the airflow in connected rooms, based on various air quality metrics such as air humidity, CO2 levels or the presence of volatile organic compounds (odours). The ventilation shaft of a room is connected to the Healthbox using a valve outfitted with sensors. Different types of valves exist, each having sensors tailored for a specific type of room.

The Healthbox can be configured to upload the metrics of the valve and Healthbox sensors to Renson, who uses this data for a number of reasons. First and foremost, this data is used to allow users to monitor their indoor air quality using an app. Besides this, Renson aims to use this data to monitor the performance of their units, detecting specific in-house events that affect air quality (e.g. showering), or to flag common installation errors for new customers, such as a valve being used for the wrong type of

room, which could lead to improper ventilation. Each valve, as well as the Healthbox itself, tracks between 8 to 12 metrics. For this use case, aggregated values are sent in 30 second intervals.

6.2.3 Semantic Web

Originating from the initial goal to allow intelligent software agents to perform sophisticated tasks autonomously, Semantic Web is now a wide research domain and covers a wide range of technologies. We limit this introduction to the two topics mentioned in this work: the semantic data model, i.e., the Resource Description Framework (RDF), and semantic reasoning.

In the semantic data model, knowledge is structured as graphs, as opposed to hierarchical or tabular formats common in databases. RDF is an abstract data model that has to be serialized before it can be exchanged. Three common serializations are RDF/XML (XML based), JSON-LD (JSON based) and Turtle (text based). Traditionally, merging (non-semantic) datasets from different sources is cumbersome because different (local) identifiers are used which do not match across datasources or due to differences in the data format (e.g. JSON versus XML). As a result, merging different non-semantic datasets often involves manual operations.

In RDF, instances and the relationships between them are identified using globally unique URIs instead of names or numbers, allowing disambiguation between concepts that share the same name. Ideally, each URI is resolvable and provides (multilingual) documentation about the corresponding concept. When different data sources follow best practices and correctly reuse existing concepts to construct a dataset, these datasets can be merged without additional work. Conceptually, reuse is straightforward: properties are typically bundled in online vocabularies or ontologies and instance identifiers should be reused from an authoritarian source. RDF is a well suited means to exchange data between independent parties due to these properties [3], and has been adopted by instances who actively share data, such as government registries [4] or libraries [5].

A second advantage of using URIs is that data becomes usable for machines since there is no need for human interpretation. This has lead to semantic reasoners, i.e. tools that derive new data by combining existing data with rules of varying complexities. Various semantic reasoners exist, with more expressive reasoners typically being slower [6].

6.3 Related Literature

In this section, we present related literature. We first discuss architectural designs for processing streaming data, which focuses on system design. Next, we discuss literature on stream processing, i.e. how data is used by a system.

6.3.1 Streaming Architectures

Processing streaming data is often discussed in the context of big data analytics and has been applied to many use cases including system monitoring [7, 8], smart cities [9, 10], service monitoring [11, 12, 13] or marketing [14]. The lambda [15] and kappa [16] architectures are two well known high level architectures in this domain. The lambda architecture consists of a fast (sometimes approximated) analysis flow which consumes real-time data and a batch-based flow which uses stored historical data. The lambda architecture is suited for tasks where both real-time updates and historical insights are useful, such as traffic monitoring [17], or when data may be corrected afterwards [14]. The kappa architecture simplifies the lambda architecture by dropping the batch analysis flow. Code reuse is better in kappa architectures because the same flow is used for updating old analytics or calculating newly added analytics.

Both architectures only provide high-level guidelines and are typically implemented using stream processing frameworks such as Apache Storm, Flink or Spark. The concept of these frameworks is simple: a collection of worker nodes is set up and divides the workload in an efficient manner. This approach is both resilient and horizontally scalable as the workload of failed workers is automatically reassigned and new worker machines may be added as needed. Several works exist that compare the features or performance of these frameworks for different use cases [18, 19, 20]. Another work additionally evaluates Kafka Streams and IBM Streams [21]. However, the choice of technology remains a difficult one, as frameworks under active development may improve over time and performance can even be influenced by the size of the messages being consumed [22].

Another common architecture for stream processing is the microservice architecture, where different subtasks of an application are captured into dedicated services. Each service is an independent component that, depending on configuration, interacts with other services to create the full application. Similar to software libraries or modules, microservices are a form of abstraction making them easier to test or maintain and promote reuse. However, they distinguish themselves by being reusable across applications through instance sharing, their ability to scale independently and their ease of integration with other technologies or languages [23, 24]. Again, similar to the lambda and kappa architectures, the microservice architecture also only exists on a high level and leaves much freedom. Practically, they are often implemented as Docker images that interact using REST APIs or configured message brokers such as Kafka or RabbitMQ. Note that a microservice architecture can be combined with a lambda or kappa architecture [17, 25]. Microservice architectures have been used on data streams for e-commerce [24] and intelligent transport systems [10].

Fog computing is another paradigm often associated with stream processing at large scale [26]. Instead of collecting and processing all data streams in a central location, the data streams are partially analyzed or aggregated in advance. This can result in lower bandwidth needs or a faster response detection rate for simple events [9].

Coming from the semantic domain, the MASSIF platform suggests a modular system for IoT services [27]. Data is ingested and a semantic converter is chosen using a data attribute. After conversion the data is added to a semantic data bus from which independent services consume, process and publish data. As MASSIF was mainly meant for low-frequency event data from IoT services, it was later adapted to Streaming MASSIF to handle high frequency data streams [6]. Here, data is filtered by fast operators before being passed to slower, but more expressive reasoners in a process called cascading reasoning. The platform has been used in healthcare [28] and smart (nursing) home [29] prototypes.

While some works do include descriptions of data ingestion [17, 11], visualisation [30, 31], feedback mechanisms [17], deployment [11] or practical lessons learned [30, 14], most do not. In fact, none of these works covers the complete picture despite being very relevant for less experienced readers. We aim to fill this gap with our work, and detail every part of our streaming architecture, as well as valuable experiences we learned during the development and testing process.

6.3.2 Stream Processing

In the context of big data stream processing, two major goals can be discerned: deriving insights or statistics to help humans make informed decisions [14, 32], or anomaly detection about the process generating the data stream [8, 17, 11]. The latter is a common theme in data stream processing that lacks the volume to be seen as big data, with use cases including computer network monitoring [12], water analytics [30] or HVAC fault detection [33]. Many of these approaches extract statistical features, typically over predefined time windows, and use clustering, correlations or other well known machine learning methods to find outliers [34, 33, 12]. Many of these operations are well supported within the aforementioned streaming frameworks, resulting in many works using them for anomaly detection [7, 8, 25, 12].

Pattern based approaches such as the Matrix Profile [35] work by comparing series subsequences rather than using statistical features. The Matrix Profile can be used to find unique subsequences (discords) which can be considered anomalous, as well as matching patterns (motifs).

Semantic streams can be processed using expressive reasoning techniques. These RDF Stream Processors infer additional facts from the data using background data and domain context. These additional facts can then be used to inform users or trigger other actions. Stream reasoning is a broad research domain and more details can be found in the survey by Dell'Aglio et al. [36].

The system described in this work utilises three different data analysis techniques, packaged in four microservices. First, a random forest classifier is used to detect faulty installations by comparing configured versus detected room types. The next two components find unique respectively repeating patterns in incoming signals using the Matrix Profile in an online fashion. Finally, a semantic reasoning component evaluates

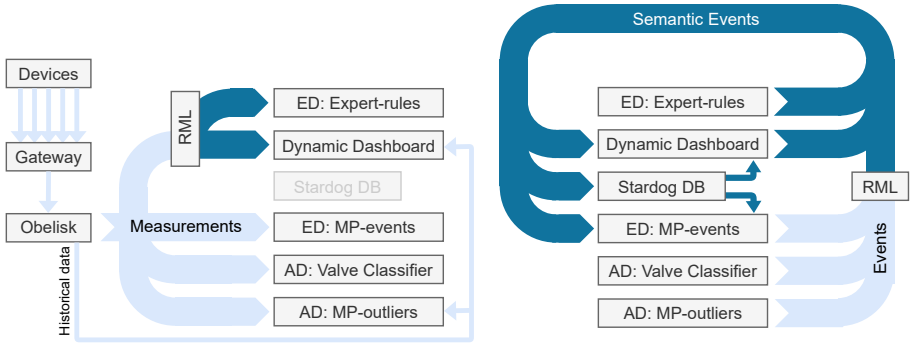


Figure 6.1: Flow of measurement (left) and event data (right) throughout different components. Events are generated by the components after processing measurements. The thick bands represent data flow through Kafka topics, while arrows represent flow through HTTP API calls. Light colors indicate non-semantic (JSON) data; dark colors indicate semantic (JSON-LD) data. Measurements are ingested and persisted by Obelisk. The event detection (ED) and anomaly detection (AD) components process the measurements and produce messages for certain observed patterns. The RML component maps non-semantic data to a semantic format and Stardog is a semantic database that is used to store the semantic events. The dashboard is the only user-facing component and visualises both measurements and detected anomalies/events.

the signal for expert-defined patterns using reasoning over signal windows. These are described in detail in Section 6.4.6.

6.4 Dyversify Architecture

In this section we discuss the full-stack architecture of our system. We start with a high level overview and continue to discuss individual components in the order they are encountered by incoming data.

6.4.1 High-level Overview

To understand the flow of data throughout the system, it is easiest to consider measurement data and event data (created by processing measurements) separately. Both flows are shown in Figure 6.1.

The left of Figure 6.1 shows the flow of measurements throughout the system. Measurements originate from individual Healthbox devices, as described in Section 6.2.2, and are received by a gateway on Renson premises. The Healthboxes have an internal memory to buffer measurements to prevent data loss in case of connectivity issues to the gateway. The gateway forwards all measurements to the *Obelisk* component, which serves as the ingestion system as well as long term storage. Obelisk pushes all measurements in a JSON format to a *Kafka message bus* that is consumed by other

components. Two *event detection (ED)* and two *anomaly detection (AD)* components consume all measurements and output an event message when a specific condition or pattern is observed. The *RML* component is responsible for mapping measurements to their semantic form, which are needed by the expert rules and dashboard components. The *dynamic dashboard* component is intended for user interaction: it can visualise the data streams for all devices, updating as new measurements flow in. We utilise the term *dynamic* as the dashboard allows the user to construct visualisation widgets entirely to his liking while the dashboard reduces choice overload and manual configuration by suggesting visualisation widgets, for selected sensors, based on the sensor's semantic annotation.

The event stream is shown on the right of Figure 6.1. Events are generated by the anomaly and event detection components whenever they find a relevant pattern in the incoming measurements. All components output events in a JSON format which are converted to a semantic format by the RML component, except for the expert-rules component, which works internally using semantics as well. The semantic events are ingested by three different components. The dynamic dashboard triggers (or updates) user notifications whenever an event is ingested. Whenever the MP-events component receives an anomaly event from the MP-outliers component, it will start a new pattern detector to find that specific pattern in new measurements. Finally, the Stardog component is a database that permanently stores the semantic events and can be used by the dashboard and MP-events component to retrieve historical events. Note that the dashboard also outputs events, these are in fact previously ingested events that have been updated by user interaction, such as the labeling of an anomalous pattern.

The precise definition of "anomaly" varies from source to source. Some consider anomalies as *strictly undesired* (e.g. a malfunction), others prefer *outside of normal conditions* (e.g. machine maintenance) or simply as *previously unknown* (e.g. higher power consumption for specific configuration). In this work, we use the term "anomaly" to indicate a previously unknown pattern, which may be undesired (e.g. malfunction) or normal behavior (e.g. opening a window for the first time) and use the term "event" to indicate patterns recognised using some known rule (e.g. a humidity peak in a bathroom indicates a shower). However, when discussing the Kafka message bus, the *event topic* contains both events and anomalies, as hinted by Figure 6.1.

6.4.2 Microservices & Deployment

All components are implemented as microservices using Docker containers and are deployed on a Kubernetes cluster, using Helm as management tool. Services that require data persistence are configured with Kubernetes persistent volumes, so data is not lost if a service is restarted. We used Kubernetes resource management to specify CPU and RAM quota for all services, as to prevent system degradation in the case of misbehaving services, which is not uncommon in development. By using microservices, each team could develop and test their service independent of other teams, this allowed freedom in both planning and choice of technology. In stream processing frameworks

such as Storm, Flink or Spark, it can be more challenging to include techniques that are not available out of the box. The only downside we experienced from using a microservice architecture was the increased effort needed when the format of messages (e.g. events) changed.

All services are deployed on the IDLab Virtual wall[2], a server park of over 350 machines. Practically, the Dyversify project uses four pcgen3 nodes (2x Hexacore Intel E5645 2.4 GHz CPU, 24 GB Ram, 250 GB harddisk) and two pcgen4 nodes (2x 8core Intel E5-2650v2 2.6 GHz CPU, 48 GB Ram, 250 GB harddisk) for all services except for Obelisk. Obelisk is deployed on a dedicated Kubernetes cluster and hardware, but is used by 12 other projects alongside Dyversify. Obelisk is deployed on three nodes with the following hardware: Intel Xeon Silver 4114 2.2 Ghz CPU (40 cores) and 264 GB Ram.

6.4.3 Time Series Ingestion & Persistence: Obelisk

The ingest system is responsible for accepting all sensor data. For a general IoT case, this component needs to be secure, resilient and able to handle parallel, high throughput time series. Long-term data persistence is also required for visualisation and initializing new stream processing models or updating old ones.

As ingestion service, we use Obelisk [37, 38], a scalable platform for building applications on IoT-centric time series data that was developed in-house by Ghent University and imec in the scope of several IoT projects [39, 40]. Obelisk provides a stateless HTTP API for storing and retrieving data and uses authentication based on OpenID and OAuth 2.0, authorization is done by assigning users to project scopes. Internally, Obelisk consists of several distributed microservices that are managed by Kubernetes and are deployed on dedicated hardware with fail-over capacity (see Section 6.4.2).

Obelisk is based on Vert.x, an event-driven and non-blocking JVM framework that can handle high concurrency using a small number of kernel threads. It uses InfluxDB to persist time series and mongoDB for storing metadata and allows rates of up to 6000 measurements per second [38]. Obelisk is foreseen to become open-source and free for non-commercial use or with licensing options for commercial usage by the second half of 2021.

Alternatives to Obelisk include the Open Source FiWare ecosystem. Testing showed scalability issues with passing data to and retrieving data from the FiWare context broker, and only in later versions was historical data aggregation added. These performance issues were also observed by external researchers [41]. Other options include proprietary solutions like Microsoft Azure IoT Hub, Google Cloud IoT Core and the AWS IoT Platform. As these latter solutions come with a substantial vendor lock-in (with no control over the evolution of APIs a.o.), this approach was avoided.

As shown in Figure 6.1, the measurements made by the Healthbox devices are sent to Obelisk. Once received, the measurements are validated, persisted and forwarded to

[2] https://doc.ilabt.imec.be/ilabt/virtualwall/

other components through the Kafka message broker. An example of a measurement message is shown in Listing 6.1, it contains a single humidity measurement from a single device for a specific timestamp.

Listing 6.1: Example of a measurement JSON message. The geohash key specifies information regarding location of the measurement, but was not used in Dyversify.

```
{
  "metricId":"sensor.indoor_relative_humidity.humidity::number",
  "timestamp":1549973410001,
  "timeUnit":"MILLISECONDS",
  "sourceId":"HEALTHBOX3.171030SD0005.2",
  "geohash":null,
  "value":40.052812500,
  "tags":{
    "partner":"renson",
    "context":"icon",
    "project":"dyversify"
  }
}
```

6.4.4 Message Broker: Kafka

The message broker is responsible for the communication between all other components. Together with the ingest system, it should provide enough throughput and be both scalable and fault tolerant. Additionally, our use of time series analysis techniques required that messages remained ordered and were not lost in the case of failure, and messages had to be processed according to the *at-least-once* principle.

We selected Kafka as message broker. Kafka is a high-throughput, low-latency, resilient and scalable message passing platform suitable for handling real-time data feeds. Message streams are organised in *topics* which can be further subdivided into one or more disjoint *partitions*. One topic typically corresponds to one type of message (e.g., measurements or events), whereas partitions divide a topic into logical groups (e.g., groups of machines being monitored). This structure supports Kafka consumers to be horizontally scalable through the use of *consumer groups*. In Kafka, partitions are automatically divided amongst all consumers that belong to the same consumer group, meaning that the workload decreases as more consumer instances are created. This principle is visualised in Figure 6.2.

Kafka also allows for resilient data processing by having data consumers commit their position in each stream at periodic intervals. When a consumer crashes or gets added in response to an increased system load, it can resume processing from the most recent checkpoint. This way, every message is guaranteed to be processed, though some messages may be processed multiple times as a result of a crash. Kafka itself is made resilient through its distributed, redundant deployment.

We utilise Kafka topics to differentiate four types of messages: measurements, semantic measurements, anomalies and semantic anomalies. The measurement topics contain the sensor data as received by the ingest system, while the event topics contain information about the detected events. By using Kafka partitions, any consumer can be made horizontally scalable through consumer groups. For example, we use five

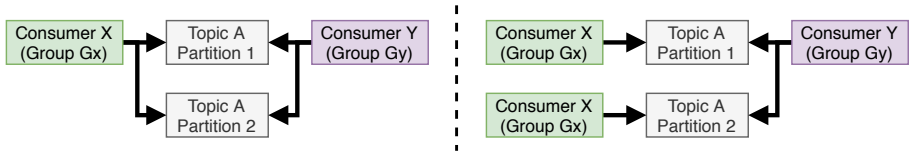


Figure 6.2: Scalability through group based partition assignment in Kafka. Left: One consumer of type X and group Gx and one consumer of type Y and group Gy reading from a topic with two partitions, each consumer is assigned all partitions. Right: A second instance of consumer X and group Gx is added. Kafka divides the partitions over consumers of the same group, lowering the load of both instances.

partitions for the measurement topic, meaning that up to five instances of a single anomaly detection algorithm can process incoming measurements at the same time if required.

Alternative message brokers would be RabbitMQ (a traditional message broker that does not focus on persisting messages), ZeroMQ (a lightweight messaging system which is aimed at high throughput but which lacks advanced features), or Apache ActiveMQ (supports both broker and peer-2-peer messaging). We choose Apache Kafka due to its inherent focus on high reliability and scalability. An alternative to our choice for Kafka would be found in Apache Pulsar, which have similar feature sets, but Kafka was the more mature offering at the time this research was performed, especially when deployed in a Kubernetes cluster environment.

6.4.5 Semantic Conversion: RML

The Dyversify project combines machine learning and semantic technologies, as such, a step to transform data to a semantic format is needed. Specifically, the dynamic dashboard and expert-rule-based components rely only on semantic data. It would be possible to include this conversion for every data outputting service in our system, but this would require teams not familiar with semantic technology to include it in their service. This in turn would have required more testing and development effort for those teams. Furthermore, Dyversify wanted to actively explore how semantic and non-semantic services could work together, so isolating semantic components was not useful in this regard. We opted for a separate conversion service that could be reused by other services. Data throughput is the main requirement for this service.

We selected our in-house RMLStreamer [42] for this component, which is a streaming implementation of the RMLMapper [43], a tool that executes RML (RDF Mapping Language [44]) mappings. Using RML, we can define a mapping from various common input formats, including JSON, XML or CSV, to a semantic format. Note that RML mappings are themselves specified using RDF. The actual conversion is done by the RML-streamer, a tool that executes the mappings defined in RML. Under the hood,

the RMLStreamer uses Apache Flink to distribute the workload to different nodes. We utilised four worker nodes and one supervising node in Dyversify.

The format of measurements was defined by Obelisk and always had the same structure, making mapping straightforward. However, the information in the event messages could differ depending on the origin service. To simplify conversion, we defined a JSON format (discussed in Section 6.4.6) with optional fields for event messages that would be used by all event-generating components. As missing fields are simply ignored in the mapping process, we only need a total of two mappings: one for measurements and one for events.

The mappings were made by a semantic expert using YARRRML [45], a more user-friendly textual format that compiles to RML. While several formats are possible to serialize semantic data, we selected JSON-LD since it allows data extraction using well-known JSON constructs (opposed to SPARQL constructs), which was again useful for lesser semantic-experienced teams and early prototyping. Mappings used the SSN (Semantic Sensor Network) [46], SOSA (Sensor, Observation, Sample, and Actuator) [47] and Folio [48] ontologies, the latter of which was developed specifically for both use cases in Dyversify. Listing 6.2 shows the semantic version of the measurement in Listing 6.1. Note that this example is not fully valid RDF, as the *resultTime* does not specify a valid ISO timestamp, but rather an epoch timestamp. This is due to a limitation in the RMLStreamer, which did not yet support functions in mappings at the time of the Dyversify project. Consumers of semantic events took this quirk into account instead.

Listing 6.2: The semantic JSON-LD equivalent of the measurement shown in Listing 6.1.

```
{
  "@id":"http://dyversify-stack.idlab.be/scopes/icon.dyversify.renson/things/HEALTHBOX3.171030
       SD0005.2/metrics/sensor.indoor_relative_humidity.humidity%3A%3Anumber/observations
       /1549973410001",
  "@type":"http://www.w3.org/ns/sosa/Observation",
  "hasSimpleResult":"40.0528125",
  "observedProperty":"http://dyversify-stack.idlab.be/scopes/icon.dyversify.renson/things/
       HEALTHBOX3.171030SD0005.2/metrics/sensor.indoor_relative_humidity.humidity%3A%3Anumber"
       ,
  "resultTime":"1549973410001",
  "@context":{
    "observedProperty":{
      "@id":"http://www.w3.org/ns/sosa/observedProperty",
      "@type":"@id"
    },
    "resultTime":{
      "@id":"http://www.w3.org/ns/sosa/resultTime",
      "@type":"http://www.w3.org/2001/XMLSchema#dateTime"
    },
    "hasSimpleResult":{
      "@id":"http://www.w3.org/ns/sosa/hasSimpleResult",
      "@type":"http://www.w3.org/2001/XMLSchema#float"
    }
  }
}
```

SPARQL-Generate is an alternative tool for semantic conversion which is based on an extended SPARQL syntax. However, it is less performant than the RMLStreamer [42]. RocketRML [49] is a Node-JS based mapper which also uses RML mappings.

It claims faster timings than the RMLMapper but does not implement the full functionality. Finally, CARML is a converter that struggles with large datasets in batch conversion, but might be suited for streaming cases [50]. Of course, mapping through a custom-made script is also possible, but this ignores the benefits that supporting tools might offer, such as graphical editing or correctness checking [51].

6.4.6 Event/Anomaly Detection

A wide range of different analysis techniques exists, as shown in Section 6.3. The choice of technique often depends on the use case, effort to implement and characteristics of the technique such as throughput or data requirements. Techniques can also be combined, either as single service using boosting or bagging approaches or as multiple, independent services to further improve performance [52]. Some common requirements of techniques include availability of historic data, availability of labeled data and scalability.

In Dyversify, we investigated how to leverage the Healthbox data to detect certain user events as well as anomalous system behavior. Three teams, each with different backgrounds and a focus on different techniques, implemented their techniques as separate microservices. This resulted in four different microservices, as shown in Figure 6.1. We discuss each in detail below.

Each of the detector instances reads measurements from the (semantic) measurement Kafka topic and writes any detected anomaly/event to the (semantic) event Kafka topic. Components that require historic data query Obelisk for measurements or the Stardog database for anomalies and events. Each detected anomaly/event is described with an event identifier, the originating Healthbox, the metric, the time range of the event, the timestamp of the detection, a description and an identifier of the detector.

Listing 6.3 shows a JSON event as generated by the MP-events component. The *id* field contains an identifier for the event. We opted to have each component create HTTP ids, as to simplify the semantic conversion and for easier differentiation of the events when debugging. The *update* field indicates whether the event is an updated version of a previously published event, and is used to update the records in Stardog. The *generatedBy* field provides provenance information. A versioning scheme for the component was foreseen, but did not see real use in Dyversify. The *timestamp* indicates the detection time, and the *anomaly* field gives all information regarding the event, including the occurrence time of the event (in epoch time). Note that the timestamp and anomaly may differ when older data is being processed, which might occur when a processor has been offline for a while. Finally, the *matches* field is specific to the MP-events component, it links the pattern that was matched.

Listing 6.3: JSON representation of an event detected by the MP-events service.

```
{
   "id": "https://gitlab.ilabt.imec.be/dyversify/dyversify-ml-anomaly-detector/KPD/HEALTHBOX3
         .171030SD0005.2/sensor.indoor.CO2.concentration::number/1532217600000/1532217601000",
   "update": false,
   "generatedBy": {
```

```
      "id": "https://gitlab.ilabt.imec.be/dyversify/dyversify-ml-anomaly-detector/ns/known-
          pattern-detector/1",
      "algo": "MatrixProfile",
      "version": 1
   },
   "timestamp": "2019-11-25T15:33:04.333950+00:00",
   "anomaly": {
      "type": [
         "Anomaly",
         "KnownPatternAnomaly"
      ],
      "description": "Pattern similar to Window opened",
      "parts": [
         {
            "thing": "HEALTHBOX3.171030SD0005.2",
            "property": "sensor.indoor.CO2.concentration::number",
            "from": 1532217600000,
            "to": 1532217601000
         }
      ]
   },
   "matches": {
      "id": "https://gitlab.ilabt.imec.be/dyversify/dyversify-ml-anomaly-detector/UAD/
          HEALTHBOX3.171030SD0005.2/sensor.indoor.CO2.concentration::number
          /1532449470000/1532478210000",
      "similarity": -1
   }
}
```


The same information is present in the semantic format shown in Listing 6.5 in 6.6. The JSON-LD version is more verbose due to constraints imposed by the used ontology and the need to include a JSON-LD context, which links the JSON keys to semantic concepts.

Because detector deployments may also crash or be rescaled, they use Kafka to persist their progress periodically. To avoid issues with reporting a single event multiple times when measurements are re-processed after a crash, we generate event identifiers based on the timestamp of the measurements. This way, duplicate detections will have the same identifier and require no special treatment.

6.4.6.1 Anomaly Detection: Valve Classifier

The valve classifier is the most straightforward component. It aims to detect incorrectly installed valves, a common installation error as described in the use case explanation. It does this by using the measurements to determine the most likely room type and comparing this prediction against the type of configured vent.

Techniques evaluated include neural networks, decision trees, random forest and Gaussian processes. Ultimately, we settled for a random forest classifier that discriminates between bedrooms, bathrooms and an unknown type. Features were obtained by subsampling the measurements to 5 minute intervals, collecting a full day of absolute humidity measurements, and generating 7 statistical features. The humidity signal was chosen as it is present in every possible valve type. The classifier was trained on data of 209 bathrooms and 114 bedrooms and utilised oversampling to compensate the data imbalance.

The valve classifier service reads data from Kafka and buffers it until a complete day is ingested, then it uses the pretrained random forest to classify the room and out-

puts an anomaly message if there is a mismatch to the configured room type. As each prediction took only 50 ms per room, scalability was not an issue for this service. The classifier was built in Python using Scikit-learn [53].

6.4.6.2 Anomaly Detection: MP-outliers

This service was created to investigate how the Matrix Profile [35] could be used to detect anomalies in the measurements. These anomalies in turn could indicate anomalous system behavior or specific user actions that affect air quality such as cooking, showering or opening a window.

The Matrix Profile is a technique that analyzes temporal patterns rather than individual values and can be used to find both unique and repeating patterns. It works by sliding a window of a predefined size over the incoming series and calculating the distance to the best matching window in previous data. High distances indicate unique patterns (i.e. discords) and low distances indicate repeating patterns (i.e. motifs).

As measurements enter the component, they are subsampled to 1 minute intervals and passed to a Matrix Profile instance specific for the originating device and metric pair. To ensure each Matrix Profile has reference data, historic data is queried from Obelisk when first encountering a new pair. If no historic data is available (as would be the case with a newly connected device), data is buffered until a predefined number of days is available after which a complete Matrix Profile is calculated. The resulting distances are normalized [54] and checked against a predefined anomaly threshold. Anomalies are not triggered right away when the threshold is passed, but the next distances are also considered and are used to find a local maximum. Once a local maximum is found, it is reported and a cooldown period is initiated, which avoids new reports for the same underlying pattern. This approach avoids multiple anomalies being reported by the same underlying behavior.

The committing of progress to Kafka was somewhat complicated through the use of windows and delayed reporting: an index could only be committed if all preceding data was no longer used for a Matrix Profile window or in a delayed anomaly report. Horizontal scalability is straightforward and only limited by the number of partitions used in the measurements Kafka topic.

In Dyversify, we evaluated this technique for the CO2, temperature and relative humidity measurements, as we expected these signals to be periodic in nature. Near the end of the project, we dropped the temperature signal as this signal produced the least interesting results. We configured each Matrix Profile to track one year of data (to have representative data for all seasons), used a window length of 1 hour (humidity) or 8 hours (CO2), and utilised a method to reduce the adverse effects of noise [54]. Furthermore, we only allowed pattern comparisons for the humidity signal if the standard deviation ratio between both windows was similar (ratio below 1.5). This was needed to discriminate activity patterns from random noise in a period without activity. The MP-outliers component was implemented in Python using the in-house Series Distance Matrix library [55].

Evaluations showed the MP-outliers processes around 2000 (non-filtered) measurements per minute or 1 measurement every 30ms. Since the average Healthbox generates 15 relevant measurements per minute, a single MP-outliers instance can process up to 133 Healthboxes in real-time. Further performance gains can be gained by using coarser subsampling, reducing the amount of reference data or by using GPU implementations of Matrix Profile [56].

6.4.6.3 Event Detection: MP-events

This component assumes that repeating patterns represent repeating system or consumer behavior and uses the Matrix Profile to search for new occurrences of relevant patterns. Relevant patterns are provided by the user interacting with the dashboard, any labeled event will be tracked.

This and the previous component differ in how they use the Matrix Profile. The MP-events component compares a *query pattern* sequence against the incoming measurements, while the previous component performs a so-called *self-join* [35] to look for unique patterns in the incoming measurements. The processing flow and parameters are similar as the previous component, except here a local minimum is used, as we are interested in the best possible match for each query pattern.

The query patterns originate from two sources, as shown in Figure 6.1. The first source is the Kafka event topic, where all detection events are submitted, but because we filter on labeled events, effectively only those of the dashboard are used. Still, by reading from the Kafka topic, rather than creating a direct link to the dashboard, we have a loose coupling that would allow transparent changes in the future. The second source of query patterns is the Stardog database, which is queried using SPARQL. Where Kafka acts as the live event feed, Stardog is the historical repository. Retrieval of historical events is needed in cases where a MP-events instance does not start processing a stream from the start. This can occur due to recovery of a crashed instance or due to reassignment after horizontal scaling. Horizontal scalability is achieved in the same way as the MP-outliers component. Reading from two Kafka topics did pose some challenges, which are discussed in Section 6.5.

As a Matrix Profile instance can only process a single measurement stream, MP-events creates one Matrix Profile per pattern. As MP-events consumes measurements, it forwards each measurement to all Matrix Profile instances tracking that specific stream. This does mean this approach will have scalability issues if the number of patterns would keep growing, though this was not a problem for Dyversify. Evaluations showed that incoming measurements for a single pattern were processed in 15ms.

6.4.6.4 Event Detection: Expert-rules

This service processes the incoming measurements using expert rules that were converted into semantic rules. These rules follow an if-then structure, where both parts contain RDF triples with variables in them. Note that outputs of one rule may be used

as input for a different rule. As semantic reasoners apply rules to collections of triples, rather than streams, we use windowing to reason over the most recent measurements per device/metric combination.

A semantic expert created the rules in cooperation with domain experts. As preparation, experts were asked to complete FMEA (failure mode and effects analysis [57]) and FTA (fault tree analysis [58]) documents, which were converted to semantic rules [59]. An example rule, shown in Listing 6.4, describes the conditions for humid weather, which is used as condition for other rules. Reasoning is done by an internal Stardog database. This database is loaded with the rules, the last 10 measurements and any available metadata of the device. When the reasoner outputs an anomaly, it is submitted to Kafka.

Listing 6.4: A semantic rule that expresses humid weather based on available measurements. Note that some lines were commented out for performance reasons.

```
IF {
    #?h1 a <http://www.w3.org/ns/sosa/Observation> .
    #?h2 a <http://www.w3.org/ns/sosa/Observation> .
    ?h1 <http://www.w3.org/ns/sosa/hasSimpleResult> ?r1 .
    ?h2 <http://www.w3.org/ns/sosa/hasSimpleResult> ?r2 .
    FILTER (?r2-?r1>4) .
    ?h1 <http://IBCNServices.github.io/Folio-Ontology/Folio.owl#hasEpochTime> ?t1.
    ?h2 <http://IBCNServices.github.io/Folio-Ontology/Folio.owl#hasEpochTime> ?t2.
    FILTER(?t1<?t2) .
    #?h1 <http://www.w3.org/ns/sosa/observedProperty> ?o1 .
    #?o1 a <http://IBCNServices.github.io/dyversify/Renson#RelativeHumidity> .
    #?h2 <http://www.w3.org/ns/sosa/observedProperty> ?o2 .
    #?o2 a <http://IBCNServices.github.io/dyversify/Renson#RelativeHumidity> .
}
THEN {
   ?h1 <http://IBCNSercices.github.io/dyversify/model_renson#hasWeather> "humid_weather" .
}
```

As the reasoner only uses static metadata and measurements coming from a single device, this service is horizontally scalable. We deployed 3 instances of this service, where each instance ran on a 10 CPU core node with 10 Stardog instances. Evaluations showed that on average, a single instance processes bedroom measurements in 320ms and bathroom measurements in 730ms. This difference in timings can be attributed to the complexity of the applicable rules. Further details can be found in a dedicated work [48].

6.4.7 Semantic Database: Stardog

The semantic database is needed to persist all relevant semantic data. This includes all reported events, as well as available metadata about all Healthbox devices. Notably, the semantic measurements are not persisted, as semantic databases are simply less optimized for storing large collections of time series data.

Stardog was selected based on previous experiences of the semantic team as well as practical considerations towards the project partners regarding licensing. One advantage of Stardog is that it is not just a RDF store, but also a Graph DBMS, which facilitated data isolation for the industrial partners. Some alternatives include Virtuoso, which has a more complex setup, or Apache Jena, which is limited to Java.

6.4.8 Dynamic Dashboard & Feedback

The last step in the system is a dashboard where users can visualise the various metrics and investigate any event in detail. Dashboards need to balance between flexibility and ease of use. Many dashboards require the user to specify and configure the widgets of each desired visualisation. While wizards somewhat ease this task, they can still be burdensome when the number of devices or metrics grows. Dashboards are also a location to gather user input or feedback, since they are typically the only point of interaction.

In Dyversify, we further developed a semantics-driven dynamic dashboard [60, 61]. This dashboard suggests suitable visualisations by reasoning over the semantic descriptions of the sensors and supported visual widgets. The dynamic reasoning component was also developed as an independent microservice, so it could be used to suggest visualisations in other, commercial dashboards like the one from our industry partner cumul.io, as well. When an event is selected by a user, the dashboard automatically selects and configures a set of widgets that are suited to investigate it. Manual configuration of widgets is also possible.

The dashboard is implemented as an Angular application interacting with three other microservices: a data streamer, broker and gateway. These are written in Kafka Streams, Django REST framework and AIOHTTP respectively. The data streamer acts as the data access point to which dashboard widgets subscribe for the visualised metrics. The relevant metrics are then filtered from Kafka and forwarded to the widgets. Note that the streamer service is foreseen to be integrated into the gateway in future versions of the dashboard. The broker stores and provides user state (e.g., dashboard layout), keeps track of the available metrics and houses the semantic reasoner that suggests widgets based on the semantic description of visualised data. Finally, the gateway acts as an API to retrieve the metadata and assets for dashboard widgets.

Only the dashboard microservices in our stack are exposed to the public. This way the dashboard can provide real-time updates, while all other services of our system remain in an isolated, secure setting.

Users can validate or remove events flagged by the system and can assign labels to events. These interactions act as feedback for the other components of the system. After user interaction, the label and semantic type of the event are updated and the updated event is resubmitted to the event Kafka topic where it can be picked up again by other services.

The dashboard is described in more detail in other works. Vanden Hautte et al. describe the dashboard in the context of Dyversify, and include a demo movie[3] in their work [61]. Moens et al. describe the interaction between the dashboard and Obelisk in more detail for a different, industrial IoT case [40].

[3] https://www.dropbox.com/s/lhg7v5wz09ffvun/Dyversify%20demo.avi?dl=0

6.4.9 Monitoring

Ensuring components are behaving as expected is of vital importance both during development and in production settings. Monitoring systems give insight into the underlying system to ensure this is the case. Even if different teams do not know the specifics of components maintained by other teams, it is straightforward to interpret metrics such as CPU usage, message throughput or Kafka messages to quickly validate components are well behaving.

As monitoring solution, we compared the TICK stack and Prometheus. TICK (Telegraf, InfluxDB, Chronograf and Kapacitor) is a set of open-source tools that can be combined together or used separately to collect, store, visualise and manipulate time series data. It is developed by InfluxData, mostly known for its time series database InfluxDB. The second candidate, Prometheus is fully open-source monitoring solution inspired by Google Borg Monitor [62]. It was initially developed by SoundCloud and later donated to the Cloud Native Computing Foundation which also houses Kubernetes, fluentd, Helm, Envoy and others. Each Prometheus server is standalone, ensuring correct functioning of Prometheus even when other parts of the infrastructure are broken. While both monitoring systems have the same capabilities, we chose Prometheus as monitoring solution because it collects metrics in a pull-based manner, has a more streamlined data store, has a less verbose query language and found it easier to configure for small setups.

Monitoring solutions such as Prometheus collect metrics which can be broadly divided into three categories: service metrics (e.g. input or error rate), resource metrics (e.g. CPU usage or network I/O) and events (e.g. alerts or configuration changes). Some metrics like cache hits or database locks cannot be put in any of these categories, but may still prove to be useful in representing the operability of the system. Existing methodologies can prove good starting grounds for deciding which metrics to collect. The USE-method [63] is a system-agnostic methodology that focuses on the resource utilisation, saturation and errors. Another methodology comes from the Google Site-Reliability Engineering (SRE) team, which formulated four golden signals: latency, traffic, errors and saturation [62]. Finally, the RED-method (rate, errors, duration) was created by Tom Wilkie, a former Google SRE employee, and focuses on microservice monitoring.

All of the metrics above were made available in a Grafana dashboard. Still, we found the Kafka consumer lag and latest Kafka messages the most useful metrics during service development. We also experimented with a fully automatic rule-based scaling mechanism for the microservices, based on the number of incoming messages, the number of consumed messages, the average consumer lag and the derivation thereof. This approach proved successful, but it took some time before the system reached a stable state after rescaling, so specific configuration for every use case would be required in order to be truly efficient.

6.5 Lessons Learned

After this overview of the architecture of the Dyversify stack, we now discuss lessons we learned while developing and testing our stack.

6.5.1 Scalability Requirements

Scalability is achieved by up- or downscaling the instances of components based on the workload, and having each instance process a subset of the incoming data. When these components and their data flow adhere to certain guidelines, the complexity can be greatly reduced.

First, each instance should be able to work independently, so there is no need for synchronization between all instances. Ideally, the input data also forms disjoint logical groups. In our proof-of-concept, we wanted to perform event detection across all ventilated rooms per building, so the data had to be partitioned in such a way that all streams originating from a single building were assigned to the same Kafka partition.

Here, we encountered an issue for the MP-events component which also uses user feedback. The event topic contains the patterns that need to be tracked, while the measurement topic contains the actual time series being tested. The issue originates from the need to process two Kafka topics at the same time (with the event topic having priority), which is not supported by Kafka. We solved this by interweaving the reads from both topics, though this introduces a timing-dependent non-deterministic behavior which might be undesired for some use cases.

A second requirement for scalability is that each instance should have a short startup and perform a clean and fast shutdown when it receives a termination signal. Here, we experienced many problems with the MP-outliers service, which determines anomalies by referencing one year of historic data. Because of the way different sensors are interwoven on the measurement topic, all relevant historic data will be loaded before the detector achieves normal operating speed. As fetching historic data could take several minutes for a single sensor, this resulted in an effective startup time of 30 to 60 minutes. In cases where data fetching took exceptionally long, Kafka marked the unresponsive consumer as a crashed instance and removed it. Unfortunately, this caused a re-initialization cascade as all data partitions were reassigned to the remaining nodes, effectively re-triggering the issue. We solved the cascade problem by using manual partition assignment, and starting the instances one at a time to avoid all instances querying Obelisk at the same moment. The long data fetches could be solved by caching the historic data sets on a shared drive, avoiding the need for lengthy data retrieval queries.

6.5.2 Setting up a Complex Microservice-based Backend

There is a need for a number of essential enablers during multi-team collaborative R&D on complex microservice-based systems such as the one described here. For

one, in-depth monitoring of individual service endpoints in terms of e.g. load, resource consumption and response times allowed us to rapidly gain insights in the dynamics of the service backend. This aided in identifying erroneous or misbehaving service instances e.g. due to overconsumption of memory. In Dyversify, we chose to employ Prometheus as a monitoring and alerting toolkit, monitoring all Dockerised services deployed on Kubernetes, with metrics visualised on Grafana dashboards and push-based developer alerting in case a back-end issue was detected. Service mesh technologies (e.g. Istio) can provide additional monitoring, specifically network metrics, by adding *sidecars* or proxies to each service that is deployed. However, due to the constraints of our architecture, i.e. high speed IoT data, we opted not to implement this technology due to the impact on performance.

A second important lesson learned was that when multiple development teams deploy services on the same container orchestration infrastructure (e.g. Kubernetes), it is important to enforce infrastructural bulkheads (fixed resource constraints per team supplying services). This follows the *embrace failure* principle, as deployment of an erroneous service version (which can and will happen) cannot escape the confines of the bulkhead and therefore has a more limited impact on other well-behaving service instances.

Thirdly, as data in our architecture was sent over Kafka, it became clear that input validation of posted messages should be mandatory. To give an example: faulty sensors were at times emitting NaN (Not a Number) as sensor data value. As consuming services expected these values to be of floating point type, these messages remained unconsumed and cluttered the different topics. Dedicated alerting or data offloading strategies for data that does not adhere to the expected input should therefore be installed.

6.5.3 Early Testing for Library Limitations

Any high-level software component relies on libraries made by third parties for certain functionality. The availability or maturity of these libraries may differ between programming languages and should be considered during the design of any component. Unfortunately, desired functionality may not be fully known in advance, or limitations of certain libraries may not be apparent through documentation.

For us, the choice of programming language was mainly based on the expertise of each team, but we did not foresee problems related to libraries. Still, two library related problems arose during the course of the project. First, we encountered problems with the Python Kafka client implementation. One issue entailed internal timeouts leading to costly consumer rebalances and were due to the inner workings of the client library. Later versions of the library fixed this issue, months after we reported the issue. Another issue was missing functionality to avoid the previously mentioned initialization cascade, which is available in the Java Kafka client, but not in the Python client. Both problems required workarounds which cost one to two weeks of work to get right. The second library problem was related to the RMLStreamer, and was due to

an undocumented interaction between Kafka and Apache Flink (used internally by the RMLStreamer). Here, we learned the hard way that Flink does not follow the consumer group semantics of Kafka. This caused input partitions to be processed zero or multiple times, leading to both duplicate and missing semantic messages. Tests had missed this problem because the testing environment only used one data partition, where the production environment used multiple. This highlights the need to test early during development and to match the testing and production environment as best as possible.

6.5.4 Semantic Microservice Communication

The independence of components in microservice architectures imposes a need for well defined message formats. Once established, changes may affect multiple other components and should be avoided. A known solution is for each microservice to provide versioned, well-defined contracts to clients. New versions incorporate changes and are used in parallel with older, deprecated contracts during a grace period.

Our stack uses two types of messages in the Kafka topics: measurements and events. Both types have a JSON and semantic model, and a mapping is possible from one format to the other. The measurement format is straightforward, as it is only outputted by the ingest system. The event format consists of a single specification with optional fields that are filled depending on the producer.

While semantics are well suited for streamlining communication from different sources, we experienced no benefit from using semantic messages over plain JSON for microservice communication. As mentioned in the introduction, semantic graphs can be serialized in multiple ways. Even in the JSON-LD serialization, there are multiple formats to represent the same data. This means developers either have to use a technique called *JSON-LD framing* to transform it to a desired JSON structure, or load the data into a triple store so it can be queried using SPARQL. While framing and SPARQL are well supported, they are unfamiliar to developers and introduce an extra level of complexity. Because the microservice environment is isolated and the format of messages is well defined, the streamlining value of semantics is somewhat lost and we are left with a complex data container.

Another aspect to take into account is the verbosity of semantic messages. Because URIs are used to identify concepts and relations, and values should be specified explicitly (i.e. they should not be extracted from URIs), semantic messages are typically longer than a pure JSON representation. Methods exist to reduce verbosity, such as the use of a context which maps (long) URIs to short, simple string keys, as is done in Listing 6.2 and Listing 6.5. However, this has no effect for short messages such as measurements or events because the mapping context has to be included in the message as well. The JSON-LD context can also be included as an external link in a HTTP header. Conceptually, this requires clients to validate the context for every message they process, incurring a processing overhead instead. For us, semantic messages were 1.5 to 6 times larger than the JSON messages.

6.6 Conclusion

In this chapter we discussed the architecture designed and validated within the Dyversify project to create a working, scalable and resilient proof-of-concept software stack that combines machine learning and semantic technologies. This stack is used for event detection on time series data stemming from internet-enabled ventilation devices. The stack was developed in cooperation with three industry partners, and validated using real data from train bogie monitoring and ventilation monitoring systems, though this chapter only discusses the former.

We explain how data ingest, storage, transfer, processing, visualisation and capturing of user feedback are performed by interacting, independent microservices. Event and anomaly detection is performed by multiple machine learning and semantic expert-based components, whose output is unified to a semantic format. A dashboard uses semantic sensor descriptions to dynamically generate visualisations for events with only limited human intervention. We also discuss our system monitoring approach and considerations to guarantee scalability. We believe this work can provide a valuable starting reference for parties considering IoT anomaly or event detection combining both data-driven and semantic technologies, as well as looking at full-stack design.

Research continues on all techniques mentioned in this chapter, as well as their integration. Anomaly detection methods for IoT still suffer from the varying deployment situations and uncertainty on what should be considered anomalous. Incorporating the context of sensors into anomaly detection might prove useful. Monitoring of distributed systems faces similar challenges, since the normal system behavior changes as services are added or replaced on the network. Finally, more fine grained models might also be useful for recommending anomaly visualisations, since users will have different focuses based on their company role or expertise.

Appendix: Semantic event

Listing 6.5: JSON-LD representation of the event from Listing 6.3.

```
{
   "@graph": [
      {
         "@id": "http://dyversify-stack.idlab.be/scopes/icon.dyversify.renson/things/
               HEALTHBOX3.171030SD0005.2/metrics/sensor.indoor.CO2.concentration%3A%3Anumber",
         "isObservedBy": "http://dyversify-stack.idlab.be/scopes/icon.dyversify.renson/things/
               HEALTHBOX3.171030SD0005.2"
      },{
         "@id": "http://dyversify-stack.idlab.be/scopes/icon.dyversify.renson/things/
               HEALTHBOX3.171030SD0005.2/metrics/sensor.indoor.CO2.concentration%3A%3Anumber/
               observations/1532217600000",
         "resultTime": "1532217600000"
      },{
         "@id": "http://dyversify-stack.idlab.be/scopes/icon.dyversify.renson/things/
               HEALTHBOX3.171030SD0005.2/metrics/sensor.indoor.CO2.concentration%3A%3Anumber/
               observations/1532217601000",
         "resultTime": "1532217601000"
      },{
         "@id": "http://example.com/procedure_bn/2019-11-25T15%3A33%3A04.333950%2B00%3A00",
         "label": "MatrixProfile"
```

```
    },{
      "@id": "http://example.com/stimulus/HEALTHBOX3.171030SD0005.2/sensor.indoor.CO2.
          concentration%3A%3Anumber/1532217600000/1532217601000",
      "@type": "http://www.w3.org/ns/ssn/Stimulus",
      "fromObservation": "http://dyversify-stack.idlab.be/scopes/icon.dyversify.renson/
          things/HEALTHBOX3.171030SD0005.2/metrics/sensor.indoor.CO2.concentration%3A%3
          Anumber/observations/1532217600000",
      "observedProperty": "http://dyversify-stack.idlab.be/scopes/icon.dyversify.renson/
          things/HEALTHBOX3.171030SD0005.2/metrics/sensor.indoor.CO2.concentration%3A%3
          Anumber",
      "toObservation": "http://dyversify-stack.idlab.be/scopes/icon.dyversify.renson/things
          /HEALTHBOX3.171030SD0005.2/metrics/sensor.indoor.CO2.concentration%3A%3Anumber/
          observations/1532217601000"
    },{
      "@id": "https://gitlab.ilabt.imec.be/dyversify/dyversify-ml-anomaly-detector/KPD/
          HEALTHBOX3.171030SD0005.2/sensor.indoor.CO2.concentration::number
          /1532217600000/1532217601000",
      "@type": [
        "http://IBCNServices.github.io/Folio-Ontology/Folio.owl#Anomaly",
        "http://IBCNServices.github.io/Folio-Ontology/Folio.owl#KnownPatternAnomaly"
      ],
      "description": "Pattern similar to Window opened",
      "resultTime": "2019-11-25T15:33:04.333950+00:00",
      "usedProcedure": "https://gitlab.ilabt.imec.be/dyversify/dyversify-ml-anomaly-
          detector/ns/known-pattern-detector/1",
      "wasOriginatedBy": "http://example.com/stimulus/HEALTHBOX3.171030SD0005.2/sensor.
          indoor.CO2.concentration%3A%3Anumber/1532217600000/1532217601000",
      "update": "false",
      "metricId": "sensor.indoor.CO2.concentration::number",
      "thingId": "HEALTHBOX3.171030SD0005.2"
    },{
      "@id": "https://gitlab.ilabt.imec.be/dyversify/dyversify-ml-anomaly-detector/ns/known
          -pattern-detector/1",
      "@type": "http://www.w3.org/ns/sosa/Procedure",
      "specializationOf": "http://example.com/procedure_bn/2019-11-25T15%3A33%3A04.333950%2
          B00%3A00"
    }
  ],
  "@context": {
    "metricId": {"@id": "https://idlab-iot.tengu.io/api/v1/vocabulary/metricId"},
    "thingId": {"@id": "https://idlab-iot.tengu.io/api/v1/vocabulary/thingId"},
    "label": {"@id": "http://www.w3.org/2000/01/rdf-schema#label"},
    "description": {"@id": "http://purl.org/dc/terms/description"},
    "update": {"@id": "https://idlab-iot.tengu.io/api/v1/booleans/update"},
    "resultTime": {
      "@id": "http://www.w3.org/ns/sosa/resultTime",
      "@type": "http://www.w3.org/2001/XMLSchema#dateTime"
    },"usedProcedure": {
      "@id": "http://www.w3.org/ns/sosa/usedProcedure",
      "@type": "@id"
    },"wasOriginatedBy": {
      "@id": "http://www.w3.org/ns/ssn/wasOriginatedBy",
      "@type": "@id"
    },"isObservedBy": {
      "@id": "http://www.w3.org/ns/sosa/isObservedBy",
      "@type": "@id"
    },"observedProperty": {
      "@id": "http://IBCNServices.github.io/Folio-Ontology/Folio.owl#observedProperty",
      "@type": "@id"
    },"fromObservation": {
      "@id": "http://IBCNServices.github.io/Folio-Ontology/Folio.owl#fromObservation",
      "@type": "@id"
    },"toObservation": {
      "@id": "http://IBCNServices.github.io/Folio-Ontology/Folio.owl#toObservation",
      "@type": "@id"
    },"specializationOf": {
      "@id": "http://www.w3.org/ns/prov#specializationOf",
      "@type": "@id"
    }
  }
}
```

References

[1] *eSIM Technology to Spur IoT Connections in APAC by 2025*. Accessed Jan 2021. URL: https://www.forest-interactive.com/newsroom/esim-technology-to-spur-iot-connections-in-apac-by-2025/.

[2] *Grand View Research - Predictive Maintenance Market Size Worth $28.24 Billion By 2025*. Accessed Jan 2021. URL: https://www.grandviewresearch.com/press-release/global-predictive-maintenance-market.

[3] Christian Bizer, Tom Heath, and Tim Berners-Lee. "Linked data: The story so far". In: *Semantic services, interoperability and web applications: emerging concepts*. IGI Global, 2011, pp. 205–227.

[4] Raf Buyle et al. "Raising interoperability among base registries: The evolution of the Linked Base Registry for addresses in Flanders". In: *Journal of Web Semantics* 55 (2019), pp. 86–101. ISSN: 1570-8268. DOI: 10.1016/j.websem.2018.10.003.

[5] Corine Deliot. "Publishing the British national bibliography as linked open data". In: *Catalogue & Index* 174 (2014), pp. 13–18.

[6] Pieter Bonte, Riccardo Tommasini, Emanuele Della Valle, Filip De Turck, and Femke Ongenae. "Streaming MASSIF: cascading reasoning for efficient processing of iot data streams". In: *Sensors* 18.11 (2018), p. 3832.

[7] Mohiuddin Solaimani, Mohammed Iftekhar, Latifur Khan, Bhavani Thuraisingham, and Joey Burton Ingram. "Spark-based anomaly detection over multi-source VMware performance data in real-time". In: *2014 IEEE Symposium on Computational Intelligence in Cyber Security (CICS)*. IEEE. 2014, pp. 1–8.

[8] Mohiuddin Solaimani, Latifur Khan, and Bhavani Thuraisingham. "Real-time anomaly detection over VMware performance data using storm". In: *15th Int. Conf. on Information Reuse and Integration*. IEEE. 2014, pp. 458–465.

[9] Bo Tang, Zhen Chen, Gerald Hefferman, Tao Wei, Haibo He, and Qing Yang. "A Hierarchical Distributed Fog Computing Architecture for Big Data Analysis in Smart Cities". In: *Proc. of the ASE BigData & SocialInformatics*. Kaohsiung, Taiwan: Association for Computing Machinery, 2015. ISBN: 9781450337359.

[10] Suriya Priya R Asaithambi, Ramanathan Venkatraman, and Sitalakshmi Venkatraman. "MOBDA: Microservice-Oriented Big Data Architecture for Smart City Transport Systems". In: *Big Data and Cognitive Computing* 4.3 (2020), p. 17.

[11] Laura Rettig, Mourad Khayati, Philippe Cudré-Mauroux, and Michał Piorkówski. "Online Anomaly Detection over Big Data Streams". In: *Applied Data Science: Lessons Learned for the Data-Driven Business*. Ed. by Martin Braschler, Thilo Stadelmann, and Kurt Stockinger. Springer International Publishing, 2019, pp. 289–312. ISBN: 978-3-030-11821-1. DOI: 10.1007/978-3-030-11821-1_16.

[12] S. Zhao, M. Chandrashekar, Y. Lee, and D. Medhi. "Real-time network anomaly detection system using machine learning". In: *2015 11th International Conference on the Design of Reliable Communication Networks (DRCN)*. 2015, pp. 267–270. DOI: 10.1109/DRCN.2015.7149025.

[13] M. S. Parwez, D. B. Rawat, and M. Garuba. "Big Data Analytics for User-Activity Analysis and User-Anomaly Detection in Mobile Wireless Network". In: *IEEE Transactions on Industrial Informatics* 13.4 (2017), pp. 2058–2065. DOI: 10.1109/TII.2017.2650206.

[14] Fangjin Yang, Gian Merlino, Nelson Ray, Xavier Léauté, Himanshu Gupta, and Eric Tschetter. "The RADStack: Open source lambda architecture for interactive analytics". In: *Proceedings of the 50th Hawaii Int. Conf. on System Sciences*. 2017.

[15] Nathan Marz and James Warren. *Big Data: Principles and best practices of scalable real-time data systems*. New York; Manning Publications Co., 2015.

[16] Jay Kreps. "Questioning the lambda architecture". In: *Online article* (2014).

[17] P. Ta-Shma, A. Akbar, G. Gerson-Golan, G. Hadash, F. Carrez, and K. Moessner. "An Ingestion and Analytics Architecture for IoT Applied to Smart City Use Cases". In: *IEEE Internet of Things Journal* 5.2 (2018), pp. 765–774. DOI: 10.1109/JIOT.2017.2722378.

[18] Martin Andreoni Lopez, Antonio Gonzalez Pastana Lobato, and Otto Carlos MB Duarte. "A performance comparison of open-source stream processing platforms". In: *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2016, pp. 1–6.

[19] Jeyhun Karimov, Tilmann Rabl, Asterios Katsifodimos, Roman Samarev, Henri Heiskanen, and Volker Markl. "Benchmarking distributed stream data processing systems". In: *34th Int. Conf. on Data Eng. (ICDE)*. IEEE. 2018, pp. 1507–1518.

[20] Valerio Persico, Antonio Pescapé, Antonio Picariello, and Giancarlo Sperlí. "Benchmarking big data architectures for social networks data processing using public cloud platforms". In: *Future Generation Computer Systems* 89 (2018), pp. 98–109. ISSN: 0167-739X. DOI: 10.1016/j.future.2018.05.068.

[21] H. Isah, T. Abughofa, S. Mahfuz, D. Ajerla, F. Zulkernine, and S. Khan. "A Survey of Distributed Data Stream Processing Frameworks". In: *IEEE Access* 7 (2019), pp. 154300–154316. DOI: 10.1109/ACCESS.2019.2946884.

[22] Ben Blamey, Andreas Hellander, and Salman Toor. "Apache Spark Streaming, Kafka and HarmonicIO: A Performance Benchmark and Architecture Comparison for Enterprise and Scientific Computing". In: *Int. Symposium on Benchmarking, Measuring and Optimization*. Springer. 2019, pp. 335–347.

[23] Zheng Li, Diego Seco, and Alexis Eloy Sánchez Rodríguez. "Microservice-oriented platform for internet of big data analytics: A proof of concept". In: *Sensors* 19.5 (2019), p. 1134.

[24] W. Hasselbring and G. Steinacker. "Microservice Architectures for Scalability, Agility and Reliability in E-Commerce". In: *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. 2017, pp. 243–246. DOI: 10.1109/ICSAW.2017.11.

[25] Christoph Augenstein, Norman Spangenberg, and Bogdan Franczyk. "An Architectural Blueprint for a Multi-purpose Anomaly Detection on Data Streams". In: *ICEIS (1)*. 2019, pp. 470–476.

[26] Valeria Cardellini, Gabriele Mencagli, Domenico Talia, and Massimo Torquati. "New Landscapes of the Data Stream Processing in the era of Fog Computing". In: *Future Generation Computer Systems* 99 (2019), pp. 646–650. ISSN: 0167-739X. DOI: 10.1016/j.future.2019.03.027.

[27] Pieter Bonte, Femke Ongenae, Femke De Backere, Jeroen Schaballie, Dörthe Arndt, Stijn Verstichel, Erik Mannens, Rik Van de Walle, and Filip De Turck. "The MASSIF platform: a modular and semantic platform for the development of flexible IoT services". In: *Knowledge and Information Systems* 51.1 (2017), pp. 89–126.

[28] Mathias De Brouwer, Pieter Bonte, Dörthe Arndt, Miel Vander Sande, Pieter Heyvaert, Anastasia Dimou, Ruben Verborgh, Filip De Turck, and Femke Ongenae. "Distributed Continuous Home Care Provisioning through Personalized Monitoring & Treatment Planning". In: *Companion Proc. of the Web Conf. 2020*. Taipei, Taiwan, 2020, pp. 143–147. ISBN: 9781450370240. DOI: 10.1145/3366424.3383528.

[29] Christof Mahieu, Femke Ongenae, Femke De Backere, Pieter Bonte, Filip De Turck, and Pieter Simoens. "Semantics-based platform for context-aware and personalized robot interaction in the internet of robotic things". In: *Journal of Systems and Software* 149 (2019), pp. 138–157.

[30] V. Vercruyssen, W. Meert, G. Verbruggen, K. Maes, R. Bäumer, and J. Davis. "Semi-Supervised Anomaly Detection with an Application to Water Analytics". In: *2018 IEEE International Conference on Data Mining (ICDM)*. 2018, pp. 527–536. DOI: 10.1109/ICDM.2018.00068.

[31] P. Shah, D. Hiremath, and S. Chaudhary. "Big Data Analytics Architecture for Agro Advisory System". In: *2016 IEEE 23rd International Conference on High Performance Computing Workshops (HiPCW)*. 2016, pp. 43–49. DOI: 10.1109/HiPCW.2016.015.

[32] S. Amini, I. Gerostathopoulos, and C. Prehofer. "Big data analytics architecture for real-time traffic control". In: *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*. 2017, pp. 710–715. DOI: 10.1109/MTITS.2017.8005605.

[33] S. R. Shaw, L. K. Norford, D. Luo, and S. B. Leeb. "Detection and Diagnosis of HVAC Faults via Electrical Load Monitoring". In: *HVAC&R Research* 8.1 (2002), pp. 13–40. DOI: 10.1080/10789669.2002.10391288.

[34] Ira Assent, Philipp Kranen, Corinna Baldauf, and Thomas Seidl. "Anyout: Anytime outlier detection on streaming data". In: *International Conference on Database Systems for Advanced Applications*. Springer. 2012, pp. 228–242.

[35] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. "Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets". In: *16th Int. Conf. on Data Mining (ICDM)*. IEEE, Dec. 2016, pp. 1317–1322. ISBN: 978-1-5090-5473-2. DOI: 10.1109/ICDM.2016.0179.

[36] Daniele Dell'Aglio, Emanuele Della Valle, Frank van Harmelen, and Abraham Bernstein. "Stream reasoning: A survey and outlook". In: *Data Science* 1.1-2 (2017), pp. 59–83.

[37] *The Obelisk platform*. Accessed Jan 2021. URL: https://obelisk.ilabt.imec.be.

[38] Vincent Bracke, Merlijn Sebrechts, Bart Moons, Jeroen Hoebeke, Filip De Turck, and Bruno Volckaert. "Design and evaluation of a scalable IoT backend for Smart Ports". In: *Software: Practice and Experience* (Submitted Aug 2020).

[39] Jose Santos, Thomas Vanhove, Merlijn Sebrechts, Thomas Dupont, Wannes Kerckhove, Bart Braem, Gregory Van Seghbroeck, Tim Wauters, Philip Leroux, Steven Latre, et al. "City of things: Enabling resource provisioning in smart cities". In: *IEEE Communications Magazine* 56.7 (2018), pp. 177–183.

[40] Pieter Moens, Vincent Bracke, Colin Soete, Sander Vanden Hautte, Diego Nieves Avendano, Ted Ooijevaar, Steven Devos, Bruno Volckaert, and Sofie Van Hoecke. "Scalable fleet monitoring and visualization for smart machine maintenance and industrial IoT applications". eng. In: *SENSORS* 20.15 (2020), p. 15. ISSN: 1424-8220. DOI: 10.3390/s20154308.

[41] Victor Araujo, Karan Mitra, Saguna Saguna, and Christer Åhlund. "Performance evaluation of FIWARE: A cloud-based IoT platform for smart cities". In: *Journal of Parallel and Distributed Computing* 132 (2019), pp. 250–261. ISSN: 0743-7315. DOI: 10.1016/j.jpdc.2018.12.010.

[42] Gerald Haesendonck, Wouter Maroy, Pieter Heyvaert, Ruben Verborgh, and Anastasia Dimou. "Parallel RDF generation from heterogeneous big data". In: *Proceedings of the International Workshop on Semantic Big Data*. 2019, pp. 1–6.

[43] Anastasia Dimou, Tom De Nies, Ruben Verborgh, Erik Mannens, Peter Mechant, and Rik Van de Walle. "Automated metadata generation for Linked Data generation and publishing workflows". In: *LDOW2016*. CEUR-WS. org. 2016, pp. 1–10.

[44] Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. "RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data". In: *Proceedings of the 7th Workshop on Linked Data on the Web*. Vol. 1184. Apr. 2014.

[45] Pieter Heyvaert, Ben De Meester, Anastasia Dimou, and Ruben Verborgh. "Declarative Rules for Linked Data Generation at your Fingertips!" In: *Proceedings of the 15th ESWC: Posters and Demos*. 2018.

[46] Holger Neuhaus and Michael Compton. "The semantic sensor network ontology". In: *AGILE workshop on challenges in geospatial data harmonisation, Hannover, Germany*. 2009, pp. 1–33.

[47] Krzysztof Janowicz, Armin Haller, Simon JD Cox, Danh Le Phuoc, and Maxime Lefrançois. "SOSA: A lightweight ontology for sensors, observations, samples, and actuators". In: *Journal of Web Semantics* 56 (2019), pp. 1–10.

[48] Bram Steenwinckel et al. "FLAGS: A methodology for adaptive anomaly detection and root cause analysis on sensor data streams by fusing expert knowledge with machine learning". In: *Future Generation Computer Systems* 116 (2021), pp. 30–48. ISSN: 0167-739X. DOI: 10.1016/j.future.2020.10.015.

[49] Umutcan Şimşek, Elias Kärle, and Dieter Fensel. "RocketRML-A NodeJS implementation of a use-case specific RML mapper". In: *arXiv preprint arXiv:1903.04969* (2019).

[50] *CARML*. Accessed Jan 2021. URL: https://github.com/carml/carml.

[51] Pieter Heyvaert, Anastasia Dimou, Aron-Levi Herregodts, Ruben Verborgh, Dimitri Schuurman, Erik Mannens, and Rik Van de Walle. "RMLEditor: A Graph-Based Mapping Editor for Linked Data Mappings". In: *The Semantic Web. Latest Advances and New Domains*. Springer Int. Publishing, 2016, pp. 709–723. ISBN: 978-3-319-34129-3.

[52] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. "Unsupervised real-time anomaly detection for streaming data". In: *Neurocomputing* 262 (2017), pp. 134–147.

[53] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[54] Dieter De Paepe, Olivier Janssens, and Sofie Van Hoecke. "Eliminating Noise in the Matrix Profile". In: *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM,* INSTICC. SciTePress, Feb. 2019, pp. 83–93. ISBN: 978-989-758-351-3. DOI: 10.5220/0007314100830093.

[55] Dieter De Paepe, Diego Nieves Avendano, and Sofie Van Hoecke. "Implications of Z-Normalization in the Matrix Profile". In: *Pattern Recognition Applications and Methods*. Springer International Publishing, 2020, pp. 95–118. DOI: 10.1007/978-3-030-40014-9_5.

[56] Yan Zhu, Zachary Zimmerman, Nader Shakibay Senobari, Chin-Chia Michael Yeh, Gareth Funning, Philip Brisk, and Eamonn Keogh. "Matrix Profile II : Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins". In: *16th International Conference on Data Mining (ICDM)*. IEEE. 2016, pp. 739–748. DOI: 10.1109/ICDM.2016.126.

[57] Diomidis H Stamatis. *Failure mode and effect analysis: FMEA from theory to execution*. Quality Press, 2003.

[58] Wen-Shing Lee, Doris L Grosh, Frank A Tillman, and Chang H Lie. "Fault Tree Analysis, Methods, and Applications - A Review". In: *IEEE transactions on reliability* 34.3 (1985), pp. 194–203.

[59] Bram Steenwinckel, Pieter Heyvaert, Dieter De Paepe, Olivier Janssens, Sander Vanden Hautte, Anastasia Dimou, Filip De Turck, Sofie Van Hoecke, and Femke Ongenae. "Towards adaptive anomaly detection and root cause analysis by automated extraction of knowledge from risk analyses". eng. In: *Proc. of the 9th int. semantic sensor networks workshop, co-located with 17th int. semantic web conf. (ISWC 2018)*. Vol. 2213. Monterey, CA, USA, 2018, pp. 17–31.

[60] Sander Vanden Hautte, Pieter Moens, Joachim Van Herwegen, Dieter De Paepe, Bram Steenwinckel, Stijn Verstichel, Femke Ongenae, and Sofie Van Hoecke. "A Dynamic Dashboarding Application for Fleet Monitoring Using Semantic Web of Things Technologies". In: *Sensors* 20.4 (2020). ISSN: 1424-8220. DOI: 10.3390/s20041152.

[61] Sander Vanden Hautte, Dieter De Paepe, Bram Steenwinckel, Pieter Moens, Stijn Verstichel, Steven Vandekerckhove, Femke Ongenae, and Sofie Van Hoecke. "Event-driven dashboarding and feedback capturing for improved anomaly and fault detection and reduced human labeling effort". In: *Engineering Applications of AI* (Submitted Jan 2021).

[62] Niall Richard Murphy. *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, Apr. 2016. ISBN: 149192912X. URL: https://www.xarg.org/ref/a/149192912X/.

[63] Brendan Gregg. *The USE Method*. Accessed Jan 2021. URL: http://brendangregg.com/usemethod.html.

Chapter 7

Conclusion

This chapter summarizes the findings of this dissertation and discusses possibilities for future work.

7.1 Insight Mining

Ever more companies are starting to acknowledge the value of collecting and analyzing data, inspired by the many success stories of early adopters. Indeed, as society becomes ever more digitized, companies that delay their adoption of data analytics in their business plan will be at a disadvantage against competitors that are able to extract insights from data. Where the past decade has seen much focus on big data, i.e. extremely large quantities of data, more recent innovations in sensors and network connectivity have opened the possibility of continuous monitoring for services and machines, and the IoT as a whole. The resulting time series captured this way open a whole new dimension in which data analytics can take place.

Traditional analytics techniques can still be applied to time series by compressing series using statistical operations or by treating preceding time windows as independent features. However, the temporal (or spatial) nature of time series allows us to look for patterns within time windows as well, i.e. subsequence patterns. In the past years, several methods for insight mining on time series have been proposed, including visualization, segmentation or rule mining. Many of these techniques are based on the retrieval of repeating subsequences from time series. Two important parameters here are the proper subsequence length to consider, as well as the proper distance measure, both are highly dependant of the foreseen use case. Other considerations, also applicable to time series analytics in general, include computational complexity and the incorporation of domain knowledge.

This dissertation introduced several new techniques to extract insights from time series. In Chapter 2, we demonstrated the use of a hybrid model to detect anomalous welds in a steel production line. The hybrid model combines a classical data-driven ap-

proach with domain knowledge related to the welding procedure. The model is trained using gradient descent and hereby learns the previously unknown properties of the various steel types and the welding machine. Not only is this approach more credible and trustworthy for the engineers, the model-learned parameters can form a basis for further analytics. Chapter 3 introduced the contextual matrix profile as a new pattern-based analytics technique. It can be used to capture non-periodic repetitions in series and has direct applications in visualization or contextual anomaly detection, but may also serve as a type of feature generator. Finally, in Chapter 5 we presented a new and improved techniques for finding patterns that are repeated multiple times in one or more time series, i.e. consensus and common motifs. These motifs can easily be retrieved using the radius profile, a new series derived from the original series. Seeing how the matrix profile inspired many techniques by facilitating motif retrieval, we hope the radius profile may similarly prove as a base for further analytics techniques.

Besides new techniques, this dissertation also addressed some of the challenges facing existing techniques. As mentioned before, the hybrid model in Chapter 2 effectively integrates expert knowledge into the hybrid model. The knowledge was integrated in the form of physical laws related to the welding process. The series distance framework, introduced in Chapter 3, recognizes the fact that different use cases may need different definitions of similarity when using pattern-based techniques, and aims to make experimentation easier. In this framework, users can combine various distance measures with various analytical techniques in an easy and efficient way. Following the dogfooding principle, we used our open source implementation of this framework as the foundation for all techniques described in this dissertation. Where Chapter 3 eased the use of more distance measures in the broad way, Chapter 4 focused on extending the applicability of the most used distance measure available, i.e. the z-normalized Euclidean distance. This chapter shed light on various characteristics of this distance measure, such as its link to the Pearson correlation and suggested a way to normalize distances irrespective of subsequence length. Still, the main contribution of this chapter was a way to bring this distance measure more in line with the human intuition of similarity when dealing with noisy data that lacks distinctive patterns.

7.2 Anomaly Detection

Anomaly detection, or the more generic event detection, is one specific aspect of data analytics that many companies have a direct interest in. It is an interesting subtopic because it has many applications with obvious return on investments, such as product quality assurance, safety or automation. Though research has removed many hurdles related to anomaly detection, many of the remaining challenges are in fact related to real-world constraints. Perhaps one of the most frustrating challenge for any data scientist is the lack of labeled data that can be used for training and evaluation. One solution that is demonstrated in Chapter 4 is to use unsupervised methods to look for possible anomalies and present these to a user, ordered from most to least suspicious, a pro-

cess that continues until all suspicions are listed or until the user runs out of time. An extension of this approach is to jump-start labeling by combining rule-based anomaly detection using expert domain knowledge with unsupervised methods, as is explained in Chapter 6. As the unsupervised methods detect suspected anomalies, they are presented to the user who can verify or disregard them, after which this user feedback is fed back into the event detection methods. Such a system removes the need of big and cumbersome labeling campaigns upfront and has the potential to become more accurate as users interact more with it, though managing these evolving algorithms is a challenge on its own. A different challenge related to anomaly detection is the incorporation of context information, therefore and finally, in Chapter 3, the contextual matrix profile was introduced. This technique lets users define a set of temporal contexts where the data should display similar behavior, based on this it is straightforward to find anomalous regions in a series.

7.3 Future Work

This dissertation addressed several challenges related to analytics and anomaly detection but should be seen as a single step on a long and never ending path of improvement. Without any doubt, further improvements, specializations and new techniques will be discovered, some perhaps for use cases that are currently not even under consideration. While claims about the future inherently come with a degree of uncertainty, we discuss some research directions that seem promising.

Dynamic & Interactive Anomaly Reporting Systems

While a feedback loop between users and anomaly detection systems has great potential, it also comes with a series of new challenges. Similar to how anomalies can be diverse, so can users. Users often have different backgrounds and interests and may look for other details when examining anomalies. This means that a system that presents anomalies in a static way will not be appreciated by all users. Dynamic dashboards, as introduced in Chapter 6, that take anomaly and user information into account for automatic widget selection can be useful in this regard. Still, the task of incorporating user and anomaly metadata is not straightforward because of a lack of reference data. Both the number of users interacting with the system and the number of visualized anomalies will be small when compared to datasets used for traditional machine learning. When dividing users into different groups, the number of available interactions to learn from decreases even more. Since the dynamic dashboard will have to learn user preferences from this small amount of data, it may be worth to consider generalized models that consider all interaction data, and smaller, personalized models that are derived thereof. A similar system is used to classify priority emails in Google Mail, be it on a larger scale [1].

Pattern-based Incremental Event Detection

Pattern-based event detection systems function by scanning incoming data for the presence of a pattern that indicates a particular event. As more event instances are found, user feedback may help refine the detection criteria. However, user feedback may not always be as helpful in practice. Users may mislabel certain prediction by accident or change their labeling methodology as a result of new insights. Alternatively, distinct events may be grouped together by a user under a single label. These types of scenarios are common for anomaly detection datasets and are tackled by the data scientist. However, in systems that allow more interactivity between models and users (who are responsible for labeling), an interactive process to clear out inconsistencies may be more fruitful. A likely outcome is that the detection system is composed of multiple models that focus on distinct patterns. Since pattern-based approaches utilize a high number of computations, the question can be asked whether we can optimize multi-pattern search for a single series?

Super-speed Series Distance Matrix

The series distance matrix framework is intended to maximize the ease of experimentation, and as such, is intended for academic researchers. In contrast, the main concern of company data scientists is scale and speed, as apparent in company driven implementations [2, 3]. These implementations rely on distributed or GPU-enabled workflows, but implement independent implementations that can utilize only a single distance measure. It would be interesting to investigate to what degree the flexibility of the series distance matrix framework is compatible with the performance boosts available through GPU and distributed workflows. Additional computational optimizations may be possible for scenarios where multiple analytics techniques are combined (i.e. multiple SDM-generators or SDM-consumers) by sharing common computational operations. For example, both the Euclidean and z-normalized Euclidean distance can be efficiently calculated using dot products, so this calculation could be shared when both measures are needed. One particularly interesting route to explore here would be the exploitation of computation graphs, as used in modern machine learning frameworks like TensorFlow.

Insight Mining for Non-discrete Contexts

The contextual matrix profile allows a user to define discrete temporal contexts that can be used to find mismatches in expected behavior. Unfortunately, contexts are more complex in many application domains and representing them with distinct contexts would lose information about context similarity. Consider for example different operators manning a machine where various sensors monitor machine characteristics. The operator could be an important context factor in the resulting measurements. While it would be easy to represent each operator as distinct, it might be more interesting to

consider how similar some operators are. For example, two operators that received the same training will be more similar to each other compared to two operators that have the same age. Since contexts have no predefined form, property or knowledge graphs may be a good way to represent them. While the utilization of knowledge graphs as input for machine learning has become a topic of research [4], the question remains how to properly define context similarity and how to incorporate this into insight mining methods.

References

[1] Douglas Aberdeen, Ondrey Pacovsky, and Andrew Slater. "The learning behind gmail priority inbox". In: (2010).

[2] Sean M. Law. "STUMPY: A Powerful and Scalable Python Library for Time Series Data Mining". In: *The Journal of Open Source Software* 4.39 (2019), p. 1504.

[3] Andrew Van Benschoten, Austin Ouyang, Francisco Bischoff, and Tyler Marrs. "MPA: a novel cross-language API for time series analysis". In: *Journal of Open Source Software* 5.49 (2020), p. 2179. DOI: 10.21105/joss.02179. URL: https://doi.org/10.21105/joss.02179.

[4] Palash Goyal and Emilio Ferrara. "Graph embedding techniques, applications, and performance: A survey". In: *Knowledge-Based Systems* 151 (2018), pp. 78–94. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2018.03.022. URL: https://www.sciencedirect.com/science/article/pii/S0950705118301540.

Multiple time series with various detected events and other insights, including a few anomalies.