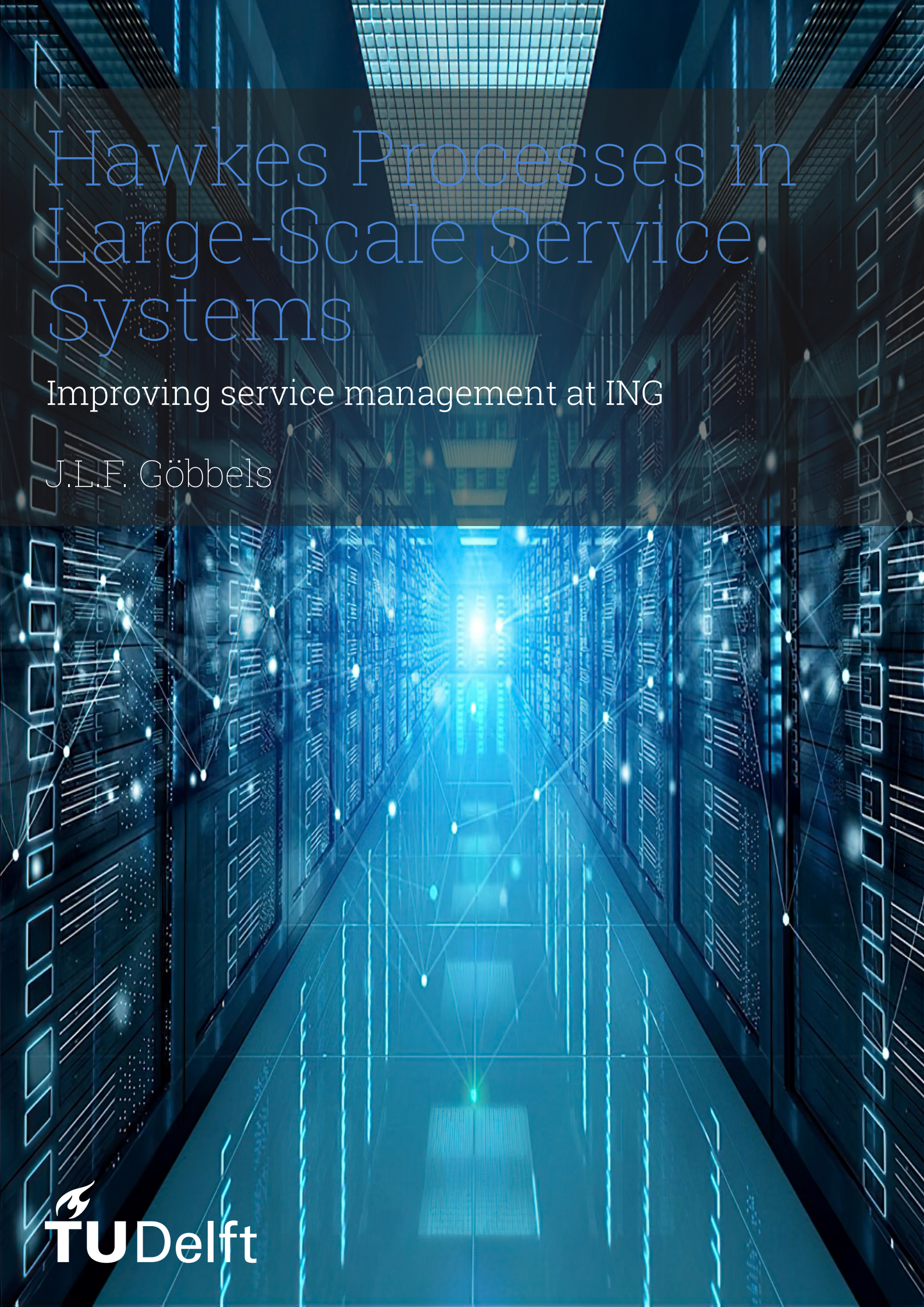

Hawkes Processes in Large-Scale Service Systems

Improving service management at ING

J.L.F. Göbbels

TU Delft

Hawkes Processes in Large-Scale Service Systems

Improving service management at ING

by

J.L.F. Göbbels

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday 13 September, 2023 at 15:00.

Student number:	4596498	
Project duration:	September 24, 2022 – September 13, 2023	
Thesis committee:	Prof. Dr. Ir. G. Jongbloed	TU Delft, chair
	Dr. A. F. F. Derumigny,	TU Delft, supervisor
	Dr . L. M. Da Cruz	TU Delft, supervisor
	F. Den Hengst,	VU & ING, daily supervisor

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

'The object of this study is
to produce a class of theoretical models
which may be applicable to a variety of problems.'

HAWKES (1971A)

Preface

This is an incomplete version of the original thesis. In particular, details regarding incident arrivals are removed.

This thesis examines how marked Hawkes processes contribute to providing insights into large-scale service systems. To achieve this, we construct a mark space based on the product of clusters of IT messages combined with distinct service levels. This thesis has been written in order to obtain the degree of Master of Science in both Applied Mathematics and Computer Science. The research was carried out under the supervision of Dr. Derumigny from the Applied Mathematics side and Dr. Da Cruz from the Computer Science side.

Selecting a topic that is closely aligned with industrial applications and also meets the criteria for Applied Mathematics and Computer Science has been a challenging endeavor. I, therefore, express gratitude to Elvan Kula and Evert-Jan van Doorn for creating a thesis position at ING that fulfills these requirements. Furthermore, I would like to thank the AI for Fintech Research (AFR) lab for their continuous feedback throughout the process, as well as the I3 team for their support in addressing business-related matters that emerged throughout the process. I especially acknowledge Arkadiusz Trawiński for providing the opportunity to present this work at the EuroScipy Conference 2023 in Basel.

I would like to express my heartfelt gratitude to Dr. Derumigny, whose invaluable constructive feedback greatly enriched the project. Our extensive discussions substantially enhanced my mathematical understanding, and your prompt feedback enabled me to explore various aspects within a tight timeline. Similarly, I extend my gratitude to Dr. Da Cruz for his flexibility, which allowed me to refine the scope of the project when confronted with an abundance of ideas. Finally, my heartfelt thanks go to Floris den Hengst for acting as a mentor throughout the process, contributing to various facets beyond the realm of academic guidance. Furthermore, I would like to extend my gratitude to Prof. Dr. Jongbloed for serving as the thesis committee chair. As my instructor for "Introduction to statistics," precisely six years ago, your role in this academic milestone brings a sense of closure to a symbolic cycle.

Finally, I want to thank all my friends from EEMCS, whose support proved to be invaluable during the final phase of my academic journey at TU Delft. Your friendship, shared coffee moments, and uplifting words were priceless. Additionally, I would like to thank my parents for their unconditional support, not just this year but throughout my entire lifetime.

J.L.F. Göbbels
Rotterdam, October 2023

Abstract

Through the expansion of large-scale service systems and the exponential growth of data generated by complex IT infrastructure components, gaining a comprehensive overview of the different levels of service within an IT system has become increasingly challenging. In particular, this brought to the fore the question from a large commercial bank of how IT monitoring data streams generated by their complex IT infrastructure can be associated with one another.

In more detail, the data from the monitoring stream consists (among other things) of a message and a time stamp. Moreover, the monitoring data stream of this bank consists of two natures of information. These natures are either automatically generated warnings in the form of *events* or unplanned outages, referred to as *incidents*. The events and incidents are referred to as *arrivals*. As a first requirement to obtain better granularity, both event and incident messages with similar semantics should be grouped together. To this extent, the message component from each arrival is transformed into a numerical vector, the dimension of the obtained vector is reduced, and the collection of vectors is clustered. Once the individual arrival from the IT monitoring data stream is attached to a cluster based on their message component, the arrival is assigned a *mark*. This mark consists of a combination of the assigned cluster, the nature, and three different levels of service from the IT architecture on which the arrival occurred.

From a mathematical point of view, we can now view the monitoring data stream from different levels of service as a marked point process. Our primary focus centers on a specific category of marked point processes, known as marked Hawkes processes. Given the marked Hawkes process, we assume that each arrival from the IT monitoring data stream results in an instantaneous increase in the probability of some other arrivals in the near future. From here, we estimate the *excitation matrix*, representing the instantaneous increases among all assigned marks. Once the estimated excitation matrix is obtained, we decompose it into the different levels of service as defined within the mark. In particular, the decomposition has been performed through means of hierarchical linear models. Finally, the decomposition resulted in a comprehensive overview of the excitation behavior in large-scale service systems. This overview can directly be incorporated into the field of Software Architecture in order to uncover associations within complex IT infrastructures.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
MLE	Maximum likelihood estimator
LSE	Least squares estimator
NLP	Natural language processing

Symbols

Symbol	Definition	Reference
Number sets		
$\mathbb{N}$	Natural numbers	
$\mathbb{N}^*$	Positive natural numbers $\mathbb{N}\backslash\{0\}$	
$\mathbb{R}$	Real line	
$\mathbb{R}_+$	Non-negative real line	
$\mathbb{R}_+^*$	Positive real numbers	
$\mathcal{B}(\mathbb{R})$	Borel set on real line	
Arrivals		
i	Arrival	Section 3.2
t_i^*	Arrival time for arrival i	Section 3.1
n^*	Total number of arrivals	Section 3.1
n	Total number of records	Section 3.1
T_{start}	Starting time from which to consider arrivals	Section 3.1
T_{start}	Ending time until which to consider arrivals	Section 3.1
R_i	Feature vector for arrival i	Section 3.1
F	Space of feature vectors	Section 3.1
$\mathcal{D}^*$	Set of all arrivals	Section 3.1
$\mathcal{D} = \mathbf{Comp}(\mathcal{D}^*)$	Set of all records	Section 3.1.2
$\tilde{N}_{k,s}^{\cdot}$	Cardinality function for index k and nature s	
Service systems		
CI	Set of configuration items $\{ci_1, \dots ci_{n_{ci}}\}$	Definition 4.1.1
BA	Set of business applications $\{ba_1, \dots, ba_{n_{ba}}\}$	Definitions 4.1.2
$LEVEL$	Set of level identifiers $CU \cup BA$	Section 3.1.1
BU	Set of business units $\{bu_1, \dots, bu_{n_{bu}}\}$	Definition 4.1.3
Q_i	Mapping vector for service level components	Section 4.1.1
G	Space of mapping vectors	Section 4.1.1

Symbol	Definition	Reference
Hawkes		
$N(\cdot)$	Counting process	Definition 5.1.5
$\mathbb{T}$	Point process	Definition 5.1.2
$\mathcal{H}_t$	History up to but not including time t	Section 5.1.5
$\lambda(\cdot\|\mathcal{H}_t)$	Conditional intensity function	Definition 5.1.15
$\phi(\cdot)$	Memory kernel function	Definition 5.2.1
$\mu(\cdot)$	Background intensity function	Definition 5.2.1
μ	Constant background intensity	Definition 5.2.1
τ	Characteristic time	Definition 5.2.5
α	Excitation	Definition 5.2.5
$\mathcal{U}$	Mark space	Section 5.3.1
$N_u(\cdot)$	Marked counting process	Section 5.3.1
$\mathbb{T}_{MPP}$	Marked point process	Section 5.3.1
$\lambda_u(t\|\mathcal{H}_t)$	Marked conditional intensity function	Definition 5.3.3
$\mathcal{M}$	Background intensity matrix	Definition 5.3.8
$\mathcal{T}$	Characteristic time matrix	Definition 5.3.7
$\mathcal{A}$	Excitation matrix	Definition 5.3.5
$L(\theta;\mathcal{H}_T)$	Likelihood functional	Theorem 5.2.9
$\mathcal{L}(\theta;\mathcal{H}_T)$	Log-likelihood functional	Section 5.2.2
$R(\theta;\mathcal{H}_T)$	Least squares functional	Theorem 5.2.14
$\hat{\theta}_{MLE}$	Maximum likelihood estimator	
$\bar{\theta}_{LSE}$	Least squares estimator	
θ_0	True parameter	

Part I

Preliminaries

1 Introduction

The transition to modern software applications has greatly benefited society and the way we do business. Particularly within the banking sector, the adoption of these advanced systems has ushered in a new era, replacing archaic paper-based processes with streamlined digital workflows. The surge in data volume and complexity resulting from this digital metamorphosis has brought to the fore an unexpected consequence: the metamorphosis of major banks into IT companies, equipped with banking licenses.

With the expansion of large-scale service systems and the exponential growth of data generated by complex IT infrastructure components, gaining a comprehensive overview of the systems at hand has become increasingly challenging. Without a clear understanding of the interdependencies within the IT system, this can lead to chaotic IT scenarios and ultimately result in substantial system downtime. Such prolonged downtime not only affects the reliability of the organization but also directly impacts revenue. According to estimates by the Ponemon Institute, service downtime costs an average of $9,000 per minute [16] in revenue loss to a typical large-scale company. Consequently, the prevention of service downtime has become a top priority and a demanding task of companies' IT departments.

1.1. ING bank

ING, a prominent multinational bank headquartered in the Netherlands, has solidified its position in the financial industry, serving a vast network of approximately 37 million customers, corporate clients, and financial institutions across more than 40 countries. With a workforce of over 60.000 employees, ING operates at a global scale, catering to diverse financial needs. Notably, 18.000 employees, more than a quarter of its workforce, are engaged in IT infrastructure management. This integral segment of the organization is responsible for overseeing and maintaining the bank's IT infrastructure, which includes data centers, networks, servers, and other hardware components. The IT process teams ensure the infrastructure is scalable, secure, and able to handle the increasing demands of the digital age. Moreover, the IT teams play a crucial role in exploring and integrating emerging technologies to improve banking services and remain competitive in the industry.

1.2. Incidents & events

Service downtime is captured by documenting *incidents*. An incident is an occurrence that disrupts or has the potential to disrupt the normal operation of IT services. This encompasses a wide range of affairs, from minor issues such as a single user being unable to access a specific application to major problems such as an outage of the bank's online banking system, which prevents customers from accessing their accounts or making transactions [18].

On the other hand, *events* refer to the automatically generated information produced by computer systems and other devices during their operation. Events are sometimes also referred to as *alerts*. Events

capture the state of different components within the service system. Unlike incidents, events do not directly indicate service disruptions. They encompass a wide range of data streams, including user interactions with the system, system performance metrics, and other details that offer insights into the system's state. Event data is typically collected and stored in software logs, which serve monitoring and analysis purposes.

Throughout this thesis, the differentiation between events and incidents is crucial. When addressing both events and incidents at the same time, we refer to them as *arrivals*. To clarify the distinction between these two possible natures of arrivals, we use the following formal definitions:

- *Incidents* are unplanned interruptions and outages of the service system [12]. These should be resolved as soon as possible to restore normal service operations and ensure business operations are minimally impacted.
- *Events* are automatically generated occurrences that report the state of different components within the monitoring system, including metrics, logs, and traces. Events are characterized by dozens of attributes and thus come with the challenge of extracting useful information [82].

1.3. AIOps

This thesis can be viewed within the broader framework of AIOps, which stands for Artificial Intelligence for IT Operations. The term *AIOps* was originally introduced by IT consulting company Gartner in 2016. According to Gartner's definition [38]:

> "AIOps combines big data and machine learning to automate IT operations processes, including event correlation, anomaly detection and causality determination."

AIOps employs sophisticated statistical analysis and machine learning methods on IT operations data to elevate the performance, availability, and overall quality of IT services. It empowers organizations to automate and optimize their IT operations by analyzing vast amounts of data from diverse sources, such as logs and metrics. AIOps offers organizations the capability to recognize patterns, anomalies, and trends within their IT monitoring data stream.

At ING, AIOps plays a critical role in ensuring continuous service availability. This not only enhances customer satisfaction but also serves as a crucial factor for compliance with specific standards imposed by regulators. Non-compliance to these standards can result in penalties up to 10% of the bank's annual turnover, imposed by the European Central Bank [5].

1.4. Hawkes processes

For our purpose, the Hawkes process emerges as a suitable choice for modeling the IT monitoring data stream. The Hawkes process extends the Poisson process by incorporating temporal dependencies based on previous arrivals. Through the introduction of an intensity function with cross-excitation, this model enables us to explore how arrivals from one process influence the intensity of other processes in the near future.

We provide a formal mathematical definition of the Hawkes process and thoroughly explore its implications within the realm of Software Architecture later on in Chapter 5. By delving into the results derived from this analysis, we aim to gain a deeper understanding of the association contained within the IT monitoring data stream. These findings shed light on the dynamics of service systems and their operational implications, ultimately contributing to providing a comprehensive overview of the different levels of service.

1.5. Research questions

To enhance our understanding of event and incident arrivals, we initiate the analysis by examining the IT monitoring data stream from ING. We closely scrutinize the event and incident data, as well as explore the relationship between the different service levels. Based on this information, we address the following research questions:

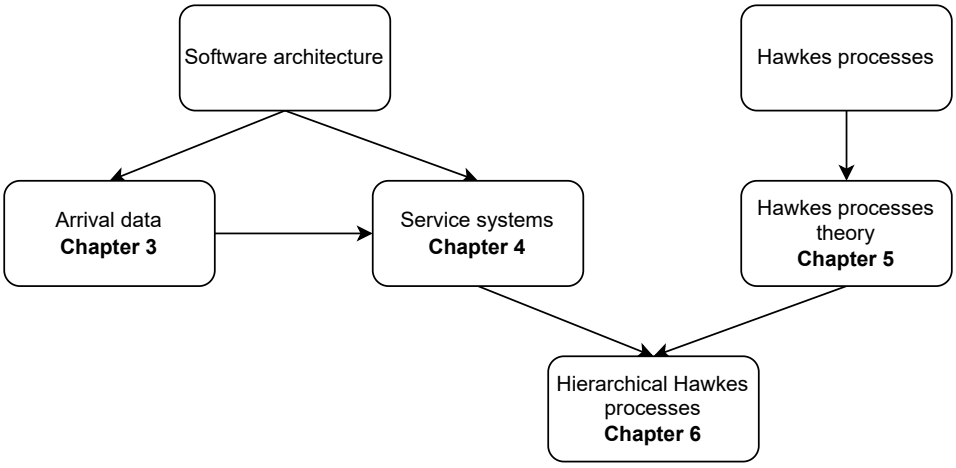

Figure 1.1: Thesis Outline.

- *Research Question 1:* How can we design a hierarchical architecture that resembles the operations of large-scale service systems?
- *Research Question 2:* How can IT messages with similar semantics be grouped together?
- *Research Question 3:* How can the marked Hawkes process be employed to capture interactions among arrivals? And how can the interactions be estimated?
- *Research Question 4:* How can the estimated excitation matrix contribute to understanding the associations within a level of service?

By addressing these questions, this thesis aims to uncover the operational dynamics of the underlying service structure in a mathematically concise and rigorous manner. The study endeavors to employ rigorous mathematical techniques to analyze and elucidate the complex relationships within the service architecture, ultimately contributing to a deeper comprehension of its underlying associations.

1.6. Thesis outline

The outline of this thesis is depicted in Figure 1.1. In *Chapter* 2, we establish the foundation with essential background information on incident management and Hawkes processes. This foundational chapter provides vital information for comprehending the subsequent analyses and findings presented in later chapters.

In *Chapter* 3, we conduct a descriptive analysis of ING's monitoring data stream, spanning a two-year period. In addition, we provide valuable insights into the volume of arrivals, reveal seasonal patterns in the arrival frequency, and offer a breakdown of these arrivals into various components, including service levels and priorities.

In *Chapter* 4, we describe the hierarchical service architecture at the heart of ING's IT infrastructure. First, we introduce the highest level of service, namely the business unit. Secondly, we show how we can create the lowest level of service through the implementation of message clusters. For the purpose of message clustering, we demonstrate how to convert the messages to numerical vectors using the TF-IDF message embedding, reduce the dimension of the obtained message vector using UMAP, and cluster the reduced message vectors using the HDBSCAN clustering algorithm. Additionally, this process has been performed on real world data from ING for one business unit. Moreover, the clustering procedure covers the first six step from Algorithm 1 for obtaining the estimated excitation matrix $\hat{\mathcal{A}}$. Finally, it should be noted this chapter constitutes the core of our *Computer Science* contribution.

Transitioning to *Chapter* 5, a study of the theory behind Hawkes processes takes center stage, encompassing both one-dimensional and marked processes. We establish a mathematically rigorous foundation, enabling for inference of the estimated excitation matrix. In particular, we discuss the maximum

likelihood and least squares function for the marked Hawkes process. Additionally, we show how we can estimate the excitation matrix using two of the estimators from the Python package *Tick*. Moreover, one of these estimators, the ADM4 estimator, will result in our final choice for obtaining the estimated excitation matrix, as can be found in step ten from Algorithm 1. Finally, it should be noted this chapter constitutes the core of our *Applied Mathematics* contribution.

In *Chapter* 6, our focus expands to show how we can decompose the estimated excitation matrix $\hat{\mathcal{A}}$ into a 5-level hierarchical linear model, offering a comprehensive examination of cascading effects within the large-scale service system at ING. Furthermore, we resolve the practical consideration between two of the levels of service, namely the configuration item and the business application, and we elaborate on the choice of characteristic time parameter τ. Once this is all established, we create mark space $\mathcal{U}_m$. This process covers steps seven to nine in Algorithm 1. We end the chapter highlighting the implications these models have for the field of Software Architecture. In particular, we show how the hierarchical linear models contribute to obtaining a comprehensive overview of the service architecture for the large-scale service systems at ING.

Finally, *Chapter* 7 presents the conclusion and discussion on the obtained result and suggests potential avenues for future research to those interested in extending the applicability of our framework. In particular, we provide valuable guidance for AIOps engineers who aim to integrate the hierarchical Hawkes model into their service architecture.

All the URL references used in this document have been saved in the *WayBackMachine* internet archive. These can be accessed through `http://web.archive.org`.

Algorithm 1 Procedure for obtaining estimated excitation matrix $\hat{\mathcal{A}}$

1: Fix business unit $m \in BU$.
2: **for** all arrival nature $s_i \in S$ **do**
3: Preprocess messages m_i.
4: Run grid search for EMBEDDING+UMAP+HDBSCAN hyperparameters.
5: Obtain message cluster space CLU^m.
6: **end for**
7: Fix characteristic time τ.
8: Resolve configuration item - business application mapping.
9: Created mark space $\mathcal{U}_m$ (using CLU^m).
10: Use the ADM4 estimator to obtain estimated excitation matrix $\hat{\mathcal{A}} \in \mathbb{R}^{U_m \times U_m}$.

2

Background & Related work

In this chapter, we delve into the incident management paradigm. This chapter is divided into two sections. To begin with, we give the definition of the incident management cycle and describe how event data can be used within this process. Subsequently, our attention shifts towards associating event and incident arrivals. Here we examine related works trying to solve the same "event incident association" problem. As we analyze these approaches, we emphasize the need for results that allow for direct interpretability.

This analysis leads us to introduce Hawkes processes. A Hawkes process is a type of stochastic process that can effectively model the self- and cross-excitation behavior observed between event and incident records. We provide a brief background on the characteristics of Hawkes processes and explain how they capture the interdependencies between different marks. Finally, we examine two related works applying Hawkes processes in settings that are comparable to the large-scale service systems that are of interest to us.

Remark 2.0.1. Events are commonly referred to as *alerts* or *monitoring data* in literature. When referring only to the textual descriptions of the events, often the name *system trace logs*, or simply *logs* is used.

2.1. Incident management

With the expansion of large-scale service systems and the exponential growth of data generated by complex IT infrastructure components, interest in Artificial Intelligence for IT operations (AIOps) has been on the rise [36, 50, 51, 52]. AIOps uses big data, AI techniques including natural language processing, and advanced analysis, including statistical analysis, to analyze system performance patterns and improve the quality of service [1]. However, in the context of large-scale service systems, monitoring and translating this data into meaningful and actionable insights poses demanding challenges. Under the umbrella of AIOps, incident management focuses on uncovering and analyzing system performance patterns by mitigating the influence of incidents.

Incident management consists of managing the life cycle of an incident. This cycle runs from 1) identifying the incident, 2) classifying the incident, 3) resolving the incident 4) closing and reviewing the incident. Within the available literature, it is common to tackle only one or two of these tasks. For instance, Chen et al. focused on the third task by prioritizing incidents to increase the fast resolution of important incidents [13]. On the other hand, Zhou et al. [82] focused on the first and second tasks by predicting incidents in the near future and classifying the incoming incidents by generating an interpretable report. In practice, all four stages can greatly benefit from associating the incident arrival with events [50]. First and foremost, events known to be associated with incidents pose early warnings for the arrival of an incident. Furthermore, the associated event provide additional information on the incident's arrival, which helps classify and resolve the incident. Finally, the associated event helps to find the root cause of the incident and helps close the event after all.

However, event data comes with many challenges on its own. Two of the major challenges arising are the high volume of arrivals, which can reach millions of events a day [49], and the large amount of noise present (i.e. irrelevant events) [70]. Filtering out irrelevant events, therefore, is an essential data preparation step, which reduces the large volume of data under consideration prior to performing analysis. Tremendous effort has therefore been devoted to event management in both academia and industry [82]. Emphasis is mainly put on clustering the log trace message and extracting useful information out of the formed clusters [39, 45, 48, 55, 70, 80, 83].

In particular, Zhao et al. [83] proposed an event clustering tool that detects bursts of events for the IT monitoring data stream from a large commercial bank. Their approach clusters associated events and selects one representative event for each of the obtained clusters. Zhang et al. [81] provided a dynamic event anomaly detection framework that overcomes the *closed-world assumption*, which assumes that the log message data stream is stable over time and the set of distinct log messages is known. Zhang et al. [80] proposed a deep learning method for clustering events and classifying which of the formed clusters contain early warning signals for incidents. Subsequently, Lin et al [47] were the first to emphasize the explicit need to distinguish between clustering paradigms for semi-structured event arrivals compared to unstructured incident arrivals. Although the above-mentioned methods successfully prioritized, or clustered, either events or incidents, they left the question of how to associate the obtained event clusters to incident clusters open for future research.

Associating events and incidents was first performed by Yurcik et al. [79], who extended the event correlation method of Li et al. [45] to an event-incident association framework. They did however restrict themselves to security data only and limit their approach to rule-based methods which do not scale to large-scale service systems.

Zhou et al. [84] proposed a framework for associating events and incidents, taking the complex and interconnected service architecture into account. They provided prediction models at both the smaller component-level as well as the larger microservice level. However, they assumed a supervised learning setting where is known which event was associated with which incident. Moreover, they restrained to only three types of incidents.

An unsupervised approach that also incorporates the service/component level distinction was developed by Chen et al. [14]. They proposed a Bayesian network-based outage prediction method, called *AirAlert*. However as demonstrated by Zhao et al. [82], *AirAlert* does not deal with noisy events.

To overcome the noisy event complication, Zhao et al. proposed a multi-instance learning approach called *eWarn* [82]. Their approach consists of feature-engineered components and was successfully evaluated on the service system of a large commercial bank. However, *eWarn*, as well as *AirAlert*, do not use of the exact arrival times but instead make use of fixed time windows. This requires explicit tuning of the prediction window step size per business unit. Furthermore, both frameworks are based on the black-box XGBoost classification algorithm. They therefore rely on an additional tool for interpretable results, called LIME. As mentioned by Slack et al. [71], post hoc explanation techniques, like LIME, can not be seen as reliable.

As state-of-the-art event-incident association frameworks heavily rely on non-interpretable black-box machine learning algorithms, it is of interest to turn our attention to statistical analysis techniques, which benefit from statistical validity, mathematical guarantees, and direct parameter interpretation. The aim is to directly link the temporal occurrence of event arrivals to the temporal occurrence of incident arrivals.

In particular, we turn our attention to marked, or multivariate, Hawkes processes. Marked Hawkes processes, combined with a Granger causality framework, have been adopted priorly within AIOps by Ide et al.[37]. In their study, they performed causal diagnoses on data center warning messages. Their Hawkes-Granger model is mathematically well-defined and allows for causal arrival diagnosis through sparse learning to rule out unlikely options. Furthermore, they provide a high degree of interpretability.

Furthermore, Wei et al. [76] recently demonstrated that marked Hawkes processes are able to achieve

prediction performance similar to XGBoost, while also recovering an interpretable Granger-causal graph structure. They mention the need to distinguish between different regularization methods based on the dimensionality of the problem under consideration, where they found the regularization hyperparameter λ by means of a grid search. They mention the need for domain knowledge in order to set parameters such as the characteristic time, which governs the decay behavior. Furthermore, their estimator exhibits favorable statistical properties such as identifiability and consistency.

Lastly, Bhatt et al. [7] employed marked Hawkes processes on COVID-19 cases by establishing a multilevel model, distinguishing per geographical region. By sharing parameters across regions, they established connections between multiple regions while allowing each region to possess its own baseline intensity. The key thought of their work is to share information between groups, while still permitting between-group variation. This aligns with a hierarchical service architecture, where we want to share information between different hardware and software components while still permitting variation between these components.

2.2. Hawkes processes

Marked Hawkes processes are a type of multivariate process that includes time as a central dimension of analysis. The key task of Hawkes processes is to capture and model relationships along the timeline for various kinds of arrivals. Hawkes processes are a type of stochastic process that carries a self-exciting property [29, 28]. Hawkes processes are characterized through their *conditional intensity function*. The conditional intensity function $\lambda(t|\mathcal{H}_t)$ is defined as the infinitesimal probability of observing an arrival at time t, given its history up to time t, $\mathcal{H}_t$ [1].

In the original paper, Hawkes established the existence of this process [28]. Hawkes and Oakes subsequently formulated a cluster processes representation [30]. Ogata [59] proved that the maximum likelihood estimator (MLE), under a stationarity assumption, is an asymptotically efficient estimator. An estimation procedure for the MLE was proposed by Ozaki [61]. He emphasized that the log-likelihood of the Hawkes model is non-linear with respect to the parameters, and therefore non-linear optimization techniques are needed if we want to use the MLE. Furthermore, Ogata [59] showed that if a random time change based on the integrated conditional infinitesimal function $\lambda(t|\mathcal{H}_t)$ is performed, a stationary Poisson process with an average of one arrival per unit of time is obtained. Additionally, Ogata [58] proposed a method for sampling stationary Hawkes processes. Chornoboy et al. [15] proved the MLE is an asymptotically efficient estimator without the stationarity condition and established an iterative expectation/minimization (EM) procedure to compute the maximum likelihood estimator.

All of these early works were established for univariate Hawkes processes. Although Hawkes and Oakes [30] described multivariate Hawkes processes in their original paper, noteworthy interest arose only after Eichler et al. [22] showed how Granger causality can be incorporated for a specific class of Hawkes processes. Eichler et al. focused on a class of Hawkes processes with an exponential memory kernel. For this class, the influence of subsequent arrivals on the intensity can be broken into two parts. First of all, parameter $\alpha_{uu'}$ governs the instantaneous increase in probability. Secondly, a decaying function $\phi_{uu'}(\cdot)$ controls the influence of the instantaneous increase over time. In particular, Eichler et al. [22] showed that for Hawkes processes with an exponentially decaying memory kernel $\phi_{uu'}(\cdot)$, Granger causality [23] on process u through process u' is fully encoded in the corresponding instantaneous jump parameter $\alpha_{uu'}$.

Guo et al. [25] proved consistency for the regularized maximum likelihood estimator (MLE) for marked Hawkes processes and provided an alternating minimization type algorithm to compute the respective estimator. Brouste and Farinetto [10] subsequently proposed a fast and asymptotically efficient estimator based on a one-step Le Cam correction of the MLE.

The least squares functional for marked Hawkes processes was first introduced by Reynaud-Bouret et al. [63] and used in the context of genome data. Their estimation method is meant for piece-wise constant memory kernels with finite support. Bacry et al. [4] alternatively used a least squares estimate method which includes dimension reduction via sparsity-inducing regulariztion. They showed superior

[1] See Definition 5.1.15

results, however, their approach is only suitable for a linear combination of exponential memory kernels. Finally, Cartea et al. [11] introduced a fast parametric estimation method for computing the least squares estimator, suitable for scenarios involving a substantial number of arrivals. They demonstrated the application of this estimation approach to a broader family of memory kernels, extending beyond only the exponential memory kernel under consideration in this thesis.

It should be noticed that for the maximum likelihood estimator $\hat{\theta}_{MLE}$ it is shown that the estimator is unbiased, consistent, and the rate of convergence is known. For the least squares estimator $\hat{\theta}_{LSE}$ it is known to be consistent, however, unbiasedness has only been proven in the asymptotic sense when the bias tends to zero as the number of observed points tends to infinity [4].

Part II

Problem framework

3

Descriptive analysis

In this chapter, we describe the IT monitoring data stream generated at different levels of service. The data stream is collected by systems engineers from ING, in order to obtain an overview of the operations of the service systems at hand. The data stream spans a two-year period.

The data stream consists of arrivals characterized by one of two arrival natures: events or incidents. As described in Section 1.2, events are warnings from IT applications that are automatically generated through business rules. Incidents are unplanned interruptions indicative of future system failure.

This chapter is divided into two sections. In Section 3.1, we provide a detailed description of the feature vector of arrivals and explain the process of compressing arrivals into records. Secondly, in Section 3.2, we provide valuable insights into the described features. In particular, we visually present the volume of arrivals, reveal seasonal patterns in the arrival frequency, and offer a breakdown of these arrivals into the service levels and priorities.

3.1. Structure of the arrival data

Within ING's data monitoring data platform, arrivals i are represented as a feature vector R_i together with a time of arrival $t_i^* \in [T_{start}, T_{end}]$. All arrivals together make up a dataset $\mathcal{D}^* = \{(t_i^*, R_i) : i = 1, \ldots, n^*\}$, where n^* denotes the total number of arrivals between T_{start} and T_{end}.
Arrivals are considered between 1 January 2021 and 12 January 2023 where T_{start} := 2021-01-01 and T_{end} := 2023-12-01. Arrivals prior to 1 January 2021 are low in volume and therefore not taken into account in our analysis. Arrivals with at least one missing value are dropped. Additionally, arrivals with a priority of 7, which we describe in Section 3.1.1, are dropped.

Arrivals occur in high volumes, and as a result, when arrivals with an identical feature vector R_i occur within a short time frame, they are compressed into one *record*. We first describe the feature vector R_i in Section 3.1.1 and subsequently elaborate on the rules for compressing arrivals into a record in Section 3.1.2.

3.1.1. Feature vector

Each arrival is associated with one feature vector

$$R_i = (id_i, s_i, level\_id_i, level\_env_i, m_i, prio_i, inc_i, Texp_i).$$

The space of feature vectors is denoted as F. We now describe each of the components of the feature vector.

- Identifier id_i: Identifier for the record to which the arrival is assigned after compression. $id_i \in ID := \{id_1, \ldots, id_n\}$.
- Arrival nature s_i: The arrival nature can either be an event or an incident. $s_i \in S := \{E, I\}$, where E represents events and I represents incidents. As we saw in Section 1.2, the arrival nature characterizes the semantics of the arrival.

- Recorded level identifier $level\_id_i$: Arrivals are recorded on a specific level of service, which can be either on a server or application level. We refer to these levels as *configuration items* or *business applications*, respectively. Notably, a business application encompasses multiple configuration items, indicating that business applications are one layer higher in service level than configuration items [1]. However, there is no feature indicating on which of the two levels the arrival is registered. The recorded level identifier can therefore take values from the set $LEVELS := CI \cup BA = \{ci_1, \dots ci_{n_{ci}}, ba_1, \dots, ba_{n_{ba}}\} = \{1, \cdots, n_{levels}\}$.
- Level environment $level\_env_i$: Environment in which either the configuration item or business application $level\_id_i$ operates. Environments can be production, disaster recovery, acceptance, test, or development. Disaster recovery environments can be interpreted as backup servers. Acceptance environments are environments coming from a test or development phase and are about to be put onto production. Test and development environments are environments that are still in an improvement stage. all in all, $level\_env_i \in ENV := \{PRD, DR, ACC, TST, DEV\}$. We only consider arrivals i for which $level\_env_i \in \{PRD, DR\}$.
- Message m_i: Message providing contextual information regarding arrival i. $m_i \in MSG$, where messages come from formal language $MSG := \{w \in \text{Unicode}^* : |w| \leq 8000\}$. In other words, messages consist of a string of Unicode characters with at most 8000 characters.
- Priority $prio_i$: Priority of arrival, with $prio_i \in PRIO := \{1, 2, 3, 4, 5, 7\}$. Arrivals of priority 1 are seen as most urgent, whereas arrivals of priority 5 are seen as least urgent. Priority 6 does not exist. Priority 7 is assigned if mandatory fields are not properly filled and therefore serves as a remainder bin. Only arrivals i with $prio_i \in PRIO \backslash \{7\}$ are considered in the rest of this thesis. Although priorities for both event and incident arrivals range from 1 to 5, they do have semantically different interpretations. For incident arrivals, the priority is based on the number of customers affected. For event arrivals, the priority is based on the source which automatically generated the arrival.
- Incident identifier inc_i. Identifier of a related record. If no related record is available, a *NaN* value is be reported. $inc_i \in INC := \{NaN\} \cup ID := \{NaN, id_1, \dots id_n\}$. The related incident identifier $inc_i = id_j$ always relates to a record of type *incident*, so $s_j = I$.
- Expiration duration $Texp_i$: Time difference feature used for determining compression of arrivals into records. $Texp \in TEXP :=$ {0 sec, 1 min, 15 min, 6 h, 24 h, 1 week}. Rules for compression are be stated below.

3.1.2. Compression rules

In order to reduce the amount of disk space needed, arrivals are compressed into records. Instead of recording the individual arrival times t^* of all arrivals into one record, only the first arrival time $Tfirst$, the last arrival time $Tlast$, and the number of arrivals within $[Tfirst, Tlast]$, called the $Tally$, are saved. Compression therefore results in the dataset $\mathcal{D} := \textbf{Comp}(\mathcal{D}^*) = \{(Tfirst_i, Tlast_i, Tally_i, R_i) : i = 1, \dots, n\}$ with $n \leq n^*$. The disadvantage of compression is that most of the original arrival times are lost.

The temporal features of a record therefore become:

- First arrival time $Tfirst_i$: First arrival time of all arrivals with the record. $Tfirst_i \in [T_{start}, T_{end}]$.
- Last arrival time $Tlast_j$: Last arrival time of all arrivals with the record. $Tlast_i \in [T_{start}, T_{end}]$.
- Tally $Tally_i$: Number of arrivals within the time interval $[Tfirst_i, Tlast_i]$ for record i.

To determine how $\mathcal{D}^*$ is be compressed in $\mathcal{D} = \textbf{Comp}(\mathcal{D}^*)$, two rules are in place. The first rule utilizes the expiration duration. If an arrival (t_i^*, R_i) is observed, a new record is created with a counter of 1. The arrival time of arrival i, t_i^*, is set as the first arrival time $Tfirst$ of the new record. If within a period of $Texp_i$ seconds, an arrival j is observed such that $R_i = R_j$, the counter is incremented. If for a period of $Texp_i$ seconds, no identical arrival occurs, the counter is stopped, and the last observed arrival time t_k^* is taken as being the last arrival time $Tlast$ of the new record. The final result of the counter is then taken as the tally value $Tally_i$. An illustration of this process is sketched in Figure 3.1.

Secondly, handcrafted rules determined by ING domain experts are set. Those rules include a cutoff at an exact point in time or a cutoff after a fixed time interval, even if new arrivals occur within the $Texp_i$.

[1] A detailed Definition of a configuration item and a business application is provided in Definition 4.1.1 and 4.1.2.

The exact characteristics of the second rule are not known to us.

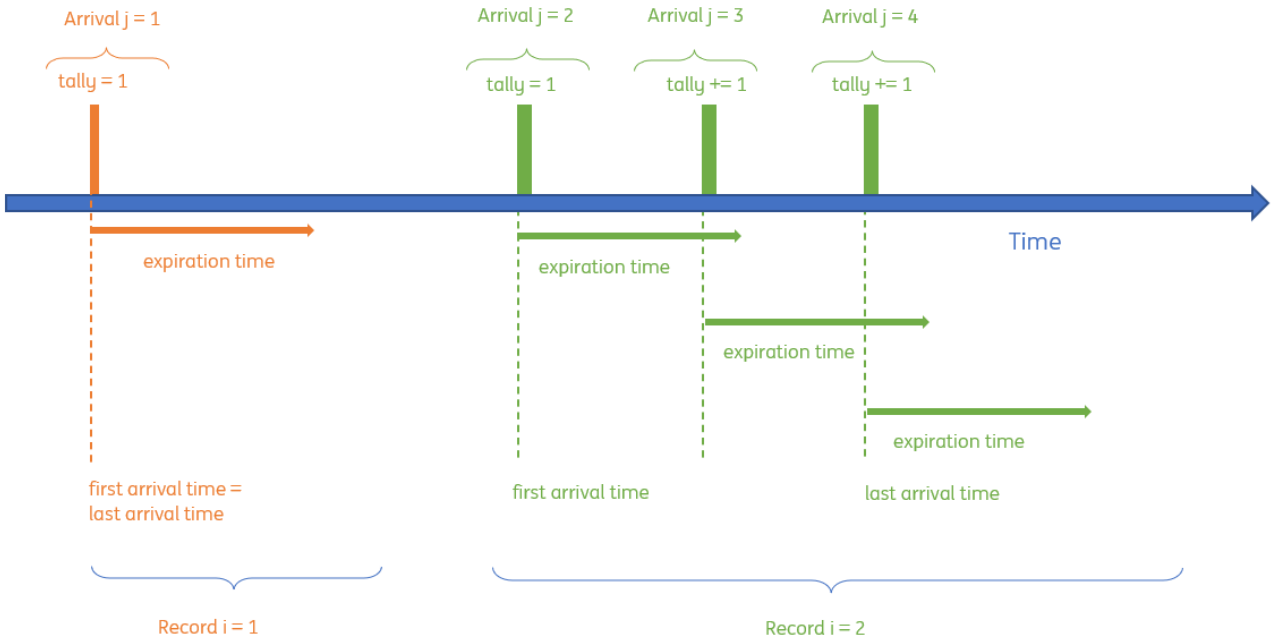

Figure 3.1: Example of $n* = 4$ arrivals and $n = 2$ records to demonstrate the characteristics of the expiration duration.

3.2. Feature analysis

Depending on whether $s_i = E$ or $s_i = I$, the characteristics of record i vary. Events (for which $s_i = E$) arrive in a larger volume than incident arrivals. Due to this high volume, only event arrivals require compression. An incident arrival is never compressed and therefore an incident record consists of one incident arrival only. This implies that for all records of type *incident*, $Tfirst_i = Tlast_i$, $Tally_i = 1$ and $Texp_i = 0$. Furthermore, this implies that speaking of an incident arrival is identical to speaking of an incident record. We can therefore simply refer to the former as being *an incident*.

As a necessary first step in correlating events and incidents, it is of interest to compare the temporal occurrences of event and incident arrivals. Unfortunately, due to the merging of event arrivals, we do not observe the exact arrival times of individual events. Instead, we only keep the first arrival time, last arrival times, and the tally. From the event record, we know that there must have been at least one arrival at $Tfirst_i$ and one arrival at $Tlast_i$, which may be the same.

In the following, we only use the first arrival time of each record.

3.2.1. Temporal occurrence

The number of first arrival times for event and incident records within a Δt-time interval of $[T_{start}, T_{end}]$ can be found using the following mapping:

Definition 3.2.1: Cardinality of binned records

Let $t \in [T_{start}, T_{end}]$, let Δt be any positive amount of time expressed in minutes and $s \in S$. Let $N^{BIN}_{t,\Delta t,s} = \boldsymbol{Card}(i : Tfirst_i \in [t, t + \Delta t), s_i = s)$ denote the cardinality of records with a given s in interval $[t, t + \Delta t)$.

Let T be the set consisting of all 10-minute rounded times between T_{start} = 2021-01-01 00:00 to T_{end} =

2023-01-13 00:00. The set of 10-minute rounding times therefore becomes

$$T = \{\text{2021-01-01 00:00},\ \text{2021-01-01 00:10},\ \ldots, \text{2023-01-12 23:50}\}.$$

Figure 3.2 and 3.3 show a visual representation of $t \to N^{BIN}_{t,10min,E}$ and $t \to N^{BIN}_{t,10min,I}$ for all $t \in T$, for both event and incident records, respectively.

Seasonality

Although not directly visible from Figures 3.2 and 3.3, the number of event and incident arrivals shows a seasonal behavior. The number of arrivals within a 10-minute interval $N^{BIN}_{t,10min,\cdot}$ can therefore be fitted following a linear regression model in order to obtain deeper insights into the seasonality. This can be done for both event and incident arrivals. We therefore fit

$$\log_{10}\left(N^{BIN}_{t,10min,\cdot}\right) = \beta_0 + \beta_1 t + \beta_2 \sin(Wt) + \beta_3 \cos(Wt) + \beta_4 I_{\{t\in\text{weekend}\}}, \tag{3.1}$$

with $\boldsymbol{\beta} := \{\beta_0, \beta_1, \beta_2, \beta_3, \beta_4\} \subset \mathbb{R}^5$. The period W is chosen using a periodogram on which the peak frequency was around 24 hours for both event and incident records. Secondly, a smaller peak at 7*24 hours is observed. Additionally, it can be observed the number of arrivals during the weekend is lower, so, therefore, it makes sense to add a weekend indicator β_4. These findings resonate with the understanding that arrivals exhibit a daily cycle in accordance with a standard workday schedule. The estimates for $\boldsymbol{\beta}$, namely $\hat{\boldsymbol{\beta}}$ can be found in Table 3.1. The periodograms can be found in Appendix C.2.

In order to visualize the seasonal behavior, we zoomed in on the 2-month time interval from 1 September 2022 to 1 November 2022. The subset of 10-minute rounding times $T_{sub} \subset T$ therefore becomes

$$T_{sub} = \{\text{2022-09-01 00:00},\ \text{2022-09-01 00:10},\ \ldots, \text{2022-10-31 23:50}\},$$

with $T_{start,sub}$ = 2022-09-01 00:00 to $T_{end,sub}$ = 2022-11-01 00:00. The fitted function zoomed in for this 2-month time interval can be found in Figures 3.6 and 3.7. For incident records, the adjusted R-squared equals 0.103, whereas, for the event records, the adjusted R-squared equals 0.0250. This difference can be explained by the fact that event arrivals are tallied into records, whereas incident arrivals are not. As we compress similar event records occurring within a specific time interval, events that happen a lot are under-represented. This is because we only use their first arrival time. Similarly, event arrivals that are rare are now over-represented. Therefore we have introduced a bias. This explains why the estimates $\hat{\beta}_2$ and $\hat{\beta}_3$ regarding seasonality have an insignificant p-value.

Table 3.1: Estimated values for Equation (3.1) for event and incident records.

	Event records			**Incident records**		
	Estimate	**Std. Error**	**p-value**	**Estimate**	**Std. Error**	**p-value**
$\hat{\beta}_0$	8.94 e-01	8.13 e-03	**<2 e-16**	1.06 e+00	6.83 e-03	**<2 e-16**
$\hat{\beta}_1$	-1.02 e-09	2.50 e-09	0.685	2.43 e-08	2.11 e-09	**<2e-16**
$\hat{\beta}_2$	-4.78 e-03	5.47 e-03	0.382	1.84 e-02	4.60 e-03	**6.38 e-05**
$\hat{\beta}_3$	-7.96 e-02	5.47 e-03	**<2 e-16**	-1.30 e-01	4.60 e-03	**< 2 e-16**
$\hat{\beta}_4$	-4.51 e-02	8.53 e-03	**1.23 e-07**	-5.32 e-02	7.17 e-03	**1.27 e-13**

Volume

To more closely capture the volume of records, we present a histogram for the number of arrivals per time interval in Figures 3.4 and 3.5. These histograms show $\boldsymbol{Card}(t : N^{BIN}_{t,10min,s} = i)$ for $t \in T$ and $s \in S$. It can be seen the majority of event and incident bins contain a low volume of records, however, for both the event and incident bins, outlier values can be found for bins exceeding a thousand arrivals for a 10-minute interval.

Authors note: This paragraph is removed due to confidentiality.

Remark 3.2.2. *Plotly* does not provide logarithmically scaled heatmaps. Custom-made code to provide this functionality can be found in Appendix A.

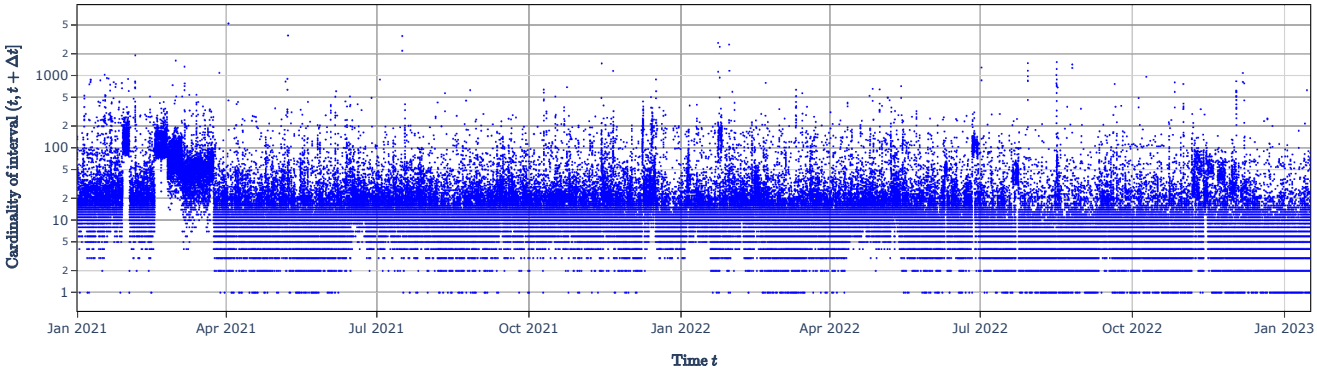

Figure 3.2: Scatter plot of the function $t \to N^{BIN}_{t,10min,E}$ for $t \in [T_{start}, T_{end}]$.

Authors note: This figure is removed due to confidentiality.

Figure 3.3: Scatter plot of the function $t \to N^{BIN}_{t,10min,I}$ for $t \in [T_{start}, T_{end}]$.

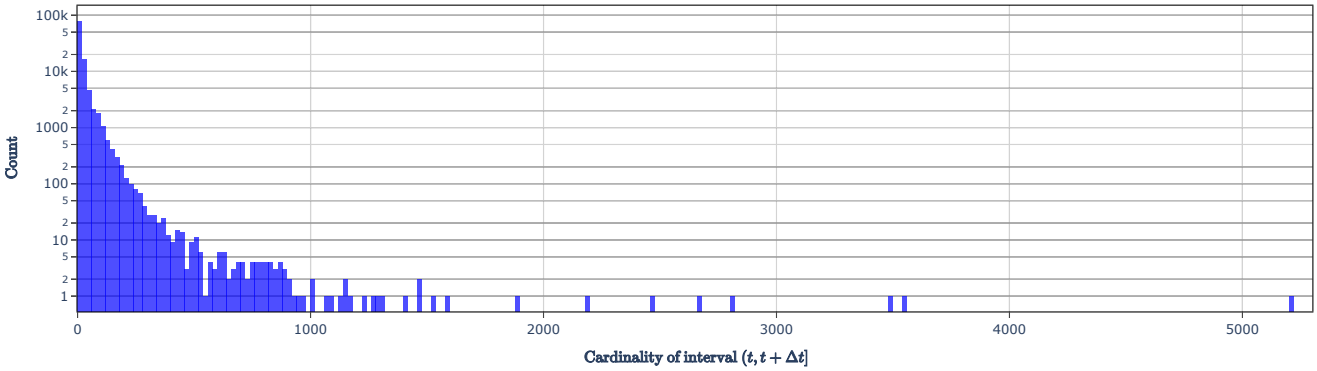

Figure 3.4: Histogram of $\left(N^{BIN}_{t,10min,E}\right)_{t\in[T\_start,T_{end}]}$. The height of each bar equals $\boldsymbol{Card}(t \in [T_{start}, T_{end}] : N^{BIN}_{t,10min,E} \in [N, N + \Delta N])$ for $N = 0$ to $N = 5500$ by step $\Delta N = 20$.

Authors note: This figure is removed due to confidentiality.

Figure 3.5: Histogram of $\left(N^{BIN}_{t,10min,I}\right)_{t\in[T\_start,T_{end}]}$. The height of each bar equals $\boldsymbol{Card}(t \in [T_{start}, T_{end}] : N^{BIN}_{t,10min,I} \in [N, N + \Delta N])$ for $N = 0$ to $N = 5500$ by step $\Delta N = 5$.

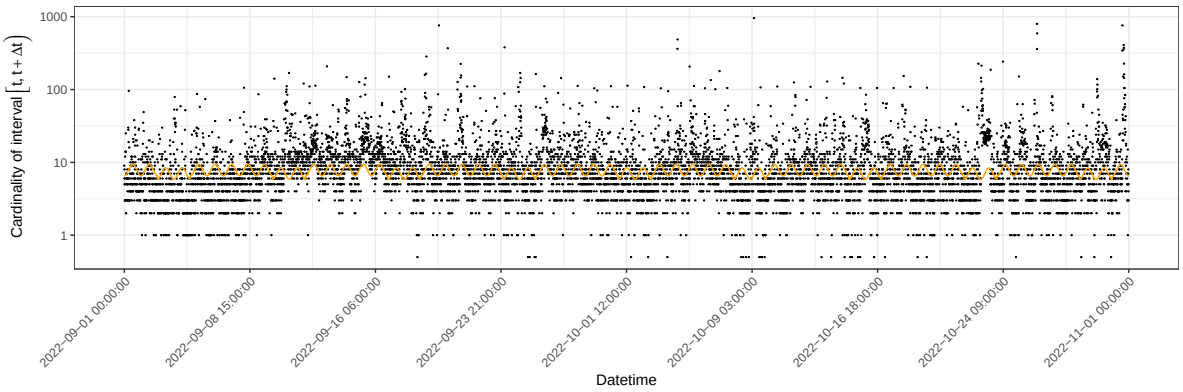

Figure 3.6: Scatter plot of the function $t \longrightarrow N^{BIN}_{t,10min,E}$ for $t \in [T_{start,sub}, T_{end,sub}]$, including seasonal fit of function 3.1.

Authors note: This figure is removed due to confidentiality.

Figure 3.7: Scatter plot of the function $t \longrightarrow N^{BIN}_{t,10min,I}$ for $t \in [T_{start,sub}, T_{end,sub}]$, including seasonal fit of function 3.1.

Tally and expiration duration

Finally, it is of interest to analyze the tally and expiration duration. The expiration duration for event records can be found in Table 3.2. Notably, the table reveals that the vast majority of event records have an expiration duration of 24 hours. It is important to reiterate that incident arrivals are not tallied, resulting in all incident records having an expiration duration of 0 seconds and a tally of 1.

Table 3.2: Mapping of $k \in TEXP$ to $N^{Texp}_k = \boldsymbol{Card}(i : Texp_i = k)$.

$k \in TEXP$		N^{Texp}_k
0	seconds	338
1	minute	199
15	minutes	579
6	hours	10.720
24	hours	2.422.489
1	week	393

The distribution of the tally is visually depicted in Figure 3.10 through a histogram. The top figure highlights the presence of a few outlier tally counts, with the largest tally surpassing 4.7 million. Meanwhile, the bottom figure presents the same histogram but with logarithmically scaled bin sizes, illustrating that the tally value counts decrease exponentially.

3.2.2. Level identifiers

The levels can either be a configuration item or a business application. We will elaborate on the characteristics of configuration items and business applications in Section 4.1. For now, we can interpret a configuration item as a server, whereas we interpret a business application as an IT application utilizing multiple servers. It is of interest to see how the cardinality of event and incident records is distributed amongst the n_{levels}. For this, we denote the cardinality as follows:

Definition 3.2.3: Cardinality of records per level item

Let $k \in LEVELS$ and $s \in S$. Let $N^{LEVEL}_{k,s} = \boldsymbol{Card}(i : levels\_id_i = k, s_i = s)$. $N^{LEVELS}_{k,s}$ denotes the cardinality of records with a given level k and a given nature s.

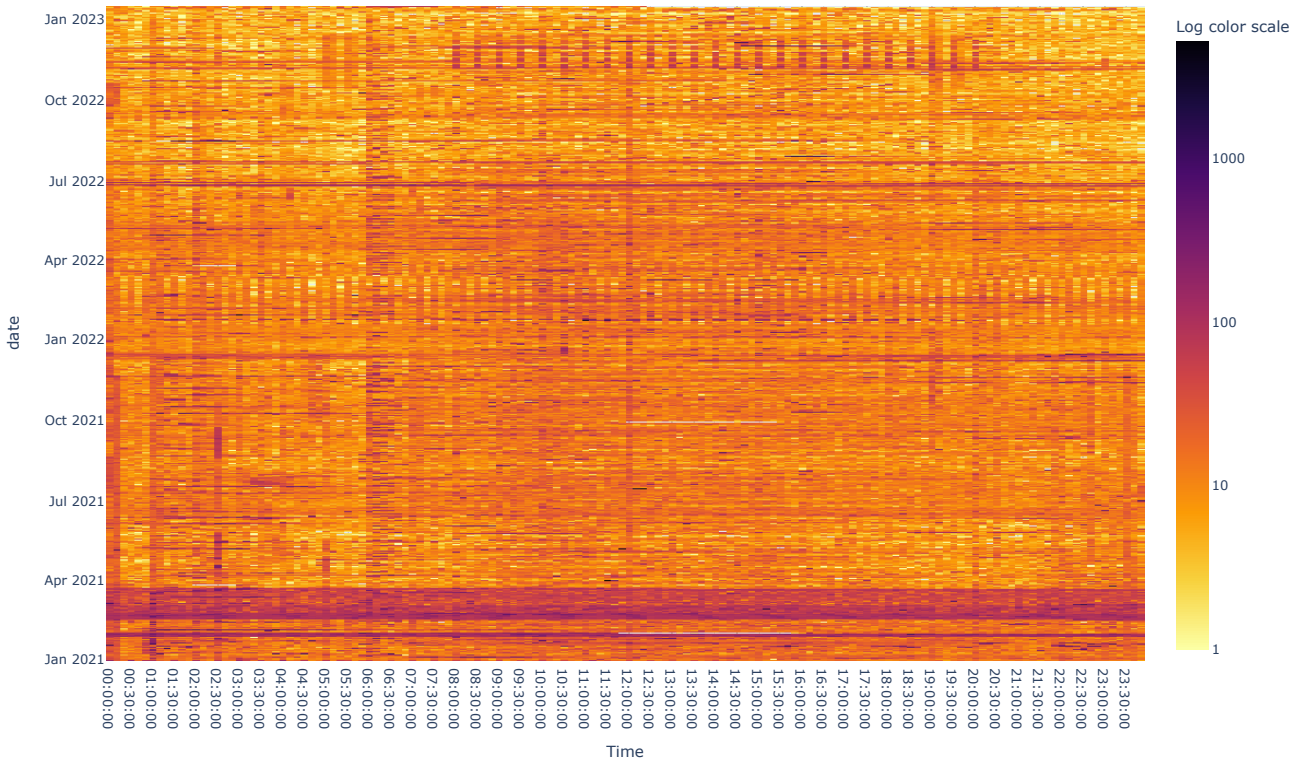

Figure 3.8: Event records heatmap for $\left(N^{BIN}_{t,10min,E}\right)_{t\in[T\_start,T_{end}]}$.

Authors note: This figure is removed due to confidentiality.

Figure 3.9: Incident records heatmap for $\left(N^{BIN}_{t,10min,I}\right)_{t\in[T\_start,T_{end}]}$.

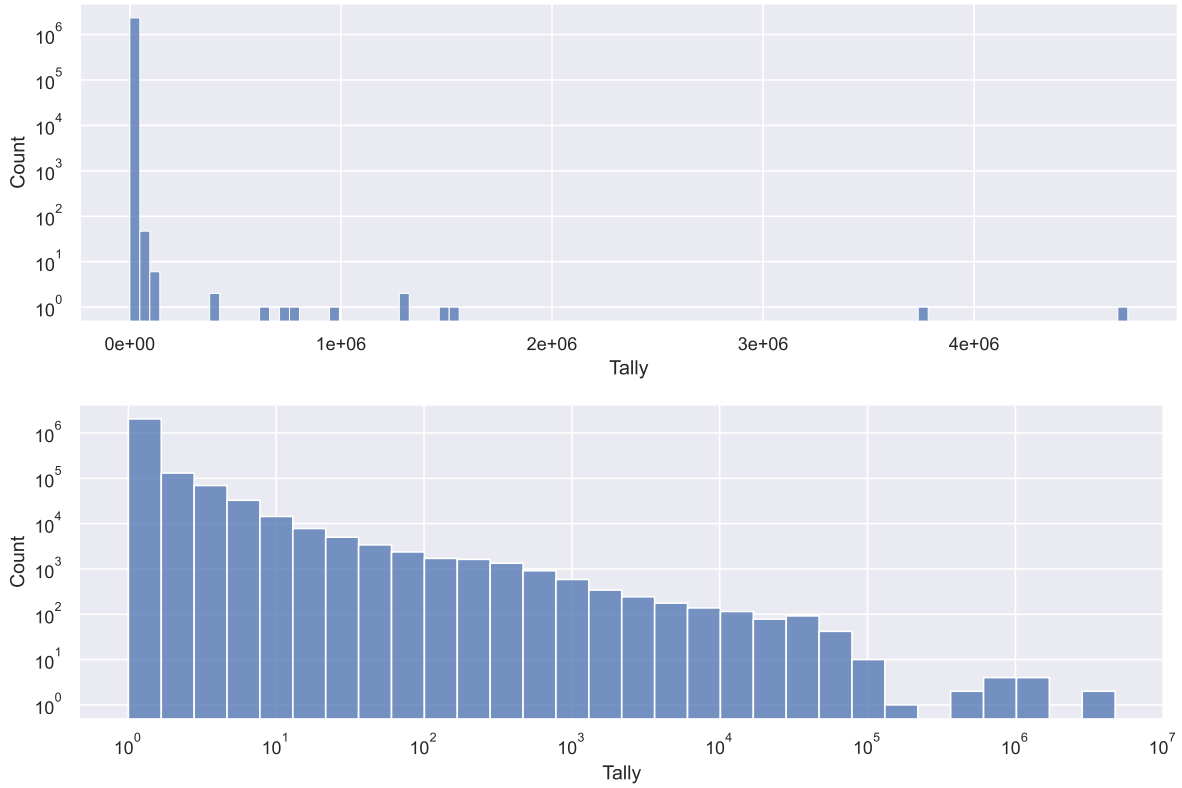

Figure 3.10: Histogram of $Tally_i$ for each record $i \in \{1, \cdots, n\}$. The top figure shows linear bin sizes whereas the bottom shows logarithmically scaled bin sizes.

We are now interested in finding out if for fixed nature $s \in S$, there is a substantial difference between $N_{lvl,s}$ for $lvl \in LEVELS$. We can recursively define a non-increasing function of cardinalities.

Definition 3.2.4: Cardinality function per configuration item

For every $k \in LEVELS := \{CI \cup BA\} = \{ci_1, \ldots ci_{n_{ci}}, ba_1, \ldots, ba_{n_{ba}}\}$, for fixed $s \in S = \{I, E\}$ we define

$$\tilde{N}_{1,s}^{LEVEL} := \max\{N_{k,s}^{LEVEL} : k \in LEVELS\},$$
$$\tilde{N}_{k,s}^{LEVELS} := \max(\{N_{k,s}^{LEVEL} : k \in LEVELS\} \backslash \{\tilde{N}_{1,s}^{LEVEL}, \ldots, \tilde{N}_{k-1,s}^{LEVEL}\}).$$

$\tilde{N}_{k,s}^{LEVEL}$ can be interpreted as the number of records of nature s observed on the level with the k^{th} most records of nature s. From Figure 3.12, it can be seen that both event and incident records are not at all uniformly distributed over the levels. The figure shows that the cardinality $N_{k,s}^{LEVEL}$ for both events and incidents can range from only 1 up to 10.000 records depending on the level $k \in LEVELS$.

Furthermore, from the tail behavior, it can be seen that the number of levels containing at least one incident exceeds the number of levels containing at least one event. This implies that there are levels containing at least one incident arrival, but containing no event arrivals. This is an important observation as it indicates that when making predictions on an item level, there will with certainty be incident records that can not be associated with any event record, on their respective level. Later on in Section 4.1.2, we see that the other way around there does also exist levels with only event records and no incident records. It should be noted that the levels for which $N_{k,E}^{LEVEL} = 0$ are not displayed in Figure 3.11 because of the logarithmic scale.

Authors note: This figure is removed due to confidentiality.

Figure 3.11: Log-log plot for the non-increasing function $k \to \tilde{N}_{k,E}^{LEVEL}$ (in orange) and $k \to \tilde{N}_{k,I}^{LEVEL}$ (in blue). k for which $N_{k,E}^{LEVEL} = 0$ are not displayed.

Authors note: This figure is removed due to confidentiality.

Figure 3.12: Log-log plot for the non-increasing function $k \to \tilde{N}_{k,E}^{MSG}$ (in orange) and $k \to \tilde{N}_{k,I}^{MSG}$ (in blue).

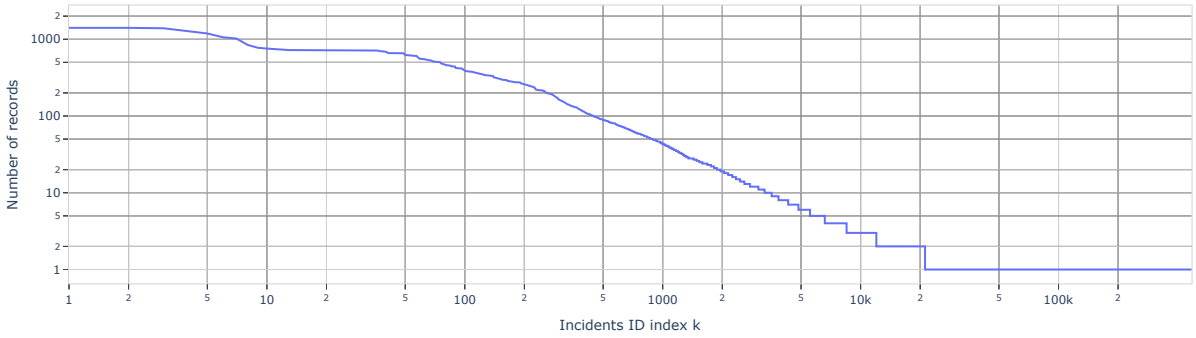

Figure 3.13: Log-log plot for the non-increasing function $k \to \tilde{N}_{k}^{ID}$.

3.2.3. Message structure

Similarly as in Definition 3.2.3, we define the cardinality per message for each arrival nature.

Definition 3.2.5: Cardinality of records per {arrival nature, message} pair

Let $k \in MSG$ and $s \in S$. Let $N_{k,s}^{MSG} = \boldsymbol{Card}(i : m_i = k, s_i = s)$. $N_{k,s}^{MSG}$ denotes the cardinality of records with a given message k and a given arrival nature s.

It is of interest to see if, for a fixed $s \in S$, there is a substantial difference between $N_{k,s}^{MSG}$ for $k \in MSG$. The function of cardinalities is defined accordingly.

Definition 3.2.6: Cardinality function per message

For every $k \in MSG := \{1, \ldots, n_{msg}\}$, for fixed $s \in S = \{I, E\}$ we define

$$\tilde{N}_{1,s}^{MSG} := \max\{N_{k,s}^{MSG} : k \in MSG\},$$
$$\tilde{N}_{k,s}^{MSG} := \max(\{N_{k,s}^{MSG} : k \in MSG\} \backslash \{\tilde{N}_{1,s}^{MSG}, \ldots, \tilde{N}_{k-1,s}^{MSG}\}).$$

From Figure 3.12, it can be found that identical event messages reoccur more often than identical incident messages. This observation can be partially attributed to the fact that event messages are automatically generated and adhere to specific templates, which are then filled in with variables. In contrast, incident messages may be manually generated, leading to variations in their structure, thereby reducing the frequency of identical occurrences.

Remark 3.2.7. It should be noted that different messages might contain a semantically similar structure. Take for instance the messages `Server ABC down` and `Server XYZ down`. We can group those semantically similar messages and extract a template and parameters out of these messages. In the example of the prior two messages, the template is `Server <*> down`, together with parameters `(ABC, XYZ)`. In Section 4.2.1, we see how parsing the messages can be used to group semantically similar messages together.

3.2.4. Incident identifier

As a result of root cause mitigation from ING's system engineers, records i of type $s_i = E$ (i.e. events), can be manually associated to one record j of type $s_j = I$ (i.e. an incident). This manual association occurs when system engineers have a strong indication that a particular event record i triggered the incident record j. The association is recorded by specifying $inc_i := id_j$, where each event record can be linked to at most one incident. In cases where no incident record is linked, $inc_i :=$ NaN.
To gain an overview of the cardinality of associated event records for each incident record, we define again a cardinality function.

Definition 3.2.8: Cardinality of records per incident identifer

Let $k \in ID$. Let $N_k^{ID} = \boldsymbol{Card}(i : inc_i = k)$. N_k^{ID} denote the cardinality of records with a given identifier k.

Definition 3.2.9: Cardinality function per incident id

For every $k \in ID := \{id_1, \ldots, id_n\}$ such that $s_i = I$, we define

$$\tilde{N}_1^{ID} := \max\{N_k^{ID} : k \in ID\},$$
$$\tilde{N}_k^{ID} := \max(\{N_k^{ID} : k \in ID\} \backslash \{\tilde{N}_1^{ID}, \ldots, \tilde{N}_{k-1}^{ID}\}).$$

Analyzing the incident records, we find that approximately 20.2% of them have at least one associated event record. Among these incident records with associations, about 95.5% are linked to only one event

record.

Although the majority of incident records have a low number of event associations, some exceptional cases reveal a substantial number of event records being associated with a single incident. In the most extreme situation, $\tilde{N}_1^{ID}$ indicates that over a thousand event records are linked to a single incident record.

On the other hand, from the perspective of events, around 69.4% of event records do not have an associated incident record (i.e., inc_i = NaN). As we proceed with our analysis, the priority information is not directly employed. Nonetheless, in the future work section, we elaborate on how leveraging the priority data could potentially yield improved outcomes.

3.2.5. Differentiating between priorities

Both event and incident records i contain a priority $prio_i$. A histogram for the distribution of priorities $\boldsymbol{Card}(i : prio_i = prio, s_i = s)$ for $prio \in PRIO$ and $s \in S$ is given in figure 3.14

Authors note: This figure is removed due to confidentiality.

Figure 3.14: Histogram for the distribution of priorities $\boldsymbol{Card}(i : prio_i = prio, s_i = s)$ for $prio \in PRIO$ and $s \in S$.

We once again generated the cardinality functions discussed in Sections 3.2.2, 3.2.3, and 3.2.4. However, this time, we distinguished the records based on their five distinct priorities. The corresponding outcomes for event records can be found in Figures 3.15, 3.16, and 3.17. The results for incident records are presented in Figures 3.18 and 3.19.

3.2.6. Pareto principle

The Pareto principle, or 80-20 law, is a familiar saying that 80% of outcomes (or outputs) result from 20% of all causes (or inputs). For instance, Microsoft has discovered that 80% of the errors and crashes in Windows and Office are caused by 20 % percent of the entire pool of bugs detected [67]. From the log-log plots in Figures 3.11 to 3.13 and 3.15 to 3.19, we can recognize regions satisfying the Pareto principle. These regions are characterized by a linearly decreasing segment. This linearly decreasing segment is called the *signature* of a power law [66].

The Pareto principle regions satisfies a power-law relation,

$$f(x) = ax^{-b}. \tag{3.2}$$

For the counting data from Figures 3.11 to 3.13 and 3.17 to 3.19, the relation would therefore become

$$\tilde{N}_k^{\cdot} \approx ak^{-b}, \tag{3.3}$$

$$\log(\tilde{N}_k^{\cdot}) \approx \log(a) - b \cdot \log(k), \tag{3.4}$$

where k is the index, $\tilde{N}_k^{\cdot}$ is the number of observations at index k, $\log(a)$ is a normalizing constant and b is the slope. It should be noticed that in practice such straightness is a necessary and not a sufficient condition of following a power law distribution. Therefore it should be noted that it is not our goal to prove any of these cardinality functions corresponds to a power law, but merely to highlight the intriguing similarities to a power law. Actually validating power-law models should be considered a research question on its own, see for example [31].

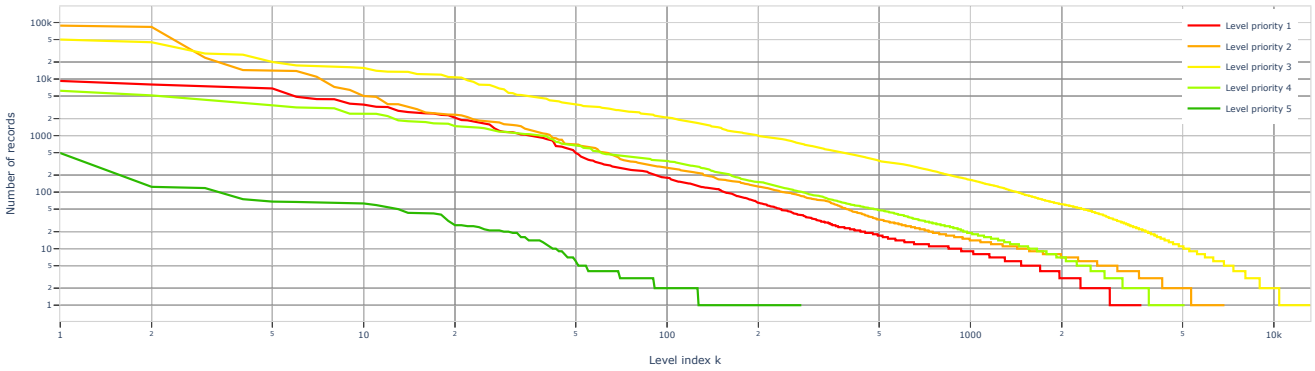

Figure 3.15: Log-log plot for the function $k \to \tilde{N}_{k,E}^{LEVEL}$, differentiating per priority.

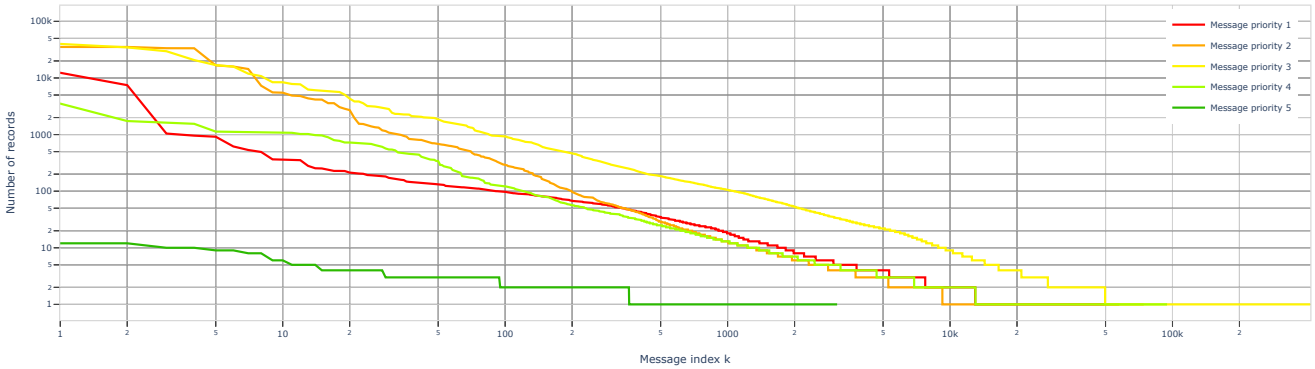

Figure 3.16: Log-log plot for the function $k \to \tilde{N}_{k,E}^{MSG}$, differentiating per priority.

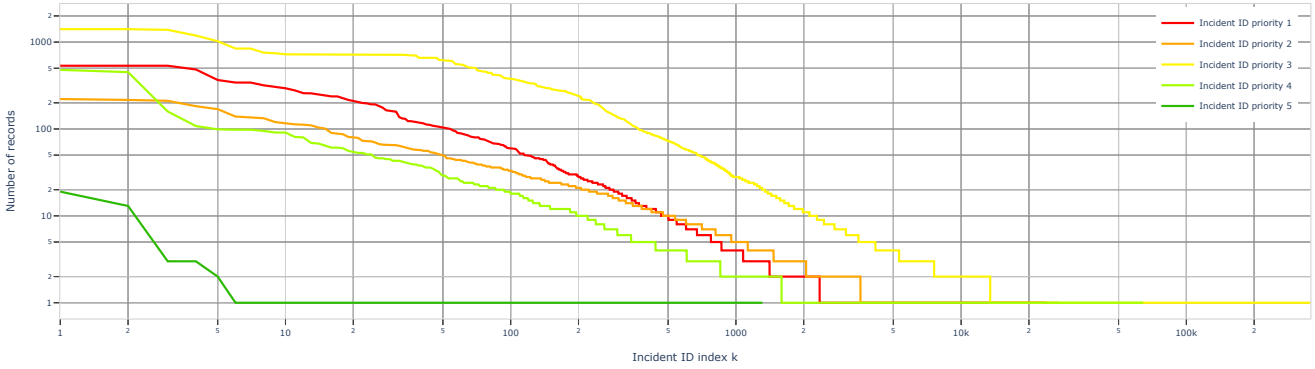

Figure 3.17: Log-log plot for the function $k \to \tilde{N}_{k}^{ID}$, differentiating per priority.

Authors note: This figure is removed due to confidentiality.

Figure 3.18: Log-log plot for the function $k \to \tilde{N}^{CI}_{k,I}$, differentiating per priority.

Authors note: This figure is removed due to confidentiality.

Figure 3.19: Log-log plot for the function $k \to \tilde{N}^{MSG}_{k,I}$, differentiating per priority.

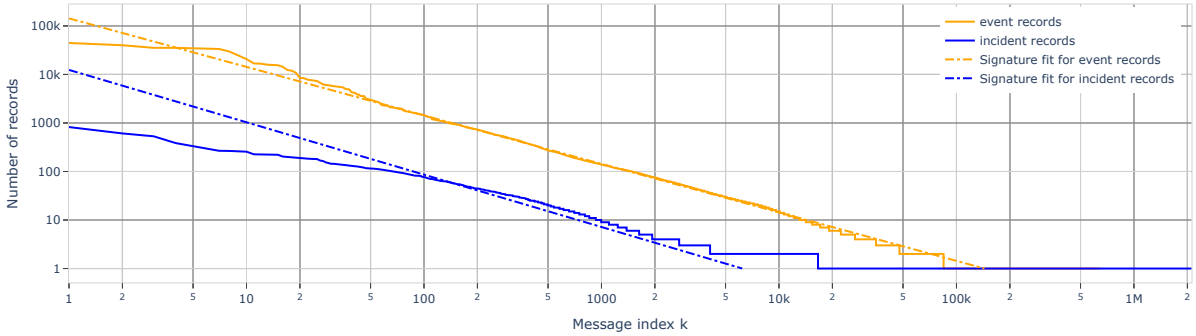

Figure 3.20: Log-log plot for the non-increasing functions $k \to \tilde{N}^{MSG}_{k,E}$ (in orange) and $k \to \tilde{N}^{MSG}_{k,I}$ (in blue). Power law functions (3.4) are fitted for event messages $\tilde{N}^{MSG}_{k,E}$ (respectively for incident messages $\tilde{N}^{MSG}_{k,I}$) on the interval [20, 3000] (resp. [100, 1000]).

As an example, we fit Equation (3.4) to the cardinality function per message from Definition 3.2.6 for the cardinality of event and incident messages. We estimated a and b by means of an ordinary least squares estimate. To reduce noise in the tails, we fitted the power law function from 3.4 on indices $k \in [20, 3000]$ for $\tilde{N}^{MSG}_{k,E}$ and $k \in [100, 1000]$ for $\tilde{N}^{MSG}_{k,I}$. The result can be found in Figure 3.20 and Table 3.3.

Table 3.3: Fitting a power law for $\log(\tilde{N}^{MSG}_{k,E}) = \log(a) - b \cdot log(k)$ and $\log(\tilde{N}^{MSG}_{k,I}) = \log(a) - b \cdot log(k)$.

	Range for k	$\log(\hat{a})$	$\hat{b}$	**Percentage of arrivals attained in first 20% of messages**
Event message	[20, 3000]	11.8	1.00	78%
Incident message	[100, 1000]	9.42	1.08	22%

Finally, it can be found in the last column of Table 3.3, that the 20% most frequent event messages appear in 78% of all event records, abiding by the Pareto principle. In contrast, incident messages do not follow the Pareto principle, since the 20% most frequent incident messages appear in only 22% of all incident records.

In Section 1.2, we discussed how event records are generated automatically, while incident records are created manually. Consequently, it is not surprising that the automated generation process leads to more structured messages, causing 20% of the most frequent event messages to account for 78% of all event records. In contrast, the manual creation of incident messages introduces writing style variability, leading to a scenario where 20% of the most frequent incident messages account for only around 22% of all incident messages.

4

Hierarchical service architecture

As seen in Section 3.1, arrivals are recorded on different levels of service, namely configuration items or business applications. In this chapter, we introduce the level on top of the configuration item and business application. This level is named the *business unit*. Additionally, we show how we can construct the lowest level of service, by creating clusters based on the arrivals message. Consequently, the resulting hierarchical service architecture comprises five levels:
(1) message cluster → (2) arrival nature → (3) configuration item → (4) business application → (5) business unit.

This chapter is divided into three sections. In Section 4.1.1, we provide definitions for the configuration item, business application, and business unit. Moreover, we analyze the mapping between the configuration item and the business application and establish a five-level hierarchical service architecture. Secondly, in Section 4.1.1 we describe the procedure of grouping messages together. For this purpose, we convert the messages to a numerical vector, reduce the dimension of the obtained vector, and cluster them together using a clustering algorithm. Finally, in Section 4.3 we apply the established clustering procedure from the previous section on the IT monitoring data stream from ING. Moreover, we show sensitivity to each of the clustering results with regard to individual hyperparameters.

In the first section of this chapter, we are concerned with answering Research Question 1

Research Question 1

How can we design a hierarchical architecture that resembles the operations of large-scale service systems?

4.1. Top level service structure

First of all, it is of interest to properly define the highest three levels in the service architecture, namely the configuration item (level 3), business application (level 4), and business unit (level 5).

Definition 4.1.1: Configuration item [34] (level 3)

A configuration item (CI) is any service component, infrastructure element, or other items that need to be managed in order to ensure the successful delivery of services.

Configuration items vary in complexity, size, and type. They can range from an entire service, which may consist of hardware, software, and documentation, to a single program module or a minor hardware component. Due to this variability, the number of events and incidents recorded on a single configuration item might vary substantially. Their definition is intentionally broad so that a wide range of industries can model things they change or manage with the same set of tools [72]. For ease of interpretation, we interpret a configuration item as a server. An example of a configuration item is a server saving all

existing *IBAN account numbers* of ING customers. This server can for instance be exploited to verify the existence of a specific account number.

Definition 4.1.2: Business application [33] (level 4)

A business application (BA) is defined as a logical entity, served by a set of software components implementing a coherent set of business functions that support one or more businesses or IT processes managed as a whole.

A business application is often delivered by multiple components: APIs, front-end features, and databases. A business application can simply be interpreted as a software system. An example of a business application is the *iDeal payment system*. iDeal is a Netherlands-based payment method that allows customers to complete transactions online using their bank credentials [77].

Definition 4.1.3: Business unit [75] (level 5)

A business unit (BU) is a fully functional, independently operational setup of a particular domain of a company. These units have their vision, growth, and direction.

ING Group consists of multiple financially separated business units. These units are characterized by the country and banking segment in which they operate. An example of such a unit is *wholesale banking Belgium*, which is responsible for services sold to large clients, such as other banks or large financial institutions [40]. Because employees in each unit use software typical for their business unit, software for each business unit is managed separately [35]. A business unit can be seen as an isolated environment of operations. The business unit is often referred to as the *area*.

4.1.1. Structure of the service data

As stated in section 3.1, event and incident records are registered on either a configuration item or business application level. To be able to associate records on all three service levels, a service mapping table from ING is used. The service mapping table consists of mapping vectors

$$Q_j = (bu_j, ba\_name_j, ba_j, ci\_name_j, ci_j).$$

The space of service vectors is denoted as G. Let us now proceed to outline each of the components comprising the feature vector.

- Business unit identifier $bu_j \in BU := \{bu_1, \ldots, bu_{n_{bu}}\}$,
- Business application identifier $ba_j \in BA := \{ba_1, \ldots, ba_{n_{ba}}\}$,
- Business application name $ba\_name_j \in BA\_NAMES := \{ba\_name_1, \ldots, ba\_name_{n_{ba\_name}}\}$,
- Configuration item identifier $ci_j \in CI := \{ci_1, \ldots ci_{n_{ci}}\}$,
- Configuration item name $ci\_name_j \in CI\_NAMES := \{ci\_name_1, \ldots, ci\_name_{n_{ci\_name}}\}$.

For both configuration items and business applications, an *identifier*, as well as a *name* exists. Due to inconsistencies in naming conventions at ING, situations arise where multiple names are linked to one identifier. Therefore $n_{ba} \leq n_{ba\_names}$ and $n_{ci} \leq n_{ci\_names}$.

Furthermore, event records are associated with a name, whereas incident records are associated with an identifier. Therefore, for all $R_i : s_i = E$, $\{level\_id_i \cup BA = \emptyset$ and $level\_id_i \cup CI = \emptyset\}$. Additionally, for all $R_i : s_i = I$ we obtain $\{level\_id_i \cup BA\_NAMES = \emptyset$ and $level\_id_i \cup CI\_NAMES = \emptyset\}$.

To be able to perform inference on the measure, we transform the level name to a level identifier for event records. In other words, for $R_i : s_i = E$, we first look up $level\_id_i \in BA\_NAMES$. If we find $level\_id_i = ba\_name_j$ for $ba\_name_j \in BA\_NAMES$, we overwrite $level\_id_i = ba_j$ for $ba_j \in BA$. If $level\_id_i \notin BA\_NAMES$, we try to find $level\_id_i = ci\_name_j$ for $ci\_name_j \in CI\_NAMES$ and subsequently overwrite $level\_id_i = ci_j$ for $ci_j \in CI$. Therefore the feature $level\_id_i$ in Section 3.1 always refers to an identifier.

Remark 4.1.4. In cases for event records where the level identifier $level\_id_i$ is not found in both BA_NAMES as well as CI_NAMES, we discard the record. This is because we simply have no other option of mapping the event record to an identifier that is consistent with the incident labeling. It should be noted this might have introduced bias.

4.1.2. Analysis of mapping

Prior, we defined the set of configuration items $CI = \{ci_1, \dots ci_{n_{ci}}\}$ and business applications $BA = \{ba_1, \dots, ba_{n_{ba}}\}$, which together make up the record level identifiers $LEVELS = CI \cup BA := \{ci_1, \dots ci_{n_{ci}}, ba_1, \dots, ba_{n_{ba}}\}$. For a strictly hierarchical structure, the relation between business applications and configuration items should be *one-to-many*, meaning a business application may be linked to many configuration items, but a configuration item is linked to only one business application. In practice, we see undesirable situations arise where a configuration item is linked to multiple business applications. In Section 6.3.3, we inspect different options at hand to recover the *one-to-many* relation. For now, we analyze the magnitude of this undesirable mapping.

After joining feature vectors R_i on $level\_id_i$ with service records Q_j on either $ci_j \in CI$ or $ba_j \in BA$, five situations can arise;

1. $level\_id_i = ci_j$ **for exactly one** $ci_j \in CI$**;**. In this case, we resort to the mapping vector Q_j and directly assign bu_j and ba_j to record i.
2. $level\_id_i = ci_j$ **for more than one** $ci_j \in CI$**;**. In this case, we obtain multiple mapping vectors $Q_j, \cdots, Q_{j'}$. Therefore, we can assign multiple business applications $bu_j, \dots, bu_{j'}$. In Section 6.3.3, we specify which choice to make.
3. $level\_id_i = ba_j$ **for exactly one** $ba_j \in BA$; In this case, we resort to the mapping vector Q_j and directly assign bu_j and ci_j to record i.
4. $level\_id_i = ba_j$ **for more than one** $ba_j \in BA$**;**. In this case, we obtain multiple mapping vectors $Q_j, \dots, Q_{j'}$. Therefore, we can assign multiple configuration items $ci_j, \dots, ci_{j'}$. As we are lacking a lower level identifier, we simply assign `CI unknown`. Furthermore, it was found that in cases with multiple business application matches, the respective business units, $bu_j, \dots, bu'_j$ are always identical, so $bu_j = \cdots = bu'_j$. We can therefore directly assign bu_j to record i.
5. $level\_id_i$ has **no matches** with either ci_j or ba_j; We, therefore, have no insight on the level of arrival, nor on the related business unit, for this arrival.

Cardinality configuration item - business application mapping

In order to inspect the mapping between configuration items and business applications, we define two cardinality functions: one for the number of configuration items associated with a business application, and one for the number of business applications associated with a configuration item.

First we consider service vector space G with service vectors Q_j.

Definition 4.1.5: Cardinality for number of business applications associated with a configuration item

Let $k \in CI$. Let $N_k^{CI \leftarrow BA} = \boldsymbol{Card}(j : ci_j = k)$. $N_k^{CI \leftarrow BA}$ denotes the cardinality of associated business applications for a given configuration item k in service space G.

Definition 4.1.6: Cardinality function per configuration item

For every $k \in CI := \{ci_1, \dots, ci_{n_{ci}}\}$, we define

$$\tilde{N}_1^{CI \leftarrow BA} := \max\{N_k^{CI \leftarrow BA} : k \in CI\},$$
$$\tilde{N}_k^{CI \leftarrow BA} := \max(\{N_k^{CI \leftarrow BA} : k \in CI\} \backslash \{\tilde{N}_1^{CI \leftarrow BA}, \dots, \tilde{N}_{k-1}^{CI \leftarrow BA}\}).$$

Similarly, we can define the cardinality of configuration items associated with a business application.

Definition 4.1.7: Cardinality for number of configuration items associated with a business application

Let $k \in BA$. Let $N_k^{BA \leftarrow CI} = \boldsymbol{Card}(j : ba_j = k)$. $N_k^{BA \leftarrow CI}$ denotes the cardinality of associated configuration items for a given business application k in service space G.

Definition 4.1.8: Cardinality function per business application

For every $k \in BA := \{ba_1, \ldots, ba_{n_{ba}}\}$, we define

$$\tilde{N}_1^{BA \leftarrow CI} := \max\{N_k^{BA \leftarrow CI} : k \in BA\},$$
$$\tilde{N}_k^{BA \leftarrow CI} := \max(\{N_k^{BA \leftarrow CI} : k \in BA\} \backslash \{\tilde{N}_1^{BA \leftarrow CI}, \ldots, \tilde{N}_{k-1}^{BA \leftarrow CI}\}).$$

The cardinality functions from Definitions 4.1.6 and 4.1.8 can be found in Figures 4.1 and 4.2, respectively. Interestingly, in Figure 4.1 it can be found a few segments of configuration item $k \in CI$ are linked to exactly the same number of business applications $N_k^{CI \leftarrow BA}$. For instance, we see three segments for $k \in [4, 45]$, $k \in [63, 110]$, and $k \in [924, 2178]$. In Section 6.3.3, we will observe that indices k with the same cardinality $N_k^{CI \leftarrow BA}$ not only correspond to the same cardinality of business applications but also to the exact same set of associated business applications $ba \in BA$.

Furthermore, in Figure 4.2 it can be found that the number of associated configuration items for business application index $k \in BA$ log-log linearly decays. Finally, by comparing Figure 4.1 and 4.2, it can be found that $N_k^{CI \leftarrow BA}$ (the number of business items associated with a configuration item) is in general higher than $N_k^{BA \leftarrow CI}$ (the number of configuration items associated with a business application) for arbitrary k.

Authors note: This figure is removed due to confidentiality.

Figure 4.1: Log-log plot for the function $k \to \tilde{N}_k^{CI \leftarrow BA}$, defining the number of business applications associated to configuration item k.

Authors note: This figure is removed due to confidentiality.

Figure 4.2: Log-log plot for the function $k \to \tilde{N}_k^{BA \leftarrow CI}$, defining the number of configuration items associated to business application k.

Arrivals per level

For each of the top three levels (business unit, business application, and configuration item), we can obtain the number of arrivals. The number of arrivals can be found in figure 4.3

Authors note: This figure is removed due to confidentiality.

Figure 4.3: Log-log plot for the number of events and incidents records per configuration item (left), business application (middle), and business unit (right).

It can be seen that even on the highest level, namely the business units, the total number of either incident or event arrivals can be zero. This implies that with certainty there will be business units where we can not relate event records to incident records, simply because there either are no event or incident records registered on this business unit.

4.1.3. A five-level hierarchy

In the course of our investigation into the hierarchical service architecture, we have examined three levels, namely the business unit, business application, and configuration item. However, it is essential to acknowledge that the service architecture of interest encompasses two more levels. The first of these additional levels is relatively straightforward and pertains to the arrival nature, which can be

categorized as either an event (E) or an incident (I), as discussed in Section 3.1. Although this level, represented by the feature vector $R_i \in F$, constitutes the lowest observable level, it lacks the desired granularity for our problem analysis. Associating temporal arrival components based solely on the arrival nature may not accurately capture the intricate dependencies between different configuration items. However, it is reasonable to anticipate that a specific cluster of arrivals, such as events indicating a server is low on memory, can be associated with another cluster of arrivals, like incidents indicating the same server is malfunctioning.

Consequently, to address this limitation and achieve a more meaningful analysis, the creation of an additional service level is required, involving semantically similar message clusters. By introducing this level, we establish a five-level hierarchical structure for the service architecture, as depicted in Figure 4.4.

The hierarchical service architecture is of utmost importance in the hierarchical Hawkes model and is extensively used in Chapter 6. The five levels from Figure 4.4 is denoted as

- Business unit $m \in BU := \{bu_1, \dots, bu_{n_{bu}}\}$,
- Business application $l \in BA := \{ba_1, \dots, ba_{n_{ba}}\}$,
- Configuration item $k \in CI := \{ci_1, \dots ci_{n_{ci}}\}$,
- Arrival nature $j \in S := \{E, I\}$,
- Message cluster $h \in CLU^m := \{clu_1^m, \dots, clu_{n_{clu}^m}^m\}$.

Remark 4.1.9. We denote the space of clusters CLU^m with a superscript m for the respective business unit $m \in BU$. This is because we perform message clustering *per business unit*. As a consequence, the number of clusters varies per business unit $m \in BU$.

In the following section, we thoroughly evaluate the various options available for constructing the message clusters, considering their potential impact on a hierarchical Hawkes model decomposition.

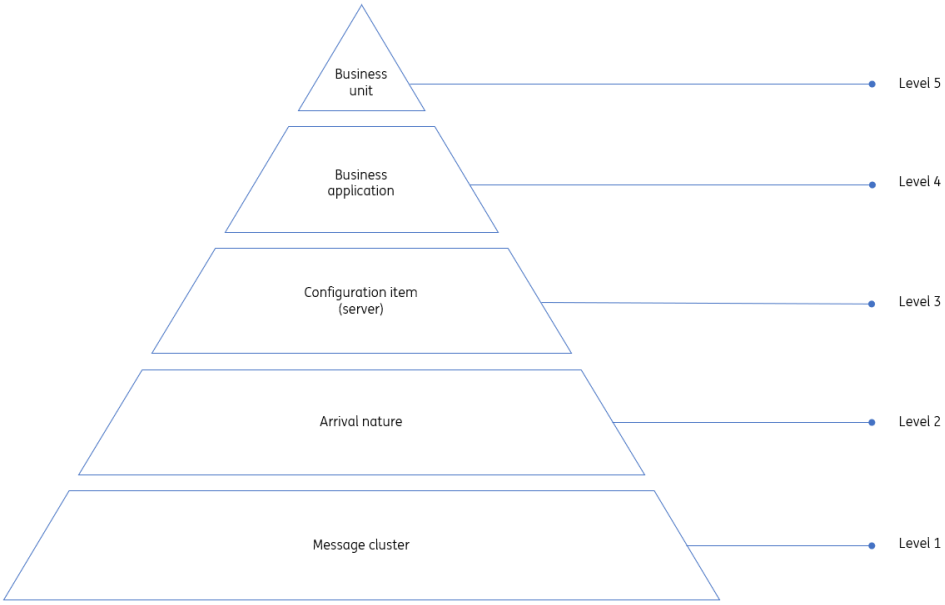

Figure 4.4: Five-level hierarchical service architecture

4.2. Bottom level message cluster

To perform message clustering, we first have to preprocess the messages at hand. This is a common and required step in natural language processing (NLP) [8]. Secondly, we describe the characteristics of parsing log messages. From here on we elaborate on message clustering. We describe message embedding methods, data dimensionality reduction using UMAP, and the HDBSCAN clustering algorithm.

Throughout the analysis, we attempt to find the "best" message clustering setting.Nonetheless, a good definition of the "best" message clustering remains hard to establish. Clearly, it must strike a delicate balance between an insufficient number of clusters and an excessive abundance of clusters. This particular instance mirrors the well-known bias-variance trade-off, which is unavoidable in high-dimensional and nonparametric statistics [21]. Another discussion of this trade-off related to the choice of the number of clusters can be found in [73, pages 14-15].

In our application, this trade-off appears in the following form. Choosing too many clusters, each encompassing only a small number of messages, will result in clusters with closely related semantic content. However, this approach might inadvertently segregate messages with comparable meanings into separate clusters. Given the small number of messages within each cluster, the estimator of the associations between message clusters would be less stable, and have larger errors. Consequently, the estimator of the parameters of interest would suffer from a high variance, as well as being very high-dimensional and harder to interpret.

Conversely, if we opt for a too small number of clusters, we are compelled to merge dissimilar message clusters, each possessing distinct semantics. As a result, the estimated associations become biased. To explain this phenomenon, we introduce a small example. Imagine that there are three true clusters C_1, C_2, C_3 of equal sizes (in terms of number of messages) with a certain association matrix (later in this thesis denoted by α) of size 3×3 such that $\alpha_{i,j}$ representing the influence of a message from a given cluster j on another cluster i. If we wrongly choose 2 clusters and merge C_1 and C_2 to become the larger cluster C_{12}, the influence of a given message of C_{12} on C_3 will be close to $(\alpha_{3,1} + \alpha_{3,2})/2$, i.e. the average of the influence of the two clusters C_1 and C_2 on C_3. Therefore a message from cluster C_1 will be identified in this situation as having an influence on C_3 of $(\alpha_{3,1} + \alpha_{3,2})/2$ (on average), which is biased compared to the true value of $\alpha_{3,1}$ in the general case where $\alpha_{3,1} \neq \alpha_{3,2}$.

More generally, merging together clusters that are (too) different will dilute the information, and loosing specificities: the estimates will be close to the average influence of the merged clusters. The more different these clusters are, the bigger the bias in the estimation will be (compared to the true value of association between the underlying true clusters).

To conclude, both a too high and a too low number of clusters will result in unsatisfactory results, but for different reasons: not enough information to estimate too small clusters precisely (high variance) or unreliable information when merging too dissimilar clusters (since a given message from a particular cluster that was merged may come from a cluster that has a very different association than the average association of the merged clusters). Such a bias-variance tradeoff will directly affect the results of our hierarchical Hawkes model from Chapter 6, and will necessitate a careful choice of the number of clusters.

This brings us to finalize this chapter by answering Research Question 2.

Research Question 2

How can IT messages with similar semantics be grouped together?

Preprocessing

In order to group semantically similar messages together, we have to preprocess the messages at hand [8, 46, 57]. First, we convert all string characters to lowercase and remove dates, excess spaces and stop words. Moreover, stop words are words that do not add much meaning to a sentence, such as *the, is, at, which,* and *on*. They can safely be ignored without sacrificing the meaning of the sentence. Secondly, we normalize verbs to their base form. This step is called lemmatization. Lemmatization is employed to transform words into their base or root forms, allowing for text normalization and reducing lexical variations. This process aids in improving the accuracy of language analysis.

Finally, it is essential to highlight that "natural" language messages differ in structure compared to IT messages obtained from the IT monitoring data stream. Specifically, these messages contain ING-specific jargon, such as server and process names. To make them more akin to natural language messages,

system engineers at ING established a mapping with over sixty regular expression patterns. For instance, all regular expressions known to represent server names were replaced by the string "<server>".

4.2.1. Log parsing

As events are automatically generated, their messages are constructed by means of a template. This template is then filled in with variables. Because of this structured approach, log parsing techniques can be employed to retrieve the original template and extract the variables. Once the template for each message is extracted, arrivals with identical templates can be grouped together. Notably, the preprocessing step of replacing ING-specific terminology already reduces the number of variables that need to be extracted.

Drain3

One of the commonly used log parsing techniques is Drain. Drain follows a three-step heuristic-based approach for log parsing. In the initial step, messages are preprocessed using user-provided regular expressions based on domain knowledge. The subsequent step involves constructing a *parse tree* with multiple heuristics, allowing log parsing based on the number of characters contained in each message. In the final step, logs' tokens are compared using a similarity metric to extract log templates and update the parse tree. Drain achieves high accuracy, especially in scenarios with limited log diversity. However, its reliance on hard-coded rules requires vigilant maintenance when adapting to structural changes in log data.

It should be noticed that parsing does not group semantically similar messages into the same cluster if they don't share the same structure. For example, consider the following messages:

```
Server down:  ABC
  Server XYZ down
```

After parsing, the respective templates would become `Server down:  <*>` and `Server <*> down.` Without any additional steps, these semantically similar messages would not end up in the same cluster. Hence, it is worthwhile to explore more advanced clustering methods to address this issue.

4.2.2. Clustering: embedding

In order to cluster messages, we first need to transform the message of interest into a numeric vector. This process is known as message embedding. Moreover, there exists a large variety of embeddings. In this thesis, we explore two of them: the traditional TF-IDF model and the neural network model known as Doc2vec.

TF-IDF

Once the messages are preprocessed and ING-specific jargon is substituted by placeholders, the messages are vectorized. For this purpose, the term frequency-inverse document frequency (TF-IDF) model can be used. This word-embedding model has been a longstanding conventional option for generating word embeddings and document embeddings. As indicated by a survey conducted in 2015, 83% of text-based recommender systems in digital libraries use TF-IDF [6].

TF-IDF is calculated based on two key statistical metrics: the term frequency (TF) and the inverse document frequency (IDF). The term frequency $TF(t, m_i)$ measures the relative frequency of the word t within the message $m_i \in MSG$. It can be defined as follows:

$$TF(t, m_i) = \frac{f_{t,m_i}}{\sum_{t' \in m_i} f_{t',m_i}}.$$

Here, f_{t,m_i} represents the frequency of word t occurring in message m_i, while the denominator $\sum_{t' \in m_i} f_{t',m_i}$ represents the total number of words in message m_i.

The inverse document frequency (IDF) measures how informative a word is based on whether it's common or rare across the entire collection of previously gathered messages. This metric is defined as:

$$IDF(t, MSG) = \frac{N}{1 + |\{m_i \in MSG : t \in m_i\}|}.$$

Here, N represents the total number of gathered messages, and $|\{m_i \in MSG : t \in m_i\}|$ represents the number of messages where the word t appeared from the collection of gathered messages MSG. The collection of gathered messages MSG is denoted the *corpus*. The addition of 1 in the denominator avoids division by zero in cases where a word did not occur in any of the gathered messages. The TF-IDF score for word t in message m_i from the corpus MSG is calculated as:

$$TFIDF(t, m_i, MSG) = TF(t, m_i) \cdot IDF(t, MSG).$$

This computation results in a high TF-IDF score when a term appears frequently in a specific message but only rarely across the entire collection of words in the corpus.
Each message m_i is embedded as an $\boldsymbol{Card}(MSG)$-dimensional vector. Each word t to ever appear in the corpus MSG but did not occur in message m_i is assigned a value zero. Conversely, words t present in m_i are assigned a value of $TFIDF(t, m_i, MSG)$.

We create two separate corpora, one for all words found in event messages $MSG_I := \{m_i \in MSG : s_i = I\}$ and one for all words found in incident messages $MSG_E := \{m_i \in MSG : s_i = E\}$. Subsequently, each message can be transformed into a vector using the corresponding corpus linked to its arrival nature. As a result, distinct IDF values can be assigned for each of the corpora, namely $IDF(t, MSG_I)$ and $IDF(t, MSG_E)$.

Doc2vec

The Doc2vec model uses a neural network framework to learn word associations from a corpus of messages. We use a continuous bag-of-words model. The bag-of-words model predicts the current word from the window of surrounding context words. The order of the words does not influence prediction. This assumption is very useful for the IT monitoring messages, as it helps overcome changes in the message template as seen in Section 4.2.1.

It should be noted that the Doc2vec model is pre-trained on regular text messages, such as Wikipedia pages. These types of text messages contain far more words per message as compared to the IT monitoring messages under consideration in this thesis. Moreover, the Doc2vec embedding is known to perform poorly for very short text messages [20]. Here 'very short' is referred to as less than 20 words, which might often be the case for event messages in particular. Furthermore, the regular text messages on which the Doc2vec model is trained abide by a more natural language structure. Conversely, the IT monitoring messages on which inference is performed often look more like computer-generated code. Therefore we could observe a distribution shift. A distribution shift arises when the distribution of the samples of data on which the model was fitted is different from the distribution of the data the model runs inference on [32].

4.2.3. Clustering: HDBSCAN

In order to cluster similar messages together, we use the HDBSCAN clustering algorithm. This density-based clustering algorithm uses a set of hyperparameters to cluster semantically similar messages together. Clustering is an essential step, as it helps to provide granularity on top of only predicting if all events on a level of service are associated with all incidents on the same service. In order to perform the HDBSCAN clustering algorithm, a number of hyperparameters can be set.

- `min_cluster_size`: The minimum size of clusters; single linkage splits that contain fewer points than this will be considered points "falling out" of a cluster rather than a cluster splitting into two new clusters.
- `cluster_selection_epsilon`: A distance threshold. Clusters below this value are merged.
- `cluster_selection_method`: The method used to select clusters from the condensed tree. The standard approach for HDBSCAN is to use an Excess of Mass algorithm to find the most persistent

clusters. Alternatively, we can instead select the clusters at the leaves of the tree – this provides the most fine-grained and homogeneous clusters.
- `metric`: The metric to use when calculating the distance between instances in a feature array.

The different hyperparameter settings trivially lead to different message clusters.

4.2.4. Clustering: dimensionality reduction using UMAP

Density-based clustering relies on having enough data to separate dense areas. In higher dimensional spaces this becomes more difficult and hence requires more data. From the embeddings in Section 4.2.2, we clearly saw messages being converted into high-dimensional vectors. The Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) technique was therefore proposed by Asyaky et al. [2]. UMAP has several hyperparameters which can have a notable impact on the resulting embedding. We tune the following two

- `n_neighbours:` Controls how UMAP balances local versus global structure in the data.
- `n_components:` Allows to determine the dimensionality of the reduced dimension space we embed the data into.

The authors of UMAP mention two more major hyperparameters, namely, `metric` and `min_dist`. In the interest of time, we did not tune these two hyperparameters and relied on their default values.

4.2.5. Clustering: validation

Finally, we need to decide how to select the optimal hyperparameter setting. As we do not have a true cluster label (in other words, we are in an unsupervised setting), validating the clustering result can be challenging. In general, clustering score functions are rather similar: they measure how similar an object is to its own cluster as compared to other clusters. One of the most commonly used clustering techniques that rely on this principle is the Silhouette score [68]. However, the Silhouette score is not suited for density-based clustering methods such as HDBSCAN for two reasons. First of all, density-based clustering methods find arbitrarily shaped, non-convex clusters. The Silhouette score is not able to account for this. Secondly, the Silhouette score can not incorporate outliers into their scoring.

To overcome the lack of appropriate measures for the validation of density- based clusters, Moulavi et al. propose a measure called the Density-Based Clustering Validation index (DBCV) [56]. It assigns a weighted sum validity indices, computed for each cluster. Here each validity index calculates the cluster compactness.

Although DBCV works well for several applications, for the problem of message clustering it favors having a smaller number of clusters and leaves too many samples as noise. This is problematic because later on in Chapter 6, we will see each cluster needs to have a substantial number of points (i.e. temporal arrivals) in order to associate its arrival with another cluster. Therefore, we not only rely on the DBCV score but also consider the probability of each message being assigned to a specific cluster. In particular, we aim to minimize the percentage of data with less than 5% cluster label confidence. We call this score the *uncertainty percentage*.

4.3. Real-world data

To illustrate the theory mentioned above, we applied the HDBSCAN clustering algorithm to one business unit. For this purpose, we choose one specific business unit. We chose this specific business unit as it contains both event and incident arrivals, records a manageable amount of arrivals, and is verified by system engineers from ING to be a business unit that is representative of the service architecture at ING. Before elaborating on the results from HDBSCAN, we will first show the results of Drain.

4.3.1. Drain

After log parsing using Drain, we obtained 918 templates for event messages and 642 templates for incident messages. These results can be found in Table 4.1. The rather high number of templates for both event and incident messages can be explained by the fact that Drain is not able to group messages with

similar semantics together. This is to be expected as Drain is a log-parsing technique and is therefore only able to extract a template.

As we want to be able to cluster messages with similar semantics together, we now inspect the combination of message embeddings in combination with the HDBSCAN clustering algorithm. The main motivation will be to reduce the number of obtained groups.

4.3.2. HDBSCAN

We ran a grid search for both arrival types $s \in S$ for both TF-IDF and Doc2vec embedding. This results in four pairs of parameter settings. We perform hyperparameter tuning for four hyperparameters regarding HDBSCAN (See Section 4.2.3) and two hyperparameters regarding UMAP (see Section 4.2.4). The grid can be found in Appendix C.1.

The grid search calculates both the DBCV score and our self-defined uncertainty percentage for each of the hyperparameter settings. We aim for a high DBCV score and a low uncertainty percentage. Conversely, the objective regarding the number of clusters and outliers is less straightforward, as we have already discussed in relation to the bias-variance trade-off in Section 4.2.

Embedding: TF-IDF & Doc2vec

In Figure 4.5, we compare the results for the grid search for the TF-IDF and Doc2vec embeddings. From the top figures, it is clear that the TF-IDF embedding results in superior results in terms of DBCV score and the uncertainty percentage as compared to a Doc2vec embedding for both events and incidents. From the bottom figures, this results from fewer outliers and a higher number of clusters.

As stated in Section 4.2.2, there might be a distribution shift between the regular text messages on which the Doc2vec model is trained and the IT monitoring messages on which we run inference. Therefore, the poor performance of the Doc2vec embedding model should not come as a surprise.

HDBSCAN: metric

The top figures in Figure 4.6 reveal that for incident messages, using the Manhattan distance for HDBSCAN clustering results in a lower DBCV score and a slightly higher uncertainty percentage. However, this doesn't seem to be the case for event messages.

The bottom figures show that the Manhattan distance leads to a higher number of clusters, which is undesirable for the computation of the excitation matrix discussed later in Chapter 6.

HDBSCAN: cluster selection epsilon

From the top figures from Figure 4.7, it can be found that a higher clustering selection epsilon does not lead to a change in the DBCV score or uncertainty score. From the bottom figures, we do obtain that a higher clustering selection epsilon results in fewer clusters. This is in line with the description of `cluster_selection_epsilon` in Section 4.2.3.

HDBSCAN: cluster selection method

Figure 4.9 shows that there is no substantial difference in performance between the Excess of Mass (eom) and Leaf clustering algorithms. As discussed in Section 4.2.3, the Excess of Mass algorithm tends to result in small homogeneous clusters, which would be preferred in the context of event and incident message clustering. Therefore, it is interesting to observe that the results do not seem to differ seem to differ.

HDBSCAN: minimum cluster size

The last hyperparameter we consider from HDBSCAN is the minimum cluster size, which, as noted by the authors of HDBSCAN, is the most sensitive parameter to tune [53]. Figure 4.9 illustrates that a smaller minimum cluster size leads to a higher DBCV count and lower cost, which is desired. However, setting the minimum cluster size too small (e.g., 2 to 3) also results in nearly no outliers and a high number of clusters, which can pose challenges when estimating the excitation matrix.

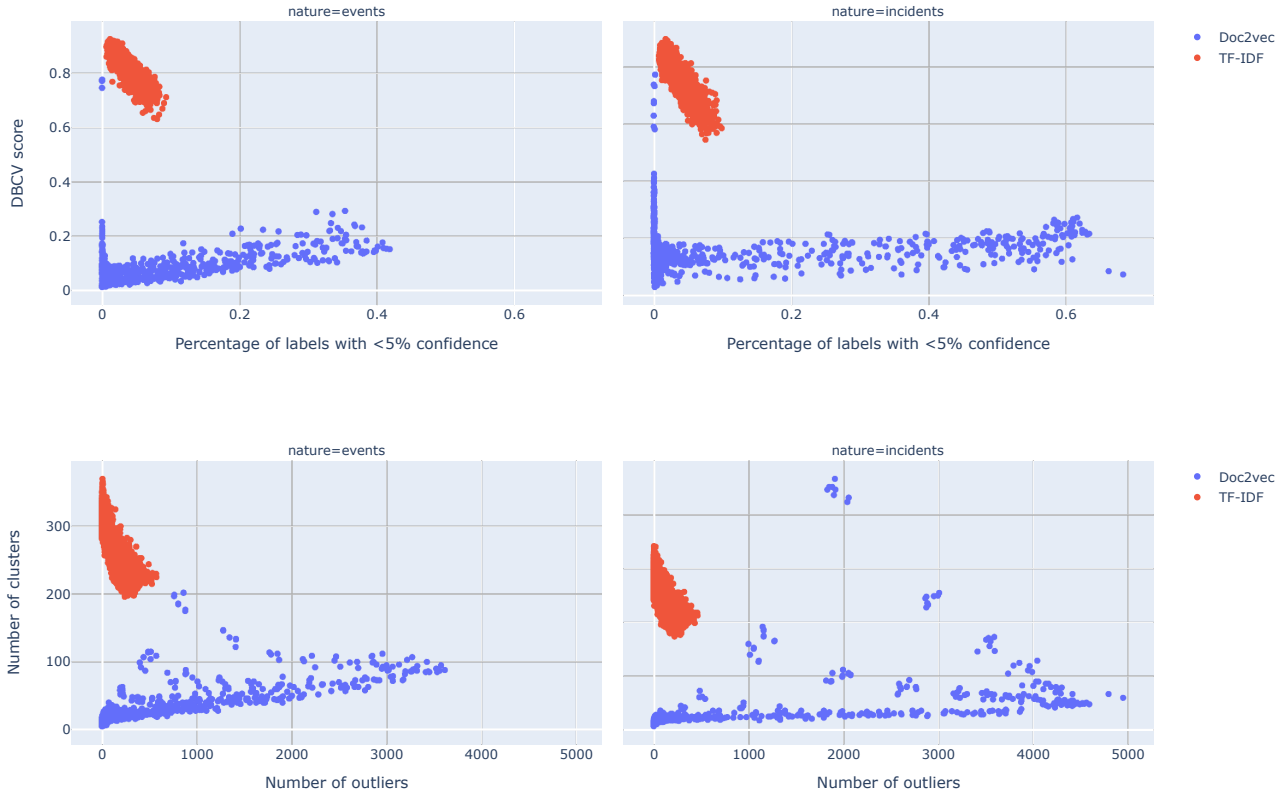

Figure 4.5: Grid search results for two different choices for the word `embeddings` (TF-IDF en Doc2vec). *Top figures:* DBCV score against the percentage of labels with less than 5% confidence for the assigned cluster for events (left) and incidents (right). *Bottom figures:* Number of clusters against the number of outliers for events (left) and incidents (right).

UMAP: n-neighbors and n-components

Finally, we evaluate the two hyperparameters related to UMAP, namely `n_neighbors` and `n_components`. From Figure 4.10, we see the reduced dimension (i.e. `n_components`) does not influence the results. From Figure 4.11, on the other hand, we obtain rather intriguing results. It can be observed that, for events, 13 components appear to yield superior results in terms of the DBCV score. In contrast, for incidents, this optimal value seems to be 14 components. Interestingly, this is the only hyperparameter where no linear trend can be discerned, and the behavior differs between events and incidents as well.

4.3.3. Results

Authors note: This paragraph is removed due to confidentiality.

Table 4.1: Number of arrivals for different stages of processing for both event and incident arrivals.

Authors note: This table is removed due to confidentiality.

Table 4.2: Scores for chosen grid search for both event and incident arrivals.

Authors note: This table is removed due to confidentiality.

Classified outliers

Authors note: This paragraph is removed due to confidentiality.

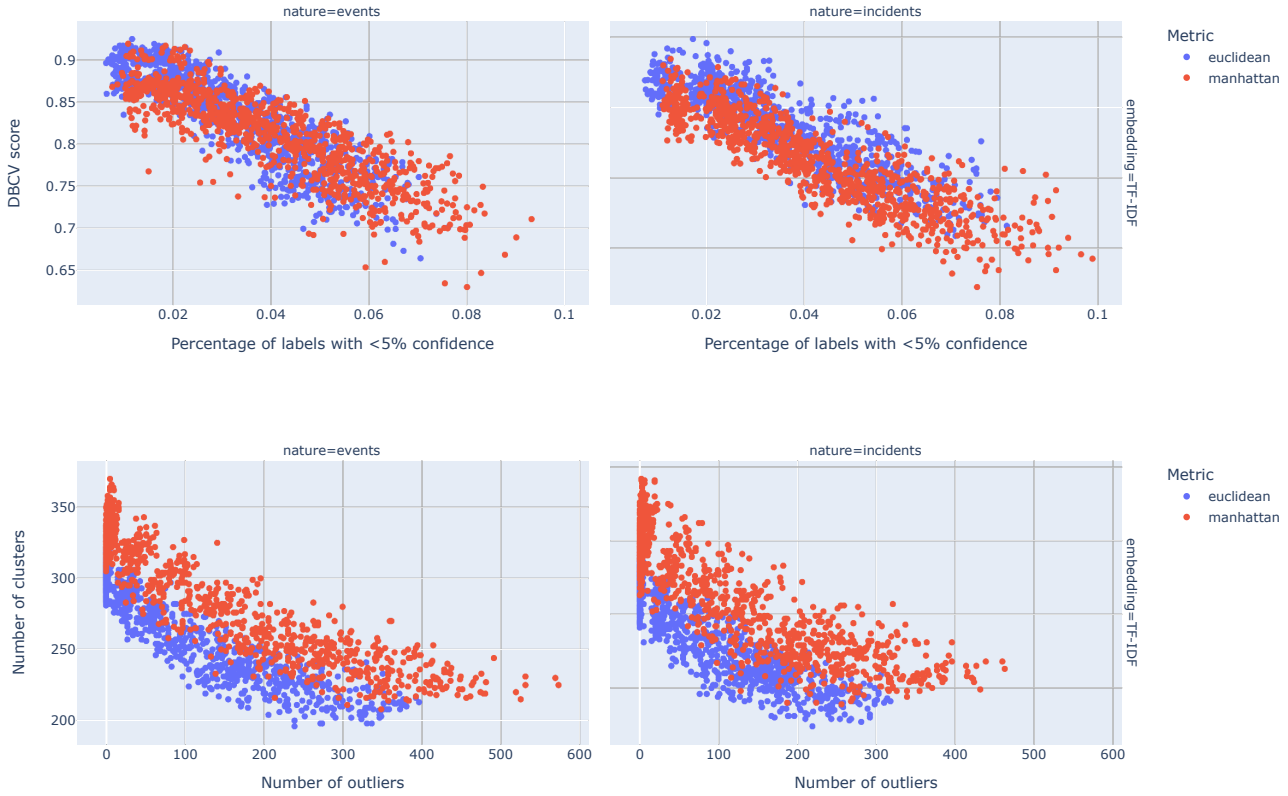

Figure 4.6: Grid search results for two different choices for the `metric` (TF-IDF en Doc2vec). *Top figures:* DBCV score against the percentage of labels with less than 5% confidence for the assigned cluster for events (left) and incidents (right). *Bottom figures:* Number of clusters against the number of outliers for events (left) and incidents (right).

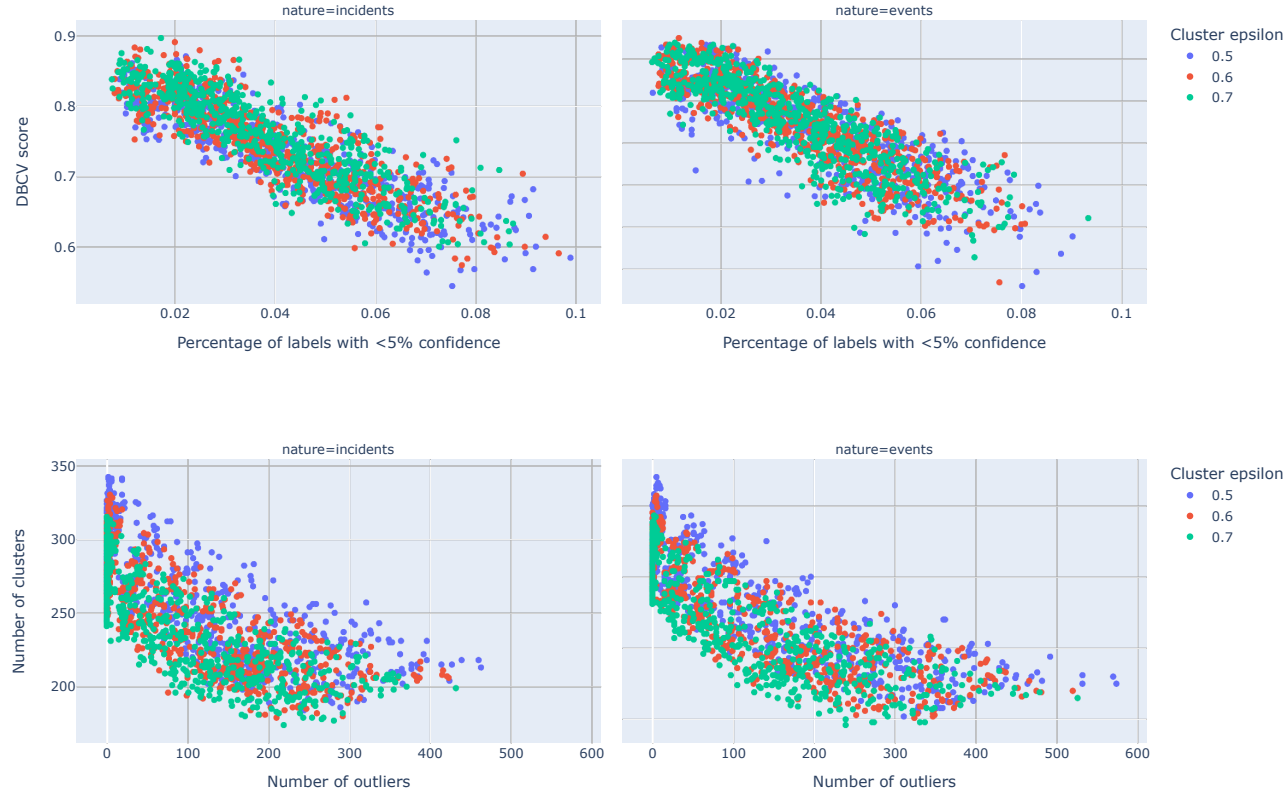

Figure 4.7: Grid search results for three different choices for the `cluster_selection_epsilon` (0.5, 0.6, 0.7). *Top figures:* DBCV score against the percentage of labels with less than 5% confidence for the assigned cluster for events (left) and incidents (right). *Bottom figures:* Number of clusters against the number of outliers for events (left) and incidents (right).

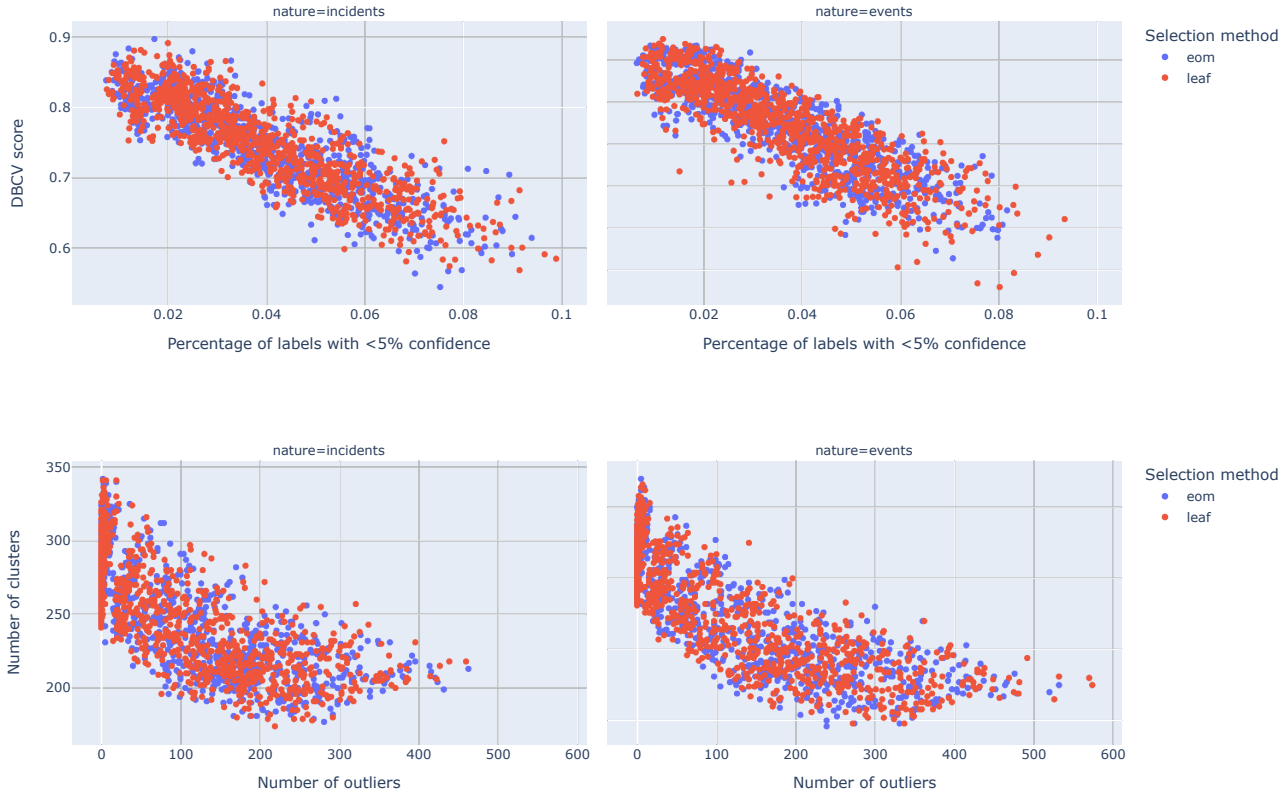

Figure 4.8: Grid search results for two different choices for the `cluster_selection_method` (EoM and Leaf). *Top figures:* DBCV score against the percentage of labels with less than 5% confidence for the assigned cluster for events (left) and incidents (right). *Bottom figures:* Number of clusters against the number of outliers for events (left) and incidents (right).

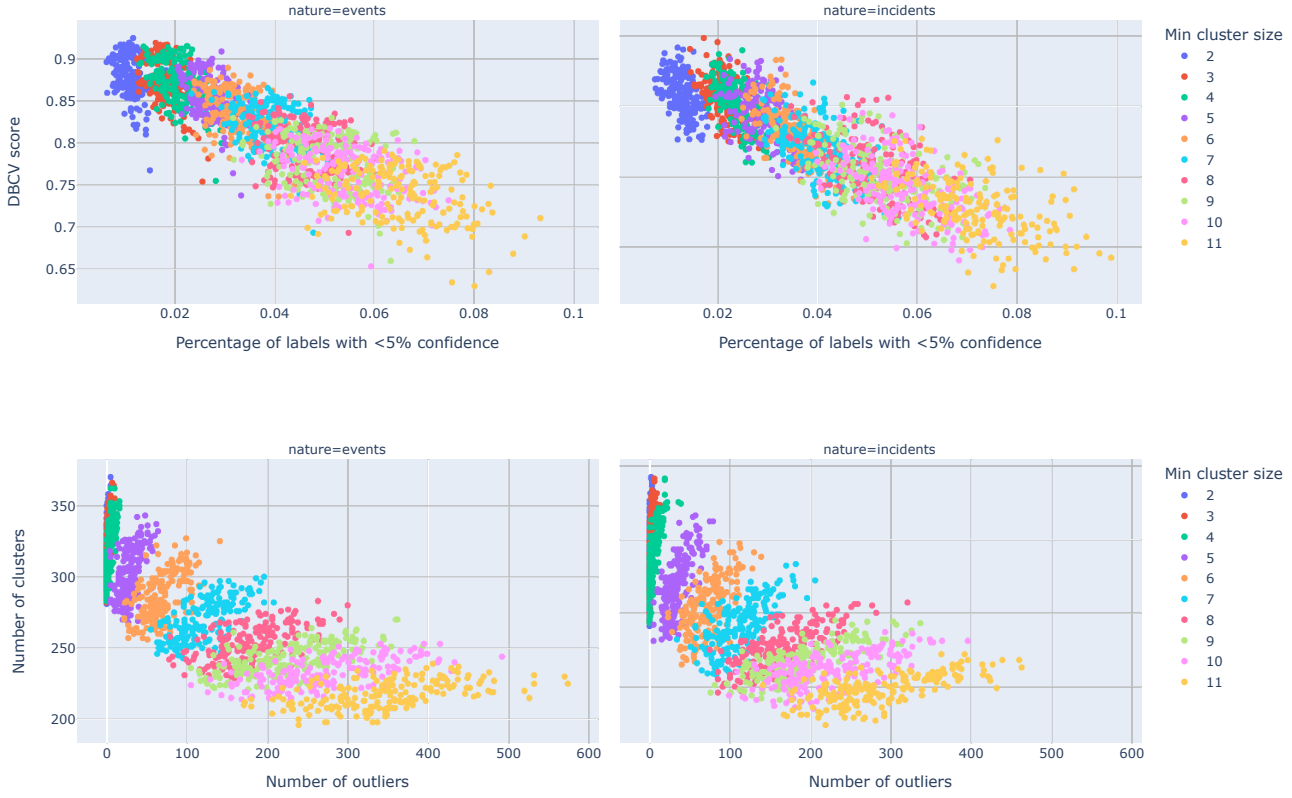

Figure 4.9: Grid search results for ten different choices for the `min_cluster_size` (2 to 10). *Top figures:* DBCV score against the percentage of labels with less than 5% confidence for the assigned cluster for events (left) and incidents (right). *Bottom figures:* Number of clusters against the number of outliers for events (left) and incidents (right).

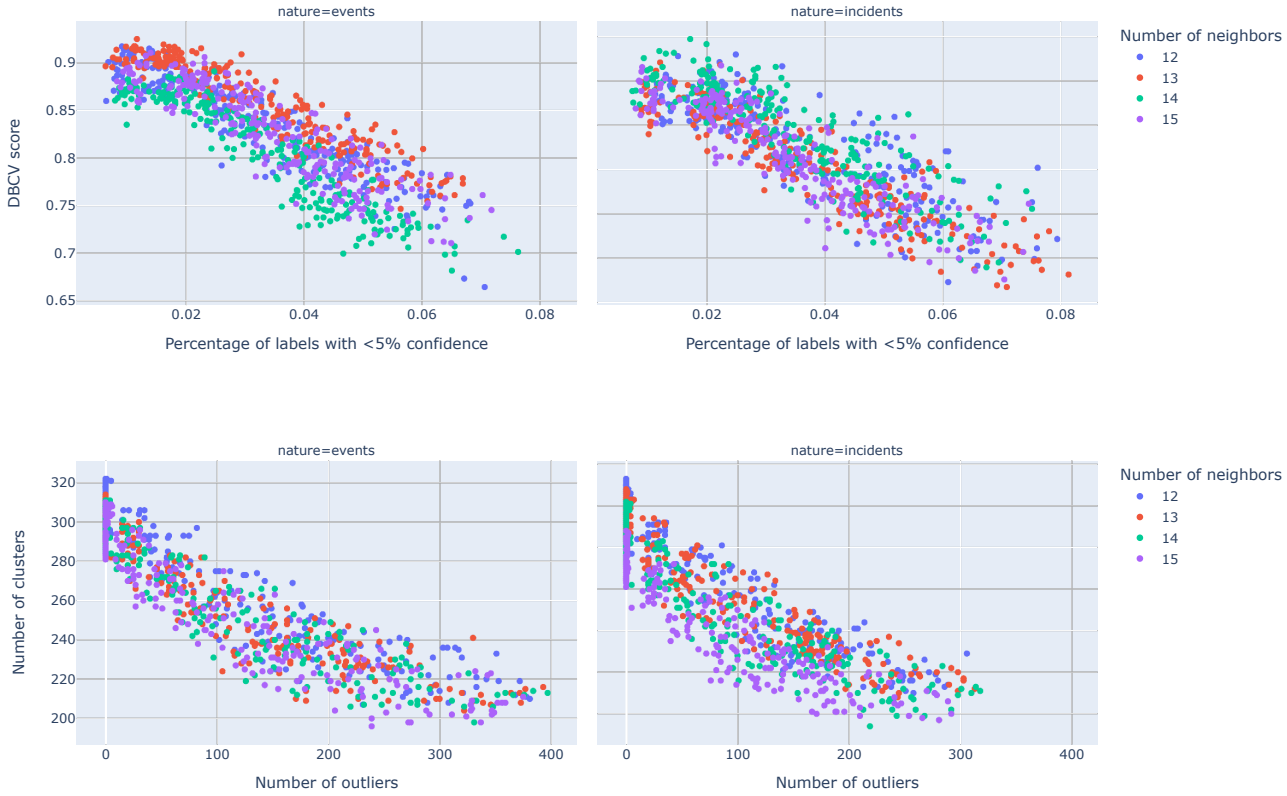

Figure 4.10: Grid search results for four different choices for the `n_neighbors` (11, 12, 13 and 14). *Top figures:* DBCV score against the percentage of labels with less than 5% confidence for the assigned cluster for events (left) and incidents (right). *Bottom figures:* Number of clusters against the number of outliers for events (left) and incidents (right).

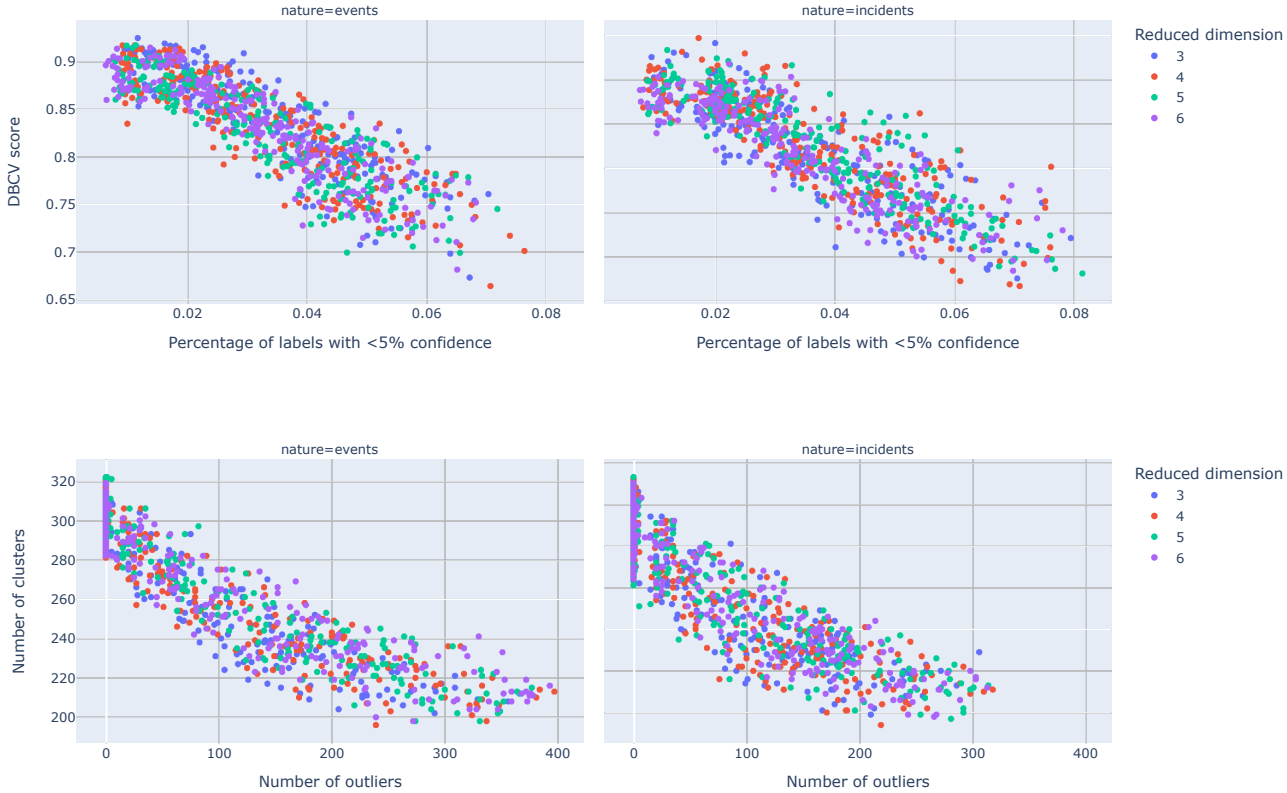

Figure 4.11: Grid search results for four different choices for the `n_components` (3, 4, 5, 6). *Top figures:* DBCV score against the percentage of labels with less than 5% confidence for the assigned cluster for events (left) and incidents (right). *Bottom figures:* Number of clusters against the number of outliers for events (left) and incidents (right).

Part III

Hawkes process analysis

5

Hawkes Processes

This chapter examines the theory behind Hawkes processes, which will play a central role in building the hierarchical Hawkes model in Chapter 6. For this purpose, event and incident arrivals are modeled as a marked point process. Subsequently, we can use the key characteristic of Hawkes process that past arrivals influence the intensity rate of future arrivals.

This chapter is divided into four sections. In Section 5.1 we recap the fundamental concepts of stochastic processes. In particular, we recap point and counting processes and examine the inadequacies of (homogeneous and non-homogeneous) Poisson processes. In Section 5.2, we introduce the one-dimensional Hawkes process. Additionally, we introduce different memory kernels, which govern the excitation behavior over time. Furthermore, we prove the consistency of the maximum likelihood and least squares estimator. Moving on, in Section 5.3 we generalize the one-dimensional Hawkes process to a marked Hawkes process. In addition, we introduce the concept of Granger-causality. Finally, in Section 5.4, we see how we can perform inference using the Python package *Tick*.

5.1. Stochastic processes

First, we look at some of the tools required to work with stochastic processes. We formally define point and counting processes. We assume the reader is familiar with measure-theoretic notation.

5.1.1. Point processes

As an essential start, we start by defining the fundamental concept of point processes. A point process is defined on some underlying mathematical space. Point processes can be studied in general settings, such as locally compact second-countable Hausdorff spaces [17]. In this thesis, we however restrict ourselves to the real line $\mathbb{R}$.

To define general point processes, we start with a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Furthermore, let $\mathcal{B}(\mathbb{R})$ be the Borel σ-algebra on $\mathbb{R}$.

Definition 5.1.1: Counting measure

Let $(\mathcal{X}, \mathcal{A})$ be a measurable space. Let $A \in \mathcal{A}$. We define the counting measure of A, denoted by ν_A to be the measure on $(\mathcal{X}, \mathcal{A})$ such that, for every $B \in \mathcal{A}$,

$$\nu_A(B) = \boldsymbol{Card}(A \cap B). \tag{5.1}$$

We define $\mathbb{M}(\mathcal{X})$ to be the space of counting measures on $\mathcal{X}$. For a counting measure ν_A, its total number of points is $N(\nu_A) := \boldsymbol{Card}(A)$. Unless otherwise specified, the considered space will be $\mathcal{X} = \mathbb{R}$. In this sense, a counting measure ν_A on $\mathbb{R}$ is characterized by the measurable set $A \in \mathcal{B}(\mathbb{R})$.

Definition 5.1.2: Point processes

A point process $\mathbb{T}$ on $\mathcal{X}$ is defined as a mapping from the sample space Ω to the space of counting measures $\mathbb{M}(\mathcal{X})$, meaning that each realization $\omega \in \Omega$ of a point process is a counting measure $\mathbb{T}(\omega) \in \mathbb{M}(\mathcal{X})$.

Remark 5.1.3. In this expression, the *point process* is denoted by $\mathbb{T}$, while $N(\mathbb{T})$ denotes the *cardinality* $\boldsymbol{Card}(A \cup \mathbb{T})$. $\mathbb{T}$ is a random counting measure and $N(\mathbb{T})$ is a random value in $\mathbb{N} \cup \{+\infty\}$.

Remark 5.1.4. We will denote individual arrivals $\mathbb{T}(\omega)$ as T_i, where i denotes the i^{th} arrival. We can write $\mathbb{T}$ with $N(\mathbb{T}) = n \in \mathbb{N} \cup \{+\infty\}$ as $\mathbb{T} = \{T_1, T_2, \dots, \}$

5.1.2. Counting processes

The last remark leads to a different representation, namely a counting process.

Definition 5.1.5: Counting processes [43]

A counting process is a stochastic process $N(\cdot) : \mathbb{R}^+ \to \mathbb{N}$ that satisfies $N(0) = 0$ and is an increasing right-continuous step function with increments of size 1.

Whereas a point process is a subset $\mathbb{T} \subset \mathbb{R}_+$ at which certain arrivals occur, a counting process represents the total number of arrivals that have occurred at a certain time. The relation between *counting processes* and *point processes* is given as follow:

Definition 5.1.6: Counting processes II [43]

Let $t \in \mathbb{R}^+$ with $[0, t] \in \mathcal{B}(\mathbb{R})$. With slight abuse of notation, we define the total number of points in $[0, t]$ as

$$N(t) = \sum_{T_i \in \mathbb{T}} \mathbb{I}_{[0,t]}(T_i). \tag{5.2}$$

Remark 5.1.7. We can define for $t_1, t_2 \in \mathbb{R}^+ : t_1 < t_2$, $N(t_1, t_2) = N(t_2) - N(t_1)$. Here $N(t_1, t_2)$ denotes the number of arrivals that occur in the interval $(t_1, t_2]$.

An illustration of the relation between a point process and a counting process can be found in Figure 5.1.

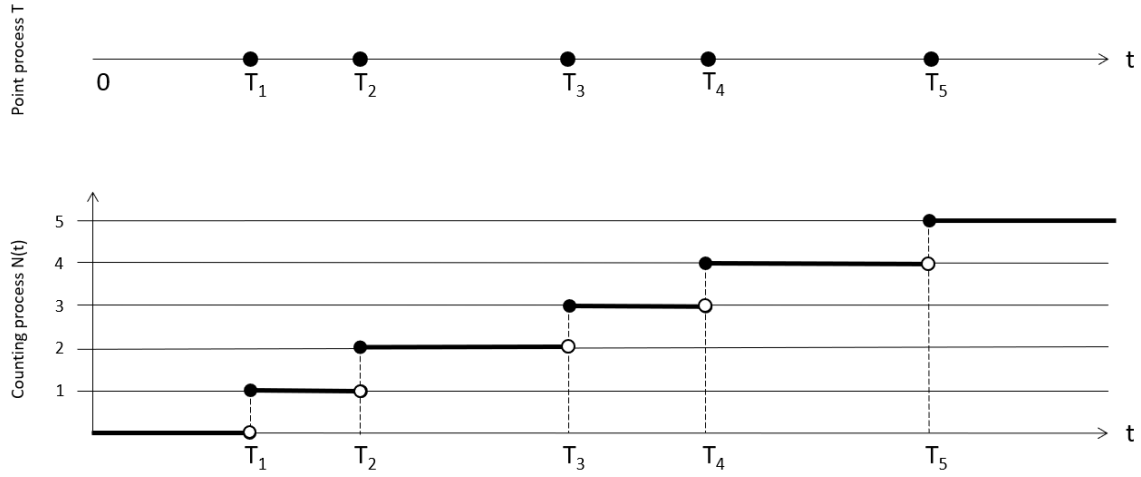

Figure 5.1: Illustration of point process $\mathbb{T}$ (top) and identical counting process $N(\cdot)$ (bottom) .

5.1.3. Homogeneous Poisson processes

As a first realization of point processes, we will describe homogeneous Poisson point processes, denoted as Poisson processes [43, 74]. Poisson processes are one of the most widely-used counting processes.

Definition 5.1.8: Homogeneous Poisson point process

A counting process $N(\cdot)$ is a homogeneous Poisson process with intensity $\lambda_0 > 0$ if

1. $\forall t_i, t_j \in \mathbb{R}_+ : t_i < t_j, N(t_i, t_j) \sim Pois(\lambda_0(t_j - t_i))$,
2. For any n disjoint intervals $(t_0, t_1], (t_1, t_2], \ldots, (t_{n-1}, t_n]$, the random variables $N(t_0, t_1), N(t_1, t_2), \ldots, N(t_{n-1}, t_n)$ are independent.

Here $Pois(\cdot)$ denotes the Poisson distribution, so $\mathbb{P}[N(t) = k] = \frac{(\lambda_0 t)^k \exp(-\lambda_0 t)}{k!}$. Special attention should be paid to the λ_0 parameter. If λ_0 is replaced by a time-varying function $\lambda(t)$, the obtained process becomes a non-homogeneous Poisson process, which we will see in the next section.

Remark 5.1.9. The intensity λ_0 is sometimes referred to as the *rate* in related literature.

As $\forall h \in \mathbb{R} : N(t_i, t_j) \overset{law}{=} N(t_i + h, t_j + h) \overset{law}{=} Pois(\lambda(t_j - t_i))$, the interarrival times exhibit two key characteristics: independence and stationarity. These properties imply that the interarrival times adhere to the memoryless property.

5.1.4. Non-homogeneous Poisson processes

Homogeneous Poisson processes are characterized by a constant intensity λ_0, independent of time. However, as we saw earlier in the case of incident arrivals, the intensity can be higher during business hours and lower during weekends and non-business hours. It is therefore appropriate to model the intensity as a function of time. We can therefore refer to an *intensity function*, $\lambda(t)$.

Definition 5.1.10: Non-homogeneous Poisson point processeses

A counting process $N(\cdot)$ is a non-homogeneous Poisson process with intensity function $\lambda(t) > 0$ if

1. $\forall t_i, t_j \in \mathbb{R}^+ : t_i < t_j \; N(t_i, t_j) \sim Pois(\int_{t_i}^{t_j} \lambda(s) ds)$,
2. For any n disjoint intervals $(t_0, t_1], (t_1, t_2], \ldots, (t_{n-1}, t_n]$, the random variables $N(t_0, t_1), N(t_1, t_2), \ldots, N(t_{n-1}, t_n)$ are independent.

As the law of $N(t_i, t_j)$ may be different from the law of $N(t_i + h, t_j + h)$ for $h \in \mathbb{R}_+$, the non-homogeneous Poisson process generally is not stationary.

5.1.5. Conditional intensity function

In Section 5.1.1, we defined a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. We can define a filtration $(\mathcal{H}_t)_{t \geq 0}$ with $\mathcal{H}_t \subseteq \mathcal{F}$. This results in a filtered probability space $(\Omega, \mathcal{F}, \mathcal{H}_t, \mathbb{P})$, where we will denote $\mathcal{H}_t$ the history up to (**so not including**) time t.

Remark 5.1.11. We will denote the last observed arrival time as T_n. We denote $\mathcal{H}_{T_n}$ the history up until (**so including**) the last observed arrival T_n.

Definition 5.1.12: Conditional cumulative distribution function

The conditional cumulative distribution function (CDF) of the next arrival time T_{n+1} given history $\mathcal{H}_{T_n}$ is defined as

$$F_{T_{n+1}}(t|\mathcal{H}_{T_n}) := \int_{T_n}^{t} \left(\lim_{\Delta s \searrow 0} \frac{\mathbb{P}(T_{n+1} \in [s, s + \Delta s] | \mathcal{H}_{T_n})}{\Delta s} \right) ds = \int_{T_n}^{t} f(s \mid \mathcal{H}_{T_n}) ds. \tag{5.3}$$

The conditional probability density function $f(s|\mathcal{H}_{T_n})$ is therefore defined as

$$f(s|\mathcal{H}_{T_n}) = \lim_{\Delta s \searrow 0} \frac{\mathbb{P}(T_{n+1} \in [s, s+\Delta s]|\mathcal{H}_{T_n})}{\Delta s}. \tag{5.4}$$

For the realizations of a point process, we can define the joint probability density function (PDF).

Definition 5.1.13: Joint probability density function

The joint probability density function of the first $n \in \mathbb{N}$ arrival times $(T_1, \dots, T_n)$ is defined as

$$f(T_1, \dots, T_n) := f(T_1) \cdot f(T_2|T_1) \cdot f(T_3|T_2, T_1) \cdot \dots \cdot f(T_n|T_{n-1}, \dots, T_1) = \prod_{i=1}^{n} f(T_i|\mathcal{H}_{T_{i-1}}). \tag{5.5}$$

Remark 5.1.14. Often in literature, the history $\mathcal{H}_t$ is not mentioned explicitly, and $\lambda(t|\mathcal{H}_t)$ is abbreviated to $\lambda^*(t)$. We will not follow this convention as it might cause confusion up to which point in time we are conditioning.

To align the probability density function with the framework of intensity functions, we will define the *conditional intensity function*. The first definition specifies the mean number of arrivals in an infinitesimal interval conditioned on the history $\mathcal{H}_t$.

Definition 5.1.15: Conditional intensity function

Consider a counting process $N(\cdot)$ with associated history $\mathcal{H}_t$. If a non-negative function $\lambda(t|\mathcal{H}_t)$ exist such that

$$\lambda(t|\mathcal{H}_t) = \lim_{\Delta t \searrow 0} \frac{\mathbb{E}[N(t+\Delta t) - N(t) \mid \mathcal{H}_t]}{\Delta t}, \tag{5.6}$$

then it is called the conditional intensity function of $N(\cdot)$.

The function $\lambda(t|\mathcal{H}_t)$ is $\mathcal{H}_t$-measurable, hence it only relies on information of $N(\cdot)$ in the past. We can now relate the conditional intensity function to the probability density function and the cumulative density function. This will be used later on when calculating the conditional intensity function.

Theorem 5.1.16: Conditional intensity function II

Consider PDF $f(t|\mathcal{H}_{T_n})$ and CDF $F(t|\mathcal{H}_{T_n})$ for $t > T_n$. Let $\mathcal{H}_{T_n}$ denote the history up to and including arrival T_n. Then we have:

$$\lambda(t|\mathcal{H}_{T_n}) = \frac{f(t|\mathcal{H}_{T_n})}{1 - F(t|\mathcal{H}_{T_n})}. \tag{5.7}$$

The representation of the conditional intensity function in Theorem 5.1.16 is originally called the *hazard function*. The proof starts by considering the infinitesimal interval around t, where T_{n+1} indicates the next arrival after observing the last arrival $T_k \in \mathcal{H}_t$.

Proof.

$$\begin{aligned}
\lambda(t|\mathcal{H}_t)dt &= \mathbb{E}\left[N([t,t+dt]) \mid \mathcal{H}_t\right] \\
&= \mathbb{P}\left(T_{n+1} \in [t,t+dt] \mid \mathcal{H}_t\right) \\
&= \mathbb{P}\left(T_{n+1} \in [t,t+dt] \mid T_{n+1} \notin (T_n,t), \mathcal{H}_{T_n}\right) \\
&= \frac{\mathbb{P}\left(T_{n+1} \in [t,t+dt], T_{n+1} \notin (T_n,t) \mid \mathcal{H}_{T_n}\right)}{\mathbb{P}\left(T_{n+1} \notin (T_n,t) \mid \mathcal{H}_{T_n}\right)} \\
&= \frac{\mathbb{P}\left(T_{n+1} \in [t,t+dt] \mid \mathcal{H}_{T_n}\right)}{\mathbb{P}\left(T_{n+1} \notin (T_n,t) \mid \mathcal{H}_{T_n}\right)} \\
&= \frac{f\left(t \mid \mathcal{H}_{T_n}\right) dt}{1 - F\left(t \mid \mathcal{H}_{T_n}\right)},
\end{aligned}$$

hence indeed $\lambda(t|\mathcal{H}_t) = \frac{f(t|\mathcal{H}_{T_n})}{1-F(t|\mathcal{H}_{T_n})}$. □

5.1.6. Compensator

For parameter estimation, often the integrated intensity function is needed. We, therefore, define the integrated intensity here already. The integrated intensity is called the *compensator* of the counting process.

Definition 5.1.17: Compensator [42]

The compensator function $\Lambda(t)$ is defined as the unique, $\mathcal{H}_t$-predictable, non-decreasing function $\Lambda(t)$ such that $\Lambda(0) = 0$ and $N(t) = M(t) + \Lambda(t)$ almost surely for $t \geq 0$ where $M(t)$ is an $\mathcal{H}_t$- local martingale.

The existence of $\lambda(t|\mathcal{H}_t)$ is guaranteed by the Doob-Meyer decomposition. For a counting process $N(\cdot)$ with conditional intensity function $\lambda(t|\mathcal{H}_t)$, the compensator can be represented as

$$\Lambda(t) = \int_0^t \lambda(s|\mathcal{H}_s)ds.$$

5.2. One-dimensional Hawkes processes

Although deterministic non-homogeneous Poisson processes offer the flexibility to vary the intensity function over time, they still assume independent arrivals. In other words, the probability of an arrival occurring at time t is independent of the history $\mathcal{H}_t$ at time t.
In many real-world applications, we know that the occurrence of an arrival affects the occurrence of an upcoming arrival, meaning arrivals actually happen *dependently* of each other. To model these types of processes, we turn our attention to *self-exciting processes*. The key characteristic is that each arrival *excites* the process in the sense that the chance of a subsequent arrival is increased for some time period after the initial arrival. Examples of this are

- Earthquakes; where the occurrence of an earthquake increases the probability of seeing more earthquakes in the form of aftershocks [60].
- Stock trades; where one market trade triggers other traders to act according to market movement by hedging their position and selling or buying [27].

To be able to model this dependent behavior, we will now turn our attention to Hawkes processes. In a Hawkes process, the intensity of events at any given time is determined by two factors: the *background intensity*, which represents the average rate of events over time, and the *memory kernel*, which describes the influence of past events on the current intensity. This background intensity can be seen as the regular intensity in the Poisson processes and can be chosen to be either a constant value, as for homogeneous Poisson processes, or as being an intensity function over time, as for non-homogeneous Poisson processes.

Definition 5.2.1: Hawkes processes [43]

A counting process $N(\cdot)$ is a Hawkes process with associate history $\mathcal{H}_t$ if for the conditional intensity function it holds

$$\lambda(t|\mathcal{H}_t) = \mu(t) + \int_0^t \phi(t-u)\mathrm{d}N(u) = \mu(t) + \sum_{i:T_i<t} \phi(t-T_i), \tag{5.8}$$

for some $\mu(\cdot) : \mathbb{R}_+^* \to \mathbb{R}_+$ and $\phi : \mathbb{R}_+^* \to \mathbb{R}_+^*$. Denote again the point process $\mathbb{T} = \{T_1, T_2, \cdots, \}$, where $\mathbb{T}$ is the point process associated to the counting process $N(\cdot)$.

Here, in the last equality of Equation (5.8), the stochastic Stieltjes integral is used.

Remark 5.2.2. It should be emphasized that we went from a constant intensity λ_0 for homogeneous Poisson processes, to a time-dependent intensity function $\lambda(t)$ for non-homogeneous Poisson processes, to a history-dependent conditional intensity function $\lambda(t|\mathcal{H}_t)$ for Hawkes processes. It should be noted that Hawkes processes are not at all memoryless and therefore a clear example of a non-Markovian process. Hawkes processes can therefore be seen as the non-Markovian counterpart of Poisson processes.

Remark 5.2.3. If we set the memory kernel $\phi(\cdot)$ to 0, we obtain the non-homogeneous Poisson process $\lambda(t|\mathcal{H}(t)) = \mu(t)$. If additionally, we set $\mu(t) = \mu_0$ to a constant, we obtain a homogeneous Poisson process $\lambda(t|\mathcal{H}(t)) = \mu_0$.

5.2.1. Memory kernel

The choice of the memory kernel $\phi(\cdot)$ in a Hawkes process characterizes the effect prior arrivals have on the conditional intensity function. In other words, the memory kernel determines how the appearance of prior arrival $\mathcal{H}_t$ will influence the likelihood of an arrival appearing in an extension of the observation region $[t, t+\Delta t)$.

Remark 5.2.4. The memory kernel is often referred to as the *triggering*, *impact*, or *excitation function* in related literature.

It is common to choose a memory kernel that is monotonically decreasing [44, 69, 82, 84]. This way, more recent arrivals have a higher influence on the conditional intensity function than arrivals further back in history.

Exponential kernel

A common choice for the memory kernel is an exponential kernel.

Definition 5.2.5: Exponential memory kernel

Memory kernel ϕ is called an exponential memory kernel if for every $t \in \mathbb{R}_+^*$,

$$\phi(t) = \alpha \cdot \exp\left(-\frac{t}{\tau}\right), \tag{5.9}$$

for some fixed $\alpha \in \mathbb{R}_+, \tau \in \mathbb{R}_+^*$.

Here, the parameter α is referred to as the *excitation*, which models the instantaneous increase a new arrival T_n has on $\lambda(t|\mathcal{H}_t)$. Immediately after the jump, the conditional intensity decreases according to $\exp\left(-\frac{t-T_n}{\tau}\right)$. The parameter τ is referred to as the *characteristic time* and it has a unit of time in seconds. The parameter τ governs the rate of decay, where a large value of τ implies slow decay towards the baseline intensity.

Remark 5.2.6. Instead of using the characteristic time τ, it is common to use *decay* $\beta = \frac{1}{\tau}$, and therefore obtain exponential memory kernel $\phi(t) = \alpha \cdot \exp(-\beta \cdot t)$. However, using characteristic time τ instead of decay β gives us a more intuitive feeling for how fast a function is decaying: for every τ seconds passed, the excitation α gets reduced by a factor of e.

Power law kernel
Although the exponential memory kernel is one of the most common kernel functions for modeling Hawkes processes, situations arise where an exponential kernel does not provide satisfying behavior. An example is that of modeling tweet cascading behavior [64]. As an alternative, the power law kernel is proposed.

Definition 5.2.7: Power law memory kernel

Memory kernel ϕ is called a power law memory kernel if for every $t \in \mathbb{R}_+^*$,

$$\phi(t) = \frac{K}{(c+t)^p}, \tag{5.10}$$

for some fixed $K \in \mathbb{R}_+, c, p \in \mathbb{R}_+^*$.

Note that this memory kernel induces a statistical model with three parameters (K, c and p) instead of two in the case of exponential kernels (α, τ).

Gamma distributed memory kernel
Additionally, we consider a more exotic kernel to accommodate for more general possibilities, namely a Gamma distributed memory kernel, as developed by Lesage et al. [44],

Definition 5.2.8: Gamma memory kernel

Memory kernel ϕ is called a Gamma memory kernel if for every $t \in \mathbb{R}_+^*$,

$$\phi(t) = \alpha \frac{t^{k_1-1} \exp\left(-\frac{t}{k_2}\right)}{k_2^{k_1} \Gamma(k_1)}, . \tag{5.11}$$

for some fixed $\alpha \in \mathbb{R}_+^*, k_1, k_2 \in \mathbb{R}_+^*$.

Here $\Gamma(\cdot)$ is the Gamma function, k_1 the scale parameter, and k_2 the shape parameter. For $k_1 = 1$, we again obtain the exponential memory kernel. We explicitly mention this kernel, as for $k_1 > 1$, we can model a delay in the resulting self-excitation. This can be particularly useful for large-scale service systems, where a delay between a warning and a service outage can be considered. We will not apply the kernel in this thesis but instead, but instead consider the usage of the Gamma distribution in future work (see Section 7.1).

Non-parametric approaches for memory kernel estimation
It should be noted that different choices for the shape of the kernels correspond to different models. Non-parametric estimators can also be used to accommodate more general possibilities. A disadvantage is that non-parametric estimators scale poorly for high-dimensional multivariate Hawkes processes [4], which we will thoroughly employ in Section 5.3.

5.2.2. Estimation procedures

To estimate the parameters of the memory kernel, the two most common methods used in literature are a *maximum-likelihood* estimate and a *least squares* estimate. In this section, we will elaborate on both approaches.

General likelihood
Assume $\mu(t)$ can be parameterized by a vector of parameters $\boldsymbol{\mu} := (\mu_1, \mu_2, \dots,)$. For instance as in Section 3.2.1, Equation (3.1), we obtain $\boldsymbol{\mu} := (\beta_0, \beta_1, \dots, \beta_4)$.

Theorem 5.2.9: Likelihood for Hawkes processes [62]

Let $N(\cdot)$ be a point process on the time interval $[0;T]$ for $T > 0$. Let $\mathbb{T} = \{T_1, T_2, \ldots, T_n\}$ denote the realization of the counting process $N(\cdot)$. The likelihood function $L(\theta; \mathcal{H}_T)$ for $N(\cdot)$ can be written in the form

$$L(\theta; \mathcal{H}_T) = \left(\prod_{i=1}^{n} \lambda(T_i | \mathcal{H}_{T_i}) \right) \cdot \exp\left(- \int_0^T \lambda(s|\mathcal{H}_s) ds \right). \tag{5.12}$$

Proof. The probability of the next arrival T_{i+1} appearing in the infinitesimal interval $(t, t + dt)$ given history $\mathcal{H}_{T_i}$ is denoted as $f_{T_{i+1}} dt$ [65]. To obtain the likelihood given $\mathbb{T} = \{T_1, T_2, \ldots T_n\}$ in time interval $[0;T]$, we calculate $L(\theta; \mathcal{H}_T)$

$$L(\theta; \mathcal{H}_T) = f(T_1, T_2, \ldots, T_n) = \prod_{i=1}^{n} f(T_i \mid \ldots, T_{i-2}, T_{i-1}) = \prod_{i=1}^{n} f(T_i \mid \mathcal{H}_{T_{i-1}}). \tag{5.13}$$

Using the hazard function from Equation (5.7), it can be obtained

$$\lambda(t|\mathcal{H}_t) = \frac{f(t|\mathcal{H}_t)}{1 - F(t|\mathcal{H}_t)} = \frac{\frac{d}{dt}[F(t|\mathcal{H}_t)]}{1 - F(t|\mathcal{H}_t)} = -\frac{d}{dt} \ln [1 - F(t|\mathcal{H}_t)]. \tag{5.14}$$

Denote the last known arrival time prior to t as T_i. Integrating from T_i to t results in

$$\int_{T_i}^{t} \lambda(s|\mathcal{H}_s) ds = \int_{T_i}^{t} \left(-\frac{\mathrm{d}}{\mathrm{d}s} \ln [1 - F(s|\mathcal{H}_s)] \right) ds = \ln(1 - F(T_i|\mathcal{H}_{T_i})) - \ln(1 - F(t|\mathcal{H}_{T_i})). \tag{5.15}$$

Since by definition $T_{i+1} > T_i$, we have $F(T_i|\mathcal{H}_{T_i}) = 0$. Therefore the second term cancels and we obtain

$$\int_{T_i}^{t} \lambda(s|\mathcal{H}_s) ds = -\ln(1 - F(t|\mathcal{H}_{T_i})), \tag{5.16}$$

and therefore the denominator in the conditional intensity function from Equation (5.7) becomes

$$1 - F(t|\mathcal{H}_{T_i}) = \exp\left(- \int_{T_i}^{t} \lambda(s|\mathcal{H}_s) ds \right). \tag{5.17}$$

Rewriting the hazard function from Equation (5.7) and substituting the conditional cumulative distribution function from Equation (5.17) results in

$$f(t|\mathcal{H}_{T_i}) = \lambda(t|\mathcal{H}_{T_i}) \cdot [1 - F(t|\mathcal{H}_{T_i})] = \lambda(t|\mathcal{H}_{T_i}) \cdot \exp\left(- \int_{T_i}^{t} \lambda(s|\mathcal{H}_{T_i}) ds \right). \tag{5.18}$$

Finally, plugging the result from Equation (5.18) into the likelihood function from Equation (5.13), we obtain indeed

$$L(\theta; \mathcal{H}_T) = \prod_{i=1}^{n} f(T_i | \mathcal{H}_{T_{i-1}}) = \prod_{i=1}^{n} \lambda(T_i|\mathcal{H}_{T_{i-1}}) \cdot \exp\left(- \int_{T_{i-1}}^{T_i} \lambda(s|\mathcal{H}_s) dt \right) = \left[\prod_{i=1}^{n} \lambda(T_i|\mathcal{H}_{T_i}) \right] \cdot \exp\left(- \int_0^T \lambda(s|\mathcal{H}_s) ds \right) \tag{5.19}$$

□

Often the logarithm of the likelihood in Equation (5.12) is considered. This results in the *log-likelihood*

$$\mathcal{L}(\theta; \mathcal{H}_T) = \log(L(\theta; \mathcal{H}_T)) = \sum_{i=1}^{n} \log(\lambda(T_i|\mathcal{H}_{T_i})) - \int_0^T \lambda(s|\mathcal{H}_s) ds. \tag{5.20}$$

The logarithmic function is monotonically increasing so maximizing the likelihood is equivalent to maximizing the log-likelihood. Additionally, maximizing the log-likelihood is equivalent to minimizing the *negative log-likelihood*

$$-\mathcal{L}(\theta;\mathcal{H}_T) = -\log(L(\theta;\mathcal{H}_T)) = \int_0^T \lambda(s|\mathcal{H}_s)ds - \sum_{i=1}^{n} \log(\lambda(T_i|\mathcal{H}_{T_i})). \tag{5.21}$$

Exponential memory kernel likelihood
The likelihood from Equation (5.12) using the exponential memory kernel from Equation (5.9), can be written in the form

Corollary 5.2.10: Likelihood for exponential memory kernel

The likelihood function $L(\theta)$ for $N(\cdot)$ using an exponential kernel can be written in the form

$$L(\theta;\mathcal{H}_T) = \prod_{i=1}^{n} \left[\mu(T_i) + \alpha \sum_{j=i}^{i-1} \exp\left(-\frac{(T_i - T_j)}{\tau} \right) \right] \cdot \exp\left(- \int_0^T \left(\mu(s) + \alpha\tau \sum_{i=1}^{k} \exp\left(-\frac{(T_i - T_j)}{\tau} \right) \right) ds \right). \tag{5.22}$$

5.2.3. Consistency of the MLE for one-dimensional Hawkes processes

We consider a family of parameterized stationary point processes $\{\lambda_\theta(t); \theta \in \Theta \subset \mathbb{R}^d\}$ which are assumed to correspond uniquely to the stationary counting process $\{N_\theta(t); \theta \in \Theta\}$. If we take for example the family of Hawkes processes with constant baseline intensity and exponential memory kernel, then $\theta = \{\mu, \alpha, \tau\}$ with $\Theta = {\mathbb{R}_+}^3 \subset \mathbb{R}^3$.

Remark 5.2.11. In this subsection, we will no longer consider a baseline intensity *function* but instead consider a constant baseline. Subsequently, all terms, $\mu(s)$ will be replaced by a constant μ. We choose to do so because a non-constant baseline function violates the stationarity assumption. Stationarity on the other hand is required for applying the usual theorems for proving consistency, as can be seen from the first assumption from Theorem 5.2.12.

In Theorem 5.2.9 we defined the likelihood function for Hawkes processes. The maximum likelihood estimator $\hat{\theta}_{MLE} = \hat{\theta}(T_i; 0 \leq T_i \leq T)$ is defined as the estimator θ which maximizes Equation (5.12), under observations from the counting process $N_{\theta_0}(\cdot)$. Here θ_0 denotes the true parameter values generating the observed counting process $N_{\theta_0}(\cdot)$.

It is now of interest to prove that under a set of assumptions, the maximum likelihood is consistent. That is, to prove that the estimator $\hat{\theta}_{MLE}$ converges in probability to θ_0 for $T \to \infty$. Formally, this means

$$\lim_{T\to\infty} \mathbb{P}(|\theta_0 - \hat{\theta}_{MLE}| \geq \epsilon) = 0, \text{ for all } \epsilon > 0.$$

This has first been proven by Ogata [59] for general intensity functions. Here a proof under observations of the infinite past $\mathcal{H}_{(\infty,t)}$ is given, which is subsequently extended to observation under $\mathcal{H}_{[0,t)}$. We will state the six assumptions from Ogata for general intensity processes and show consistency.

Consistency proof
Given the following six assumptions

Consistency of the MLE

1. The counting process $N(\cdot)$ is stationary, ergodic and absolutely continuous with respect to the standard Poisson process on any finite interval.
2. Θ is a compact metric space with some metric ρ, and $\Theta \in \mathbb{R}^d$.
3. λ_θ is predictable for all Θ. $\lambda_\theta(t,\omega)$ is continuous in θ and $\lambda_\theta(0,\omega) > 0$ almost surely for any $\theta \in \Theta$.
4. $\lambda_{\theta_1}(0,\omega) = \lambda_{\theta_2}(0,\omega)$ almost surely if and only if $\theta_1 = \theta_2$.
5. For any $\theta \in \Theta$, there exists a neighbourhood $U = U(\theta)$ of θ such that for all $\theta' \in U$,
$$|\lambda_{\theta'}(0,\omega)| \leq \Lambda_0(\omega) \text{ and } |\log \lambda_{\theta'}(0,\omega)| \leq \Lambda_1(\omega)$$
where Λ_0 and Λ_1 are random variables with finite second moments.
6. For any $\theta \in \Theta$, there exists a neighbourhood $U(\theta)$ of θ such that for all $\theta' \in U$
   - $\sup_{\theta' \in U} |\lambda_{\theta'}(t,\omega) - \lambda^*_{\theta'}(t,\omega)| \to 0$ in probability as $t \to \infty$,
   - $\sup_{\theta' \in U} |\log \lambda^*_{\theta'}(t,\omega)|$ has, for some $\alpha > 0$, finite $(2+\alpha)$th moment uniform bounded with respect to t.

We state the following theorem:

Theorem 5.2.12: Consistency of the MLE

Under the six assumptions above, we find that the maximum likelihood estimator $\hat{\theta}_{MLE}$ converges to θ_0 in probability as $T \to \infty$.

Proof. By Assumptions (3) and (5), we can directly obtain

$$\mathbb{E}_{\theta_0}\Big[\inf_{\theta' \in U} \lambda_{\theta'}(0,\omega)\Big] \to \mathbb{E}_{\theta_0}\big[\lambda_\theta(0,\omega)\big] \tag{5.23}$$

and

$$\mathbb{E}\left[\lambda_{\theta_0}(0,\omega)\log\left\{\frac{\lambda_{\theta_0}(0,\omega)}{\sup_{\theta' \in U}\lambda_{\theta'}(0,\omega)}\right\}\right] \to \mathbb{E}\left\{\lambda_{\theta_0}(0,\omega)\log\frac{\lambda_{\theta_0}(0,\omega)}{\lambda_\theta(0,\omega)}\right\} \tag{5.24}$$

as the neighbourhood U of θ shrinks to $\{\theta\}$. Let U_0 be an open neighborhood of θ_0. Then by definition of the Kullback-Leibler divergence (see Section B.1.1) and Assumption (4), there is a positive ε such that $\mathbb{E}_{\theta_0}\{KL(\theta_0;\theta)\} \geq 3\varepsilon$ for any $\theta \in \Theta\backslash U_0$.
Now for any $\theta \in \Theta\backslash U_0$, we can choose U small enough so that

$$\begin{aligned}&\mathbb{E}\left[\inf_{\theta' \in U}\lambda_{\theta'}(0,\omega) - \lambda_{\theta_0}(0,\omega) + \lambda_{\vartheta_0}(0,\omega)\log\left\{\frac{\lambda_{\theta_0}(0,\omega)}{\sup_{\theta' \in U}\lambda_{\theta'}(0,\omega)}\right\}\right] \\ &\geq \mathbb{E}\{KL(\theta_0;\theta)\} - \varepsilon.\end{aligned} \tag{5.25}$$

Select a finite number of θ_s such that $U_s = U_{\theta_s}, 1 \leqq s \leqq N$, cover $\Theta\backslash U_0$. Since $\inf_{\theta' \in U}\lambda_{\theta'}(t,\omega)$ and $\sup_{\theta' \in U}\lambda_{\theta'}(t,\omega)$ are predictable processes, by the predictability of the stationary process there exists, for any $\varepsilon > 0, T_0 = T_0(\varepsilon)$ depending on the sample such that for any $T > T_0$ and $s = 1, 2, \cdots, N$,

$$\begin{aligned}&\frac{1}{T}L_T(\theta_0) - \sup_{\theta \in U_s}\frac{1}{T}L_T(\theta) \\ &\geq \frac{1}{T}\int_0^T\left\{\inf_{\theta \in U_s}\lambda_\theta(t,\omega) - \lambda_{s_0}(t,\omega)\right\}dt + \frac{1}{T}\int_0^T \log\frac{\lambda_{\theta_0}(t,\omega)}{\sup_{\theta \in U_s}\lambda_\theta(t,\omega)}dN(t) \\ &\geq \mathrm{E}\{KL(\theta_0;\theta)\} - 2\varepsilon \geq \varepsilon.\end{aligned} \tag{5.26}$$

It follows that there exists $T_1 = T_1(\varepsilon, U_0) > T_0$ such that for all $T > T_1$

$$\sup_{\theta \in U_0} L_T(\theta) \geq \sup_{\theta \in \Theta \setminus U_0} L_T(\theta) + \varepsilon T. \tag{5.27}$$

So far, we considered the intensity function under infinite past $H_{-\infty,t}$

$$\lambda(t, \omega) = \lim_{\delta \to 0} \frac{1}{\delta} \mathrm{P}\left[N\{[t, t+\delta)\} > 0 \mid H_{-\infty,t}\right]. \tag{5.28}$$

However, using Assumption 6, inequalities (5.26) and (5.27) remain valid for the intensity under $H_{0,t}$ with probabilities going to 1 for $T \to \infty$

$$\lambda^*(t, \omega) = \lim_{\delta \to 0} \frac{1}{\delta} \mathrm{P}\left[N\{[t, t+\delta)\} > 0 \mid H_{0,t}\right] = \mathrm{E}\left\{\lambda(t, \omega) \mid H_{0,t}\right\}. \tag{5.29}$$

This implies $\hat{\theta}_{MLE} \in U_0$, which completes the proof. □

Remark 5.2.13. We take the integral of range $(-\infty, t)$, which becomes the sum for all integers i such that $t_i < t$. If instead, we take the integral on the range $(-\infty, t]$, then $\lambda_\theta(t, \omega)$ is no longer predictable.

Consistency for exponential memory kernel
A simulation study for obtaining the maximum likelihood estimates for Hawkes processes with an exponential memory kernel has been performed by Ogata [59]. For a Hawkes process with exponential memory kernel and constant baseline intensity, we have $\theta = (\mu, \alpha, \tau)$ such that $\alpha < \frac{1}{\tau}$.

Indeed, the Hawkes process with exponential memory kernel is stationary, ergodic, and absolutely continuous with respect to the standard Poisson process on any finite interval. Secondly, $\mu, \alpha \in \mathbb{R}_+$. We fix τ to a constant value for each of the computations. Therefore choosing μ, α in compact intervals results in a space Θ that is compact as well.

Consistency for non-stationary baseline function
Ogata [59] proofs consistency under a constant baseline intensity μ. However, as demonstrated in Section 3.2.1, stationarity for the number of arrivals over time is often violated, as the number of arrivals is dependent upon time, such as workday hours and weekends. If we incorporate Equation (3.1) into the baseline, we obtain a Hawkes process with baseline intensity function $\mu(t)$ and memory kernel $\phi(t)$ such that

$$\begin{aligned} \log_{10}(\mu(t)) &= \beta_0 + \beta_1 t + \beta_2 \sin(Wt) + \beta_3 \cos(Wt) + \beta_4 I_{\{t \in \text{weekend}\}} \\ \mu(t) &= 10^{\beta_0 + \beta_1 t + \beta_2 \sin(Wt) + \beta_3 \cos(Wt) + \beta_4 I_{\{t \in \text{weekend}\}}} \qquad (5.30) \\ \phi(t) &= \alpha \exp\left(\frac{t}{\tau}\right), \qquad (5.31) \end{aligned}$$

with $\theta = (\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \alpha, \tau)$.

We sketch the proof as proposed by Hall and Chen [26] for non-stationary baseline intensity. The proof assumes both the baseline intensity $\mu(\cdot; \theta)$ and the memory kernel $\phi(\cdot; \theta)$ are known up to some finite-dimensional parameter θ, which is indeed the case from Equations 5.30 and 5.31. Due to the absence of a stationary baseline function, we can not apply the asymptotic scenario of letting the observation time of the process tend to infinity. Instead, the proof states five conditions regarding the baseline intensity function $\mu(\cdot; \theta)$ and the memory kernel $\phi(\cdot; \theta)$. These five conditions are as follows:

1. Both the normalized baseline intensity $\mu(\cdot; \theta)$ and the excitation function $\phi(\cdot; \theta)$ are positive and continuous on $[0, 1]$ for all $\theta \in \Theta$.
2. The parameter space Θ is compact and its interior is connected and contains a d-dimensional nonempty open ball which, in turn, contains the true parameter.
3. For each $t \in [0, 1]$, the functions $\mu(t; \theta)$ and $\phi(t; \theta)$ are twice continuously differentiable in θ and their partial derivatives, up to order 2 with respect to θ, are all uniformly equicontinuous when regarded as families of functions of θ.

4. For each θ, $\partial_\theta \phi(t;\theta)$, and $\partial^2_\theta \phi(t;\theta)$ are continuously differentiable in t.
5. The matrix-valued function

$$I(\theta) = \int_0^1 \frac{\left\{\partial_\theta \mu(t;\theta) + \int_0^t \partial_\theta \phi(t-u;\theta) h(u;\theta) \mathrm{d}u\right\}^{\otimes 2}}{\mu(t;\theta) + \int_0^t \phi(t-u;\theta) h(u;\theta) \mathrm{d}u} \, \mathrm{d}t$$

is non singular at the true parameter value, where the function $h(t;\theta)$ is determined by the functions μ and ϕ and given as follows; let

$$R(t) = \gamma_1 \exp -(\gamma_2 - \gamma_1)t,$$

such that

$$h(t;\theta) = \mu(t;\theta) + \int_0^t R(t-u;\theta) \cdot \mu(u;\theta) du \tag{5.32}$$

$$= \mu(t;\theta) + \int_0^t \gamma_1 \exp\left(-(\gamma_2 - \gamma_1)(t-u)\right) \cdot \mu(u;\theta) du \tag{5.33}$$

$$= \mu(t;\theta) + \gamma_1 \exp\left(-(\gamma_2 - \gamma_1)t\right) \int_0^t \exp\left(-(\gamma_2 - \gamma_1)(-u)\right) \cdot \mu(u;\theta) du \tag{5.34}$$

The proof is based upon considering a sequence of self-exciting point processes with a non-constant baseline intensity $N^n(\cdot)$, where the sequence of point processes N^n_t has intensity function

$$\lambda^n_t = a_n \mu(t) + \int_0^t \phi(t-u) \mathrm{d}N(u)^n, \quad t \in [0,1], \tag{5.35}$$

where a_n is a proportionality constant that tends to infinity with index set n. In the sequel, the baseline intensity $\mu(t;\theta)$ tends to infinity as $n \to \infty$, and the memory kernel remains fixed.
For more details, see Chen and Hall [26]

Least squares
Although less common in literature, a least squares function can be defined as an alternative to the likelihood function.

Theorem 5.2.14: Least squares functional for Hawkes processes

Let $N(\cdot)$ be a Hawkes process on the time interval $[0;T]$. Let $\mathbb{T} = \{T_1, T_2, \ldots, T_n\}$ denote the realizations of the counting process $N(\cdot)$. The least squares function $R(\theta)$ for $N(\cdot)$ can be written in the form

$$R(\theta; \mathcal{H}_T) = \int_0^T \lambda(s|\mathcal{H}_s)^2 ds - 2 \sum_{i=1}^n \lambda(T_i|\mathcal{H}_{T_i}). \tag{5.36}$$

The least squares estimator $\hat{\theta}_{LSE} = \hat{\theta}(T_i; 0 \geq T_i \geq T)$ is defined as the estimator which minimizes Equation (5.36) under observation from the counting process $N(\cdot)$.
Minimizing the first term $\int_0^T \lambda(s|\mathcal{H}_s)^2 ds$ enforces low intensity at non-arrival times. Maximizing the second term $2\sum_{k=1}^n \lambda(T_n|\mathcal{H}_{T_n})$ enforces a high intensity at arrival times. The same analogy can be drawn with the *negative log-likelihood* from Equation (5.21), where maximizing the second term $\sum_{i=1}^n \log(\lambda(T_i|\mathcal{H}_{T_i}))$ enforces high intensity at arrival times.

The weak consistency of the least squares estimator of the one-dimensional Hawkes process was proved by [41] as a particular case of the more general marked Hawkes process framework.

5.3. Marked Hawkes processes

In Section 5.2, we saw that after an arrival t, an instantaneous excitation arises. This excitation is governed by memory kernel $\phi(\cdot)$. We observe only one process. Inference is therefore performed on counting process $N(\cdot)$.

In contrast to the one-dimensional Hawkes process, situations arise where we observe additional information for each arrival in the form of a *mark*. This results in a marked point process.

Definition 5.3.1: Marked point processes [19]

A marked process $\mathbb{T}_{MPP}$ on $\mathcal{X} \times \mathcal{U}$ is defined as a mapping from Ω to the space of counting processes $\mathbb{M}(\mathcal{X} \times \mathcal{U})$, meaning that each realization $\omega \in \Omega$ of a marked point process is a counting measure $\mathbb{T}_{MPP}(\omega) \in \mathbb{M}(\mathcal{X} \times \mathcal{U})$.

Although it is possible to assign a continuous mark space, we will restrict ourselves to a discrete mark space. Therefore the mark space will be denoted as in a subset of the natural numbers, $\mathcal{U} \subseteq \mathbb{N}$. Again utilizing the earthquake and stock example from the introduction of Section 5.2, we can generalize to marked Hawkes process as follows:

- *Earthquakes;* An example of the label could be the province of the earthquake. Here an earthquake in one province might strongly increase the conditional intensity in a neighboring province but do not affect provinces far away. Furthermore the self-excitation of increased probability for another earthquake in the same province, for instance, aftershocks, might variate per province [60].
- *Stocks;* An example of the label could be the individual stock. Here an increase in buy orders for stocks in a bank such as *ING* might strongly increase the number of buy orders for a bank such as *ABN Amro* but have no influence on the number of buy orders for a tech company like *Apple*. Marks offer us the flexibility to give a different triggering effect from stocks of the same marked type in contrast to stocks of an unrelated market type [9, 27].

Marked Hawkes processes are also self-exciting processes, but besides, also have cross-exciting between different marks. The intensity of a new arrival does not only depend on the number of prior arrivals but also on the label of the arrival. This allows the marked Hawkes process to capture more complex dependencies between arrivals.

This presents us to Research Question 3.

Research Question 3

How can the marked Hawkes process be employed to capture interactions among arrivals? And how can the interactions be estimated?

Remark 5.3.2. A marked Hawkes process is sometimes referred to as a multidimensional, multivariate, or mutually exciting Hawkes process. In this thesis, we will consistently refer to a marked Hawkes process.

5.3.1. Conditional intensity function

The marked point process now becomes

$$\mathbb{T}_{MPP} = \{(T_1, k_1), (T_2, k_2), \ldots, (T_n, k_n)\} \tag{5.37}$$

With T_i the arrival time and u_i the mark of the i^{th} appearance. We will assume the marks take values in the finite set of integers $u \in \mathcal{U} := \{1, \ldots, U\}$, with $U \ll n$ [37]. Each counting process $N_u(t)$ represents the number of arrivals up to time t having mark u.

$$N_u(t) = \sum_{\substack{(T_i, u_i) \in \mathbb{T}_{MPP} \\ u_i = u}} \mathbb{I}_{[0,t]}(T_i) \tag{5.38}$$

The conditional intensity function for the marked Hawkes process can now be defined.

Definition 5.3.3: Marked Hawkes Processes [42]

Consider a collection of U counting processes $\boldsymbol{N(t)} = \{N_1(t), N_2(t), \dots, N_U(t)\}$ with collective history $\mathcal{H}_t$. The conditional intensity function of each counting process $N_u(t)$ is given by

$$\begin{aligned}\lambda_u(t|\mathcal{H}_t) &= \mu_u(t) + \sum_{u'=1}^{U} \int_{-\infty}^{t} \phi_{uu'}(t-s) dN_u(s) \\ &= \mu_u(t) + \sum_{u=1}^{U} \sum_{\substack{(T_i,u_i)\in\mathcal{H}_t \\ u_i=u}} \phi_{uu'}(t-T_i),\end{aligned}$$

for $\mu_u(\cdot) : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ and $\phi_{uu'}(\cdot) : \mathbb{R}_0^+ \to \mathbb{R}_0^+$.

Exponential memory kernel

Similar as to one-dimensional Hawkes processes, we can now define marked Hawkes processes with exponential memory kernel.

Definition 5.3.4: Marked Hawkes processes with exponential memory kernel

Assume a similar setting as Definition 5.3.3. The conditional intensity function for mark u with exponentially decaying memory kernel $\phi_{uu'}(t-T_i) = \alpha_{uu'} \cdot \exp\left(-\frac{t-T_i}{\tau_{uu'}}\right)$ is defined as

$$\lambda_u(t|\mathcal{H}_t) = \mu_u(t) + \sum_{u'=1}^{U} \sum_{\substack{(T_i,u_i)\in\mathcal{H}_t \\ u_i=u'}} \alpha_{uu'} \cdot \exp\left(-\frac{t-T_i}{\tau_{uu'}}\right),$$

for non-negative constants $\{\alpha_{uu'}, \tau_{uu'} : u, u' \in \mathcal{U} := \{1, \cdots, U\}\}$.

Whereas in the one-dimensional setting, we only obtained a single *self-exciting* parameter α, we now obtain U self-exciting parameters $\alpha_{11}, \alpha_{22}, ..., \alpha_{UU}$, as well as $U(U-1)$ cross-excitation term $\{\alpha_{uu'} : u, u' \in \mathcal{U}, u \neq u'\}$. Together these U^2 parameters form the excitation matrix $\mathcal{A}$.

Definition 5.3.5: Excitation matrix $\mathcal{A}$

Given a marked Hawkes process with an exponential memory kernel $\phi_{uu'}(t) = \alpha_{uu'} \cdot \exp\left(-\frac{t}{\tau_{uu'}}\right)$. All terms $\{\alpha_{uu'} : u, u' \in \mathcal{U}\}$ together make up excitation matrix $\mathcal{A} \in \mathbb{R}^{U\times U}$.

Remark 5.3.6. $\alpha_{uu'}$ captures the increase in the conditional intensity of arrival type u given an arrival of type u'. It should be noted that in general $\alpha_{uu'} \neq \alpha_{u'u}$, because the influence of u on u' may not be the influence of u' on u.

Similarly, for the characteristic time, we can define the characteristic time matrix $\mathcal{T}$.

Definition 5.3.7: Characteristic time matrix $\mathcal{T}$

Given a marked Hawkes process with an exponential memory kernel $\phi_{uu'}(t) = \alpha_{uu'} \cdot \exp\left(-\frac{t}{\tau_{uu'}}\right)$ All terms $\{\tau_{uu'} : u, u' \in \mathcal{U}\}$ together make up characteristic time matrix $\mathcal{T} \in \mathbb{R}^{U\times U}$.

Besides excitation matrix $\mathcal{A}$ and characteristic time matrix $\mathcal{T}$, we also have U baseline functions $\{\mu(t)_u : u \in \mathcal{U}\}$. In order to ensure stationarity as well as to simplify the estimation later on in Chapter 6, we assume a constant baseline for each of the counting processes $\{N_u(t) : u \in \mathcal{U}\}$. This results in baseline vector $\mathcal{M}$.

Definition 5.3.8: Baseline vector $\mathcal{M}$

Given a marked Hawkes process with a constant baseline intensity. All terms $\{\mu_u : u \in \mathcal{U}\}$ together make up baseline vector $\mathcal{U} \in \mathbb{R}^U$.

The estimation of the constant baseline will not play a notable role in this thesis, as we will see in Section 5.3.2 that our core interest is centered around the estimated excitation matrix $\hat{\mathcal{A}}$.

5.3.2. Granger causality

In Section 4.2, we saw how to separate event and incident arrivals into different groups, either by clustering or by parsing the log messages. Our goal is now to be able to causally relate which groups of messages can be associated with subsequent groups of messages. One of the common frameworks for marked Hawkes processes is Granger causality. This framework is often applied in discrete-time continuous-valued time series [24, 54, 78]. Granger causality is a statistical concept that refers to the idea that one time series can be used to predict another time series. In other words, it suggests that past values of time series can be used to predict future values of another time series.

Granger causality solely offers insight into forecasting ability and may not reveal the true causal link between two variables. Even if event and incident records are influenced by a shared third process with varying lags, it is possible to fail to reject the alternative hypothesis of Granger causality. Granger causality assesses only the sequencing of occurrences for prediction purposes without delving into the underlying causal structure. While it is suitable for forecasting performance evaluation, it does not serve the purpose of determining the theoretical model driving the forecast. Additionally, the two variables under consideration must exhibit stationarity. This once again emphasizes the need for a constant baseline intensity, which ensures the marked Hawkes process is indeed stationary.

Similarly as Xu et al [78], we are interested in identifying, if possible, a subset of marks $\mathcal{V} \subset \mathcal{U}$ for a type$-u$ arrival such that $\lambda_u(t)$ only depends on the historical arrivals of type $\mathcal{V}$, denoted as $\mathcal{H}_t^{\mathcal{V}}$, and not those of the other types, denoted $\mathcal{H}_t^{\mathcal{U}\setminus\mathcal{V}}$. In other words, $\lambda_u(t|\mathcal{H}_t^{\mathcal{U}}) = \lambda_u(t|\mathcal{H}_t^{\mathcal{V}})$. First, we will define Local independence.

Definition 5.3.9: Local independence

For a subset $\mathcal{V} \subset \mathcal{U}$, let $N_{\mathcal{V}} = \{N_u(t) \mid u \in \mathcal{V}\}$. The filtration $\mathfrak{F}_t^{\mathcal{V}}$ is defined as $\sigma\{N_u(s) \mid s \le t, u \in \mathcal{V}\}$, i.e., the smallest σ-algebra generated by the random processes.

The counting process N_u is locally independent of $N_{u'}$, given $N_{\mathcal{U}\setminus\{u,u'\}}$, if the intensity function $\lambda_u(t)$ is measurable with respect to $\mathfrak{F}_t^{-u'}$ for all $t \in [0, T]$. Otherwise N_u is locally dependent of $N_{u'}$.

In particular, $\mathfrak{F}_t^u$ is the internal filtration of the counting process $N_u(t)$ while $\mathfrak{F}_t^{-u}$ is the filtration for the subset $\mathcal{U}\setminus\{u\}$. With this definition, we can construct the so-called Granger-causality graph $G = (\mathcal{U}, \mathcal{E})$ with mark space $\mathcal{U}$ as the nodes and the directed edges indicating the lack of local independence, i.e., $u' \to u \in \mathcal{E}$ if type- u' arrival Granger-causes type- u arrival. We can now relate the memory kernel $\phi_{uu'}(\cdot)$ to the Granger causality graph $G = (\mathcal{U}, \mathcal{E})$.

Theorem 5.3.10: Granger causality and the memory kernel

Given a Hawkes process with conditional intensity function defined in Definition 5.3.3 and Granger causality graph $G = (\mathcal{U}, \mathcal{E})$. Let furthermore $dN_{u'}(t - s) > 0$ for all $0 \le s < t \le T$. Then

$$\phi_{uu'}(t) = 0 \text{ for all } t \text{ in } [0, \infty) \iff u' \to u \notin \mathcal{E}. \tag{5.39}$$

In practice we only observe arrivals in the interval $[0, T]$ instead of $[0, \infty)$. We hope our time window of two years is sufficient to approximate the theoretical setting of Theorem 5.3.10.

Granger causality for exponential memory kernels
For Hawkes processes with an exponential memory kernel, it can trivially be obtained that

$$\alpha_{uu'} = 0 \iff \phi_{uu'}(t) = \alpha_{uu'} \cdot \exp\left(-\frac{t}{\tau_{uu'}}\right) = 0 \text{ for all } t \text{ in } [0, \infty).$$

As $\tau_{uu'} > 0$, this results in the following corollary from Theorem 5.3.10:

Corollary 5.3.11: Granger causality for exponential memory kernels

Given a Hawkes process with conditional intensity function as defined in Definition 5.3.3 and exponential memory kernel $\phi_{uu'}(t) = \alpha_{uu'} \cdot \exp\left(-\frac{t}{\tau_{uu'}}\right)$. Then it follows

type-u arrivals are Granger-caused by type-u' arrivals if and only if $\alpha_{uu'} > 0$.

For Hawkes processes with exponential memory kernel, learning the Granger causal graph $G = (\mathcal{U}, \mathcal{E})$ is therefore equivalent to learning the excitation matrix $\mathcal{A}$. Similarly, if $\alpha_{uu'} = 0$, arrival instances of u-type are Granger non-causal to those of the u' type.

5.3.3. Estimation procedure

Similarly to one-dimensional Hawkes processes from Section 5.2, we can generalize the likelihood function from Theorem 5.2.9 and the least-squares function from 5.2.14 to marked Hawkes processes.

Likelihood function
We can now start by writing down the likelihood function in the marked framework and specify further in the case of an exponential kernel. We will start with the general multivariate or 'marked' likelihood. Recall again that in the marked framework, a mark $u \in \mathcal{U}$ is available for each arrival. Besides $T_1, \dots, T_n$ on $[0, T]$, we also obtain a sequence $u_1, \dots, u_n$, where u_i is the type of arrival T_i

Definition 5.3.12: Marked log likelihood function

Consider a collection of U counting processes $\boldsymbol{N(t)} = \{N_1(t), N_2(t), \dots, N_U(t)\}$ with collective history $\mathcal{H}_t$ on $[0, T]$ for finite positive T and let $T_1, T_2, ..., T_n$ together with marks $u \in \mathcal{U} = \{1, \dots, U\}$. Then, the marked log-likelihood function $\mathcal{L}(\theta; \mathcal{H}_T)$ of such N can be written in the form

$$\mathcal{L}(\theta; \mathcal{H}_T) = \frac{1}{T} \sum_{u=1}^{U} \Big(\int_0^T \log\left(\lambda_u\left(s|\mathcal{H}_s\right)\right) dN_u - \int_0^T \lambda_u(s|\mathcal{H}_s) ds \Big). \tag{5.40}$$

Consistency of the MLE for marked Hawkes processes
A consistency proof for the MLE for marked Hawkes processes is given by Guo et al. [25]. That is, for any neighborhood U_0 of θ_0

$$\lim_{T \to \infty} \mathbb{P}(|\theta_0 - \hat{\theta}_{MLE}| \geq \epsilon) = 0, \text{ for all } \epsilon > 0.$$

They assume a constant baseline intensity $\{\mu_u : u \in \mathcal{U}\}$ associated with the respective intensity function $\lambda_u(t|\mathcal{H}_t)$. Their proof is rather similar to the proof of Theorem 5.2.12. Moreover, they extend the proof to show consistency for the maximum likelihood estimator under ℓ_1 regularization.

Least-squares function
In a similar fashion, we can define the marked least squares function

Definition 5.3.13: Marked least squares function [9]

Consider a collection of U counting processes $\boldsymbol{N(t)} = \{N_1(t), N_2(t), \ldots, N_U(t)\}$ with collective history $\mathcal{H}_t$ on $[0, T]$ for finite positive T and let $T_1, T_2, ..., T_n$ together with marks $u \in \boldsymbol{\mathcal{U}} = \{1, \ldots, U\}$. Then, the least-squares function $R(\theta; \mathcal{H}_T)$ of such N can be written in the form

$$R(\theta; \mathcal{H}_T) = \frac{1}{T} \sum_{u=1}^{U} \left(\int_0^T \lambda_u(s|\mathcal{H}_s)^2 ds - 2 \int_0^T \lambda_u(s|\mathcal{H}_s) dN_u(s) \right). \tag{5.41}$$

Kircher [41] proved the weak consistency of the least square estimator under the marked Hawkes process framework. The proof largely depends on matrix manipulations and is rather technical.

5.4. Estimating Hawkes processes using the Tick library

The *Tick* library [3] is a Python library designed for the statistical learning of time-dependent systems, such as point processes. The primary objective of the Hawkes Tick module is to provide tools and functions for simulating, modeling, and analyzing Hawkes processes efficiently. The library offers estimators to infer the parameters of a Hawkes process from observed data.

Two estimators that directly estimate the parameters that optimize Equations 5.40 and 5.41 are the `tick.hawkes.HawkesExpKern` and `tick.hawkes.HawkesADM4` estimators. Both estimators are designed for marked Hawkes processes with exponential kernels and constant baseline intensity. Let us recall that, for mark $u \in \boldsymbol{\mathcal{U}} := \{1, \ldots, U\}$, the conditional intensity at time t equals

$$\lambda_u(t|\mathcal{H}_t) = \mu_u(t) + \sum_{u'=1}^{U} \sum_{\substack{(T_i, u_i) \in \mathcal{H}_t \\ u_i = u'}} \alpha_{uu'} \cdot \exp\left(-\frac{t - T_i}{\tau_{uu'}} \right).$$

Tick exponential memory kernel parametrization

The parameterization of the exponential memory kernel from Tick library slightly deviates from our parametrization in Definition 5.3.4, namely

$$\phi_{uu'}(t) = \alpha'_{uu'} \beta'_{uu'} \exp(-\beta'_{uu'} \cdot t) 1_{t>0},$$

instead of

$$\phi_{uu'}(t) = \alpha_{uu'} \cdot \exp\left(-\frac{t}{\tau_{uu'}} \right).$$

Once we fix $\tau_{uu'}$ and set $\beta'_{uu'} = \frac{1}{\tau_{uu'}}$, we obtain

$$\alpha'_{uu'} = \frac{\alpha_{uu'}}{\beta'_{uu'}} = \alpha_{uu'} \tau_{uu'}.$$

We can now obtain the original $\alpha_{uu'}$ by

$$\alpha_{uu'} = \frac{\alpha'_{uu'}}{\tau_{uu'}}.$$

5.4.1. Two estimators

We now describe in more detail the `tick.hawkes.ExpKern` and `tick.hawkes.HawkesADM4` estimators, which we refer to as the ADM4 and Expkern estimators. Estimating both the baseline intensity $\mathcal{M}$, excitation matrix $\mathcal{A}$, and characteristic time matrix $\mathcal{T}$ at once comes down to a non-convex optimization problem. Therefore, both estimators assume a given value for the characteristic time matrix $\mathcal{T}$ and subsequently, estimate the baseline intensity $\hat{\mathcal{M}}$ and excitation matrix $\hat{\mathcal{A}}$.

We describe the available M-estimators (log-likelihood or least-squares), available choices for setting excitation matrix $\mathcal{T}$, and the regularization method.

`tick.hawkes.ExpKern`
The Expkern estimator can estimate both the likelihood functional from Equation (5.40) as well as the least-squares functional from Equation (5.41) for parameter estimation. Given decay matrix $\mathcal{T}$, it returns estimated baseline intensity $\hat{\mathcal{M}}$ and estimated excitation matrix $\hat{\mathcal{A}}$ which optimize the chosen functional. To enforce sparsity for the estimated excitation matrix, four different regularization methods exist. These regularization methods penalize large values for the estimated excitation matrix $\hat{\mathcal{A}}$.

The four methods are ℓ_1-regularization, ℓ_2-regularization, a linear combination of ℓ_1 and ℓ_2 (called *elastic net*) and *nuclear*-regularization. We give a minimalistic overview of each of these regularization methods.

- ℓ_1-regularization, also called *Lasso* regularization, is defined as

$$\|\mathcal{A}\|_1 = \max_{1 \leq u' \leq U} \sum_{u=1}^{U} |\alpha_{uu'}|, \tag{5.42}$$

  which equals the maximum absolute column sum. In terms of excitation matrix values, where all terms are positive by definition, this is the maximum value one mark 'excites' on all other marks, i.e. the maximum column sum of the estimated excitation matrix.
- ℓ_2-regularization, also called *Ridge* regularization, is defined as

$$\|\mathcal{A}\|_2 = \sqrt{\lambda_{\max}\left(\mathcal{A}^T\mathcal{A}\right)} = \sigma_{\max}(\mathcal{A}), \tag{5.43}$$

  where λ_{max} is the highest eigenvalues of $\mathcal{A}^T\mathcal{A}$ and $\sigma_{\max}(\mathcal{A})$ represents the largest singular value of estimated excitation matrix $\hat{\mathcal{A}}$.
- *Elastic net*-regularization is a combination of ℓ_1-regularization and ℓ_2-regularization. It is characterized by parameter π governing the ratio between ℓ_1 and ℓ_2 regularization. The elastic net regularization therefore becomes

$$\|\mathcal{A}\|_{\pi=\cdot} = \pi \cdot \|\mathcal{A}\|_1 + (1-\pi) \cdot \|\mathcal{A}\|_2 \text{ for } \pi \in (0,1). \tag{5.44}$$

- *Nuclear*-regularization is defined as

$$\|\mathcal{A}\|_* = \sum_{i=1}^{\text{rank A}} \sigma_i(\mathcal{A}), \tag{5.45}$$

  which is the sum of singular values.

`tick.hawkes.HawkesADM4`
In contrast to the Expkern solver, the ADM4 solver is more restricted. The ADM4 estimator can only estimate the maximum log-likelihood and assumes one characteristic time value τ between the excitation of different marks. Therefore the excitation matrix equals

$$\mathcal{T} = \tau \cdot \mathbf{1}\mathbf{1}^T \text{ for } \mathbf{1} \in \mathbb{R}^U.$$

The choice for allowing only one characteristic time τ to be set, resulted in fast computation as compared to the ExpKern estimator. Moreover, in Section 6.3.1 we will compare the runtime of both the ADM4 and ExpKern estimator to validate this is indeed the case.

The ADM4 estimator uses a mix of ℓ_1-regularization from Equation (5.42) and Nuclear-regularization from Equation (5.45). It therefore maximizes the log-likelihood under regularization

$$\min_{\mathcal{A} \geq 0, \mathcal{M} \geq 0} -\mathcal{L}(\theta; \mathcal{H}_T) + \pi\|\mathcal{A}\|_* + (1-\pi))\|\mathcal{A}\|_1. \tag{5.46}$$

The ADM4 estimator uses an expectation minimization (EM) approach in order to maximize the log-likelihood under regularization. In particular, by combining ℓ_1 and Nuclear regularization, it enforces both sparsity and a low-rank structure.

Remark 5.4.1. We established consistency for the marked Hawkes process with an exponential memory kernel for the log-likelihood and least-squares estimates. These proofs were established for marked Hawkes processes without regularization. Only Guo et al. [25] considered ℓ_1-regularization of the estimated excitation matrix $\hat{\mathcal{A}}$.

Comparison between least squares and likelihood functionals

Bompaire [9] made a comparison between the least squares and negative log-likelihood functionals in terms of runtime complexity. Given point process $\mathbb{T}$ with a total of n arrivals for mark space $\mathcal{U} = \{1, \dots, U\}$. The complexity for the least squares estimate becomes $O(U^3)$ compared to a complexity of $O(n \cdot U)$ for the log-likelihood estimate. Remarkably, the complexity of the least-squares functional is independent of the number of arrivals and only depends on the number of dimensions of the mark space. The least squares functional is therefore preferred in cases where $n \gg U^2$. In the framework of large-scale service systems, this is the case when there are many semantically similar arrivals reoccurring.

Finally, in Section 6.3.1, we will evaluate the Expkern and ADM4 estimator for their default values and compare the results in terms of runtime and enforced sparsity.

6

Hierachical Hawkes processes

In this chapter, we unveil the outcomes of a novel hierarchical Hawkes model. We apply diverse hierarchical linear models to the estimated excitation matrix $\hat{A}$, adhering to the five-level hierarchical service architecture as can be found in Figure 4.4.

This chapter is structured into five sections. In Section 6.1, we initiate by constructing an illustrative example, featuring a simple two-level hierarchical model. Subsequently, we progressively extend this model, building up to a comprehensive five-level model in Section 6.2. Upon establishing the five-level model, Section 6.3 delves into three essential practical considerations necessary to estimate the excitation matrix. These considerations encompass (1) the selection of an estimator from the *tick* library, (2) the determination of the characteristic time τ, and (3) the conduction of a simulation study aimed at tackling the configuration item - business application mapping.

Moving forward to Section 6.4, we decompose the estimated excitation matrix into different levels of service, employing various hierarchical linear models. These models will be applied to the IT monitoring data stream from the same business unit examined in Chapter 4. Finally, in Section 6.5, we will interpret the findings from the previous section and discuss their implications for the field of Software Architecture.

This chapter builds around answering the final research question.

Research Question 4

How can the estimated excitation matrix contribute to understanding the associations within a level of service?

6.1. Two-level hierarchical model

Before decomposing the two-level hierarchical model, it is important to first introduce the concepts of fixed and random effects, as well as the concept of nesting.

Fixed and random effects

In the context of hierarchical linear models, fixed effects refer to the predictor or independent variables that are presumed to exert a systematic and unchanging impact on the dependent variable, which, in this chapter, corresponds to the estimated excitation value. The fixed effects are considered to be constant across different groups or clusters. Essentially, fixed effects are employed to characterize the average associations between the predictor variables and the outcome variable.

Conversely, random effects typically pertain to grouping factors that we aim to control for. Often, we are not specifically interested in their direct impact on the response variable. Additionally, the data for random effects is usually a sample from the entire set of possibilities. For instance, in the case of configuration items, we are not primarily concerned with the individual effect of each configuration item

on the excitation matrix. However, we acknowledge that the excitation values within a configuration item might be correlated, and therefore, we seek to control for this correlation as a random effect.

Explicit nesting
Nesting can be categorized into two types: explicit and implicit. In the context discussed here, levels two to five introduce implicit nesting. This implies incidents j on configuration item k are correlated to incidents j on another configuration item k'. Additionally, arrivals on the same configuration items k but different business applications l, l' are correlated.

On the other hand, nesting at the level of message clusters (level one) is explicit. The clusters within the message cluster hierarchy are only meaningful when associated with a specific business unit. This is due to the fact that the message clustering process was conducted independently for each business unit. Therefore, clusters of type 1 for one business unit m have no relation to clusters of type 1 for another business unit m'. Consequently, clusters of type 1 in one business unit m bear no connection to clusters of type 1 in another business unit m'. They represent distinct sets of semantically similar messages unique to their respective business units.

6.1.1. Two-level mark space

In Section 5.3, we established the excitation of mark u' on u as $\alpha_{uu'}$. However, from this point forward, we will incorporate the service levels outlined in Section 4.1.3 to illustrate how we can integrate the hierarchical service architecture into the excitation value. To begin with, we will use only two of the five service levels. This gradual approach allows us to introduce the concept of decomposing the excitation value into multiple service components before moving on to a more complex model that includes all five levels of service.

For the service levels of interest, we will focus on the message cluster $h \in CLU$ and the configuration item $k \in CI$. Since each arrival originates from a single cluster and a single configuration item, our mark space is now represented as $\mathcal{U} = CLU \times CI$. Consequently, we are interested in the excitation $\alpha_{uu'}$ of arrivals from mark u' (originating from cluster h' and configuration item k') on arrivals from mark u (originating from cluster h and configuration item k).

In the context of the two-level example, the level-1 units correspond to clusters, and the level-2 units correspond to configuration items. Instead of using $\alpha_{uu'}$, we represent the excitation as $\alpha_{hk\leftarrow h'k'}$, where h and h' belong to the set of clusters CLU, and k and k' belong to the set of configuration items CI.

Remark 6.1.1. In Section 4.1.3, it was shown that the set of clusters CLU^m depends on the business unit m. However, for the sake of illustration in this section, we will use the simplified notation CLU.

Definition 6.1.2: Two-level hierarchy

Let level-1 units be $h, h' \in CLU := \{1, \dots, n_{clu}\}$ and let level-2 units be $k, k' \in CI := \{1, \dots, n_{ci}\}$. The excitation of arrivals from mark u' (originating from cluster h' and configuration item k') on arrivals from mark u (originating from cluster h and configuration item k) can hierarchically be decomposed as

$$\begin{cases} \alpha_{hk\leftarrow h'k'} := \beta_{0k\leftarrow 0k'} + r_{hk\leftarrow h'k'} & \text{Level-1,} \\ \beta_{0k\leftarrow 0k'} = \gamma_{00} + u_{0k\leftarrow 0k'} & \text{Level-2.} \end{cases}$$

Remark 6.1.3. We use the notation "*arrivals* hk" to refer to all arrivals that originate from the combination of message cluster h and configuration item k.

Furthermore, we can represent the model in a combined form, leading to the equation

$$\alpha_{hk\leftarrow h'k'} := \gamma_{00} + u_{0k\leftarrow 0k'} + r_{hk\leftarrow h'k'}. \tag{6.1}$$

Now, we can provide interpretations for each of the coefficients:

- $\alpha_{hk\leftarrow h'k'}$ is the excitation of arrivals $h'k'$ on arrivals hk.
- $\beta_{0k\leftarrow 0k'}$ is the mean effect of arrivals at configuration item k' on arrivals at configuration item k. This is called the *level-1 intercept*.
- $r_{hk\leftarrow h'k'}$ is the deviation of the excitation of arrivals $h'k'$ on arrivals hk as compared to the mean effect $\beta_{0k\leftarrow 0k'}$. This is called the *level-1 random effect*. It is assumed to be normally distributed with mean zero and standard deviation σ^2.
- γ_{00} is the grand mean. This is called the *level-2 intercept*, which is a *fixed effect*.
- $u_{0k\leftarrow 0k'}$ is the deviation of the excitation of arrivals k' on arrivals k from the grand mean. This is called the *level-2 random effect*. It is assumed to be normally distributed with mean zero and standard deviation τ_β.

Hierarchical Hawkes

We can now substitute back the excitation $\alpha_{hk\leftarrow h'k'}$ within the framework of marked Hawkes process. In this thesis, we consider a marked Hawkes process with an exponential memory kernel, as seen in Definition 5.3.4. However, our analysis can be applied to a family of memory kernels $\phi_{uu'}(\cdot)$ for which the memory kernel can be split into an instantaneous excitation $\alpha_{uu'}$ and a decreasing function $\phi'_{uu'}(\cdot)$ governing the rate decay. Clearly, the exponential memory kernel satisfies this property, where $\phi'_{uu'}(t) = \exp\left(-\frac{t}{\tau_{uu'}}\right)$.

We can now define a hierarchical Hawkes model with the decomposition from Definition 6.1.2. We denote each arrival $i = 1, \dots, n$ as $(T_i, u_i) := (T_i, h_i, k_i)$. Here T_i is the arrival time and $u_i \in \mathcal{U} := \{1, \dots, U\}$ the mark of arrival i. The mark u_i consists of the message cluster $h_i \in CLU$ and the configuration item $k_i \in CI$.

Definition 6.1.4: Hierarchical Hawkes

The conditional intensity function for mark hk is defined as

$$\lambda_{hk}(t|\mathcal{H}_t) := \mu_{hk} + \sum_{k'\in CI} \sum_{h'\in CLU} \sum_{\substack{(T_i,h_i,k_i)\in\mathcal{H}_t \\ h_i=h' \\ k_i=k'}} \alpha_{hk\leftarrow h'k'} \phi'_{hk\leftarrow h'k'}(t - T_i),$$

with

$$\alpha_{hk\leftarrow h'k'} := \gamma_{00} + u_{0k\leftarrow 0k'} + r_{hk\leftarrow h'k'}.$$

Substituting the second line into the first results in

$$\lambda_{hk}(t|\mathcal{H}_t) := \mu_{hk} + \sum_{k'\in CI} \sum_{h'\in CLU} \sum_{\substack{(T_i,h_i,k_i)\in\mathcal{H}_t \\ h_i=h' \\ k_i=k'}} (\gamma_{00} + u_{0k\leftarrow 0k'} + r_{hk\leftarrow h'k'})\, \phi'_{hk\leftarrow h'k'}(t - T_i).$$

Remark 6.1.5. Hierarchical (linear) models are referred to as *multilevel models*, or *mixed-effect models* in related literature. The "mixed-effect" refers to a mix of fixed and random effects. The name differs among research disciplines. We adopt the name hierarchical model as it conveys the key characteristic that our model consists of multiple, hierarchical levels.

6.1.2. Estimation of parameters

With regards to the two-level hierarchical model from Definition 6.1.2,we aim to estimate two types of parameters. First of all, we are interested in the fixed effect γ_{00}. This fixed effect is the grand. Secondly, our interest lies in the random effects at level-1 and level-2, denoted as $r_{hk\leftarrow h'k'}$ and $u_{0k\leftarrow 0k'}$, respectively. We assume $r_{hk\leftarrow h'k'} \sim \mathcal{N}(0, \sigma^2)$ and $u_{0k\leftarrow 0k'} \sim \mathcal{N}(0, \tau_\beta)$.

Parameter estimation

To analyze the variance of the excitation value $\alpha_{hk\leftarrow h'k'}$, it can be decomposed into two components, each corresponding to one of the levels of the random effect. Thus, the variance of $\alpha_{hk\leftarrow h'k'}$ is expressed as follows:

$$\text{Var}\left(\alpha_{hk\leftarrow h'k'}\right) = \sigma^2 + \tau_\beta$$

For each of the estimated excitation values $\alpha_{hk\leftarrow h'k'}$ obtained from the estimated excitation matrix $\hat{\mathcal{A}}$, we can now calculate the 95% confidence interval as follows:

$$95\%ConfInt(\alpha_{hk\leftarrow h'k'}) = \hat{\alpha}_{hk\leftarrow h'k'} \pm 1.96 \cdot \text{Var}\left(\hat{\alpha}_{hk\leftarrow h'k'}\right).$$

The confidence interval is particularly important as it may help answer the question of whether one mark Granger causes the other. As established in Section 5.3.2, arrivals of mark $h'k'$ Granger cause arrivals of mark hk if and only if $\alpha_{hk\leftarrow h'k'} \neq 0$. To test this, we can use a hypothesis test for $H_0 := \alpha_{hk\leftarrow h'k'} = 0$ and check if zero falls within the 95% confidence interval $ConfInt(\alpha_{hk\leftarrow h'k'})$.

6.2. Five-level hierarchical model

In Section 6.1, we introduced an initial example to illustrate the two-level hierarchical Hawkes model. However, as detailed in Chapter 4, our hierarchical service architecture comprises five levels of service. Consequently, we can expand the two-level model into a five-level hierarchical model by incorporating the arrival nature, business application, and business unit levels. We can review the levels, starting from the top (level-5) and moving down to the bottom (level-1):

- Business unit $m \in BU := \{bu_1, \ldots, bu_{n_{bu}}\}$,
- Business application $l \in BA := \{ba_1, \ldots, ba_{n_{ba}}\}$,
- Configuration item $k \in CI := \{ci_1, \ldots ci_{n_{ci}}\}$,
- Arrival nature $j \in S := \{E, I\}$,
- Message cluster $h \in CLU^m := \{clu_1^m, \ldots, clu_{n_{clu}^m}^m\}$.

In a similar manner to Definition 6.1.2, we can create a five-level hierarchy centered around the excitation value $\alpha_{hjklm\leftarrow h'j'k'l'm'}$. However, it is important to note that by Definition 4.1.3, a business unit is described as an "independently operational setup". This suggests that arrivals in one business unit will never be associated with arrivals in another business unit. Therefore, when extending the two-level hierarchy from Definition 6.1.2 to a five-level hierarchy $\alpha_{ijklm\leftarrow h'j'k'l'm'}$, it is assumed that:

$$\alpha_{hjklm\leftarrow h'j'k'l'm'} = 0 \text{ for } m, m' \in BU : m \neq m'.$$

Therefore, we will exclusively focus on arrivals within a specific business unit of interest, denoted as $m \in BU$. The corresponding excitation within this business unit can be represented as $\alpha_{hjkl\leftarrow h'j'k'l;m}$. Instead, we will focus solely on the arrivals within the specific business unit of interest denoted as $m \in BU$. The corresponding excitation within this business unit can be represented as $\alpha_{hjkl\leftarrow h'j'k'l;m}$. Since this thesis does not involve comparing results between different business units $m, m' \in BU$, we can further simplify the notation. Thus, we denote the excitation as $\alpha_{hjkl\leftarrow h'j'k'l}$, with the understanding that the specific business unit under consideration is known to the reader. The mark space is now represented as $\boldsymbol{\mathcal{U}}^m = BA \times CI \times S \times CLU^m$.

In theory, this could potentially result in a very high number of marks, calculated as $\boldsymbol{Card}(\boldsymbol{\mathcal{U}}_m) = \boldsymbol{Card}(CLU) \cdot S \cdot \boldsymbol{Card}(CI) \cdot \boldsymbol{Card}(BA)$. However, due to the sparsity of our data, it is important to acknowledge that most of these potential marks are not observed.

Definition 6.2.1: Five-level hierarchy

Fix level-5 unit $m \in BU$. Let level-1 units be $h, h' \in CLU^m := \{clu_1^m, \ldots, clu_{n_{clu}^m}^m\}$, level-2 units be $j, j' \in S := \{E, I\}$, level-3 units be $k, k' \in CI := \{ci_1, \ldots ci_{n_{ci}}\}$ and level-4 units be $l, l' \in BA := \{ba_1, \ldots, ba_{n_{ba}}\}$.

The excitation of arrivals from mark u' (originating from cluster h', arrival nature j', configuration item k' and business application l') on arrivals from mark u (originating from cluster h, arrival nature j, configuration item k and business application l) can hierarchically be decomposed as

$$\begin{cases} \alpha_{hjkl \leftarrow h'j'k'l} = \pi_{0jkl \leftarrow 0j'k'l'} + e_{hjkl \leftarrow h'j'k'l'} & \text{Level-1,} \\ \pi_{0jkl \leftarrow 0j'k'l'} = \kappa_{00kl \leftarrow 00k'l'} + r_{0jkl \leftarrow 0j'k'l'} & \text{Level-2,} \\ \kappa_{00kl \leftarrow 00k'l'} = \beta_{000l \leftarrow 000l'} + z_{00kl \leftarrow 00k'l'} & \text{Level-3,} \\ \beta_{000l \leftarrow 000l'} = \gamma_{0000} + u_{000l \leftarrow 000l'} & \text{Level-4.} \end{cases}$$

Remark 6.2.2. We use the notation "*arrivals $hjkl$*" to refer to all arrivals that originate from the combination of message cluster h of nature j, occurring at configuration item k, and business application l.

Now, we can provide interpretations for each of the coefficients:

- $\alpha_{hjkl \leftarrow i'j'k'l'}$ is the excitation of arrivals $hjkl$ on arrivals $h'j'k'l'$.
- $\pi_{0jkl \leftarrow 0j'k'l'}$ is the mean effect mean effect of arrivals jkl on arrivals jkl. This is called the *level-1 intercept*.
- $e_{hjkl \leftarrow h'j'k'l'}$ is the deviation of the excitation of arrivals $hjkl$ on arrivals $h'j'k'l'$ as compared to the mean effect $\pi_{0jkl \leftarrow 0j'k'l'}$. This is called the *level-1 random effect*. It is assumed to be normally distributed with mean zero and standard deviation σ^2.
- $\kappa_{00kl \leftarrow 00k'l'}$ is the mean effect mean effect of arrivals kl on arrivals kl. This is called the *level-2 intercept*.
- $r_{0jkl \leftarrow 0j'k'l'}$ is the deviation of the excitation of arrivals jkl on arrivals $j'k'l'$ as compared to the mean effect $\kappa_{00kl \leftarrow 00k'l'}$. This is called the *level-2 random effect*. It is assumed to be normally distributed with mean zero and standard deviation τ_π.
- $\beta_{000l \leftarrow 000l'}$ is the mean effect mean effect of arrivals l on arrivals l. This is called the *level-3 intercept*.
- $z_{00kl \leftarrow 00k'l'}$ is the deviation of the excitation of arrivals kl on arrivals $k'l'$ as compared to the mean effect $\beta_{000l \leftarrow 000l'}$. This is called the *level-3 random effect*. It is assumed to be normally distributed with mean zero and standard deviation τ_κ.
- γ_{0000} is the grand mean. This is called the *level-4 intercept*, which is a *fixed effect*.
- $u_{000l \leftarrow 000l'}$ is the deviation of the excitation of l on arrivals l' as compared to the grand mean γ_{0000}. This is called the *level-4 random effect*. It is assumed to be normally distributed with mean zero and standard deviation τ_β.

As the random effects are assumed to follow a normal distribution, there's a possibility of negative excitation values $\alpha_{ijkl \leftarrow i'j'k'l'}$. However, excitation values, by definition, should be non-negative. To overcome this issue, we apply a base-ten logarithm transformation to the excitation values, resulting in $\log_{10}(\alpha_{hjkl \leftarrow h'j'k'l'})$. Instead we therefore model $\log_{10}(\alpha_{hjkl \leftarrow h'j'k'l'})$. Additionally, for the estimated excitation values $\hat{\alpha}_{ijkl \leftarrow i'j'k'l'}$ for which $\hat{\alpha}_{ijkl \leftarrow i'j'k'l'} = 0$, we substitute them with $\hat{\alpha}_{ijkl \leftarrow i'j'k'l'} = 1e - 20$. This substitution has a notable consequence. Due to the substitution, we can no longer generate two streams of independent data. In other words, every mark now Granger causes every other mark because zero-valued excitation values have been replaced by a positive value.

6.2.1. Assumptions

It is worth noting that the current model makes certain assumptions, which include:

1. All of the random effects $e_{hjkl \leftarrow h'j'k'l'}$, $r_{0jkl \leftarrow 0j'k'l'}$, $z_{00kl \leftarrow 00k'l'}$, $u_{000l \leftarrow 000l'}$ are assumed to be normally distributed with mean zero.

2. All the random effects are assumed to be independent. Therefore all possible covariance terms among $e_{hjkl\leftarrow h'j'k'l'}$, $r_{0jkl\leftarrow 0j'k'l'}$, $z_{00kl\leftarrow 00k'l'}$ and $u_{000l\leftarrow 000l'}$ are considered to be zero.
3. *Homogeneity of variance per level;* The variance structure within each level is assumed to be consistent across all components at that level. In other words, there is no variation in the variance structure among items within the same level.

6.2.2. Variance partitioning

Due to the homogeneity of variance per level, the total variability of the estimated log-excitation $\log_{10}(\hat{\alpha}_{hjkl\leftarrow h'j'k'l'})$ can be partitioned into four components:

- (level 1) among message clusters within a cluster type, $\sigma^2 = var(e_{hjkl\leftarrow h'j'k'l'})$;
- (level 2) among cluster types within a configuration item, $\tau_\pi = var(r_{0jkl\leftarrow 0j'k'l'})$;
- (level 3) among configuration items within a business application, $\tau_\kappa = var(z_{00kl\leftarrow 00k'l'})$.
- (level 4) among business applications, $\tau_\beta = var(u_{000l\leftarrow 000l'})$.

This allows us to calculate the proportion of variation for each of the four levels: $\frac{\sigma^2}{\sigma^2+\tau_\pi+\tau_\kappa+\tau_\beta}$, $\frac{\tau_\pi}{\sigma^2+\tau_\pi+\tau_\kappa+\tau_\beta}$, $\frac{\tau_\kappa}{\sigma^2+\tau_\pi+\tau_\kappa+\tau_\beta}$ and $\frac{\tau_\beta}{\sigma^2+\tau_\pi+\tau_\kappa+\tau_\beta}$. Finally, in Section 6.4, we will partition each of the variance components and provide interpretations of the results.

6.3. Practical considerations

In this section, we will evaluate three practical aspects that need to be addressed in order to estimate the excitation matrix. Firstly, we will determine the appropriate estimator to use from the *Tick* library, specifically comparing the `ExpKernel` and `ADM4` estimators. We will analyze their differences in terms of runtime performance and sparsity. Secondly, we will elaborate on the selection of the characteristic time τ, conducting a sensitivity analysis within a limited range of characteristic times and explaining our final choice. Lastly, we will conduct a simulation study to examine the impact of spurious connections between configuration items and business applications.

For Section 6.3.1 and 6.3.2, we use the message cluster setting as established in Section 4.3. In each the the two sections we fixed the other parameters and only inspected one practical consideration at a time. Although it would be interesting to simultaneously investigate the influence of multiple practical considerations, this is not feasible due to runtime constraints. Combining all clustering settings from Section 4.3 with different *Tick* solvers and a wide range of characteristic times would result in months of runtime.

6.3.1. Choice of estimator

In Section 5.4, we introduced two estimators: the ADM4 estimator, which incorporates a combination of ℓ_1 and Nuclear regularization for the regularized log-likelihood functional, and the *ExpKern* estimator, which offers the flexibility to customize regularization functions for both the log-likelihood and the least-squares functional.

We conducted a comparative analysis between the ADM4 and ExpKern estimators in terms of runtime and their ability to impose the desired level of sparsity. We ran both estimators with their default settings and also ran an estimation with the ExpKern estimator using a log-likelihood functional instead of the default least-squares functional. In all three scenarios, we maintained a fixed characteristic time of $\tau = 10,000\ s$.

In our initial attempts, we tested all three scenarios as described above. However, we encountered convergence issues with both scenarios for the ExpKern estimator (log-likelihood and least-squares). We suspected that the high dimensionality of the mark space was the cause of the convergence problems. To address this, we reduced the mark space by increasing the minimum cluster size for the HDBSCAN clustering algorithm. Specifically, we set the minimum cluster size to 6 (instead of 4), for both event and incident clusters. This reduction decreased the mark space by a factor two. Despite the adjustment made to the mark space, the ExpKern estimator using the log-likelihood functional

still did not converge. However, the ExpKern estimator using the default least squares function was able to converge successfully. The results for the estimated excitation matrix $\hat{\mathcal{A}}$ for both the AMD4 estimator and the ExpKern estimator using the default least squares functional are visualized in Figure 6.1.

The comparison between the two estimators revealed substantial differences in terms of runtime and sparsity enforcement. Specifically, it took the ADM4 estimator approximately 195 seconds to converge, while the ExpKern estimator required a much longer time of 6436 seconds (more than 1.5 hours) to converge. Additionally, as observed in Figure 6.1, the ExpKern estimator did not effectively enforce sparsity. Given the substantial runtime and the lack of sparsity enforcement with the ExpKern estimator, the ADM4 estimator will be the primary choice for further analysis.

Remark 6.3.1. It is worth noting that we have the option to adjust the regularization parameters for the ExpKern estimator. However, given that the ADM4 estimator has already demonstrated satisfactory performance in terms of runtime and sparsity enforcement, there was no need to explore further adjustments at this stage.

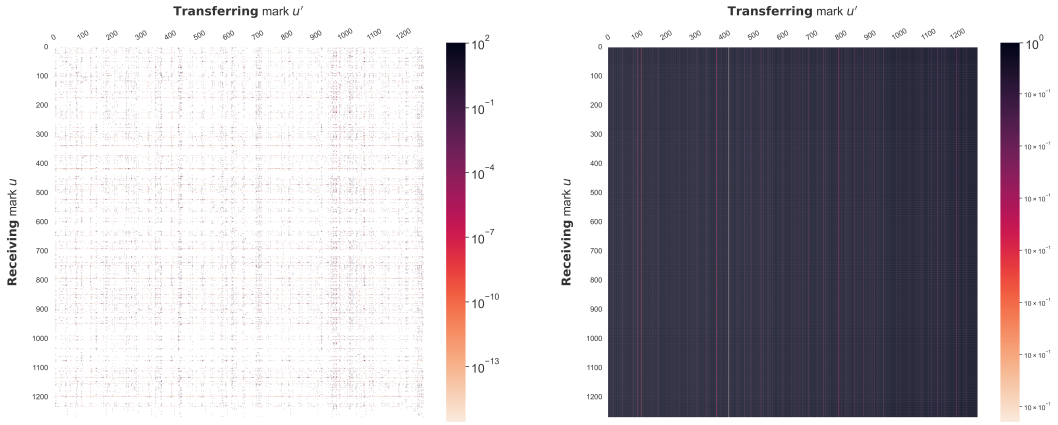

Figure 6.1: Estimated excitation matrix $\hat{\mathcal{A}}$ for the default settings of the ADM4 estimator (left), and Expkern estimator with least squares functional (right). Both estimators used characteristic time $\tau = 10.000s$.

6.3.2. Characteristic time τ

As outlined in Definition 5.3.4, we revisit the exponential memory kernel denoted as

$$\phi_{uu'}(t) = \alpha_{uu'} \cdot \exp\left(-\frac{t}{\tau_{uu'}}\right).$$

However, estimating the excitation matrix $\hat{\mathcal{A}}$ (of values $\alpha_{uu'} \in \mathbb{R}^{U \times U}$) and characteristic time matrix $\hat{\mathcal{T}}$ (of values $\tau_{uu'} \in \mathbb{R}^{U \times U}$) comes down to a non-convex optimization problem. Hence, the current practice is to select a value for τ that accurately reflects the specific context of interest. Secondly, we calculate $\mathcal{T}$ by taking the product of τ and the identity matrix $\mathbf{1}\mathbf{1}^T$, yielding $\mathcal{T} = \tau \cdot \mathbf{1}\mathbf{1}^T$. After setting a value for $\mathcal{T}$, we proceed to estimate the excitation matrix $\hat{\mathcal{A}}$

A prior baseline model designed by system engineers at ING assumed arrivals are equally likely to excite subsequent arrivals for a time period of 4 hours. After this time period, an arrival is assumed not to excite subsequent arrivals. This poses us with a trade-off for setting a value for τ. Setting a

high value for τ corresponds with the intuition that it is likely for arrivals to result in an increase in probability for subsequent arrivals over the course of the entire 4-hour interval after the initial arrival, but fails to incorporate that it is unlikely for arrivals to excite subsequent arrivals after the 4-hour interval. Alternatively, a small τ value mimics the setting where it is unlikely to excite subsequent arrivals after 4 hours but it may not adequately capture the high excitation during the initial 4-hour period.

To understand how the characteristic time relates to the baseline model, we introduce the concept of the *vanishing time*, represented as ξ. The vanishing time signifies the duration after which only 1% of the initial excitation α remains.

Definition 6.3.2: Vanishing time

Given exponential memory kernel $\phi(t) = \alpha \cdot \exp\left(-\frac{t}{\tau}\right)$ with fixed parameters $\alpha \in \mathbb{R}_+, \tau \in \mathbb{R}_+^*$. The vanishing time ξ is defined such that

$$\phi(\xi) = 0.01 \cdot \alpha.$$

The relation between vanishing time and characteristic time equals

$$\tau = \frac{\xi}{\log 100}.$$

For example, a vanishing time of $\xi = 24$ hours results in

$$\tau = \frac{24}{\log 100} \approx 5.2 \text{ hours} \approx 18.700 \text{ seconds}.$$

So after roughly 5.2 hours, the initial excitation is reduced by a factor e. The influence of different vanishing times on the original excitation, together with the baseline model, is displayed in Figure 6.2.

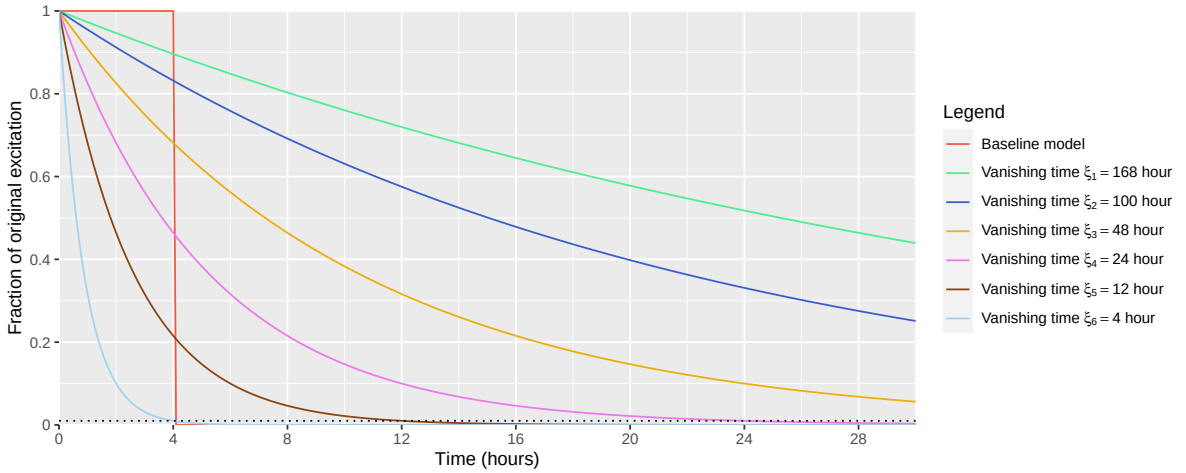

Figure 6.2: Fraction of original excitation over time for six values of vanishing time ξ. together with ING's baseline model.

Sensitivity analysis on the influence of characteristic time τ

We perform a sensitivity analysis on the same business unit that was examined in Section 4.3. We employ the optimal clustering settings as determined in Section 4.3 and use the ADM4 estimator with default regularization settings. It is worth noting that the ADM4 estimator allows for the specification of a single characteristic time denoted as τ, which is used to create the matrix $\mathcal{T} = \tau \cdot \mathbf{1}\mathbf{1}^T$. This limitation in characteristic time does not affect our investigation, since introducing distinct characteristic times $\tau_{uu'}$ between each pair of marks would introduce excessive complexity, which is not within the scope of our analysis.

We run the ADM4 estimator for vanishing times ranging from 2 to 96 hours (4 days) with a 2-hour time difference. This results in

$$\xi \in [2, 4, 6, \ldots, 94, 96].$$

We compute the maximum likelihood and runtime for each of the vanishing times.

It is worth noting that we encountered an interesting phenomenon during this sensitivity analysis. For vanishing times up to 22 hours, the ADM4 solver failed to converge to a maximum likelihood solution. This failure was attributed to the singular value decomposition's inability to converge. The intuition behind this is that the excitation decays too rapidly to effectively capture relationships between marks when the vanishing time is too short. The first vanishing time for which a maximum likelihood solution was found was 24 hours. However, even for this and subsequent vanishing times, the maximum likelihood values exhibited a high degree of instability, as illustrated in Figure 6.3. This instability highlights the non-convex nature of the optimization problem involving the characteristic time and the excitation. Finally, the runtime for each of the vanishing times was approximately 10 minutes, with a negligible variation of only a few seconds [1].

In conclusion, we settled on a vanishing time of 24 hours, which corresponds to a characteristic time of approximately 18,700 seconds. This choice was made as it allowed us to obtain a likelihood, and it resulted in the highest likelihood among the tested vanishing times. To provide a point of reference, we compared our chosen characteristic time of approximately 18,700 seconds to the baseline model. Using this characteristic time, we found that approximately 46% of the excitation remained after the 4-hour cutoff point specified in the baseline model. This result can also be found in Figure 6.3.

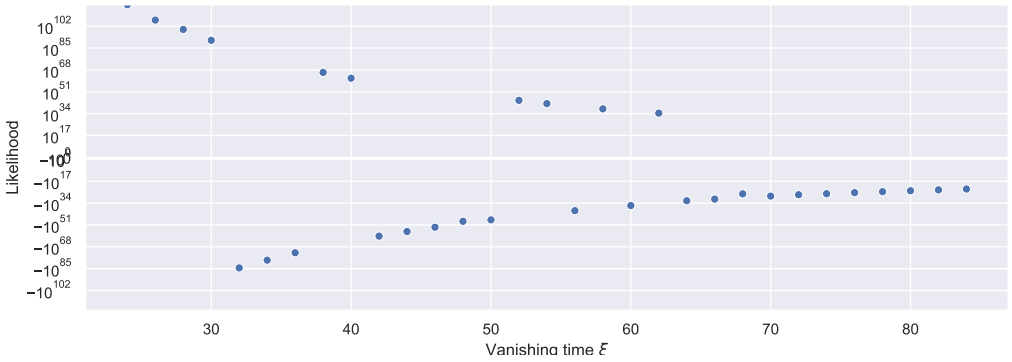

Figure 6.3: Likelihood for a range of vanishing times.

Remark 6.3.3. Additionally, we conducted six simulations with large vanishing times, ranging from 5 to 10 days with a 1-day difference. However, all of these vanishing times resulted in low maximum likelihood values. Therefore, we did not include them in the analysis.

6.3.3. Resolution for configuration item - business application mapping

In Section 4.1.2, we discussed the convoluted relation between the level-3 configuration items and the level-4 business applications. We have seen that arrivals registered on business application l, can be assigned to configuration item `UNKNOWN` for business application l. However, when an arrival is registered on a configuration item associated with multiple business applications, a straightforward solution is not readily available.

Whenever dealing with a registered configuration item that is linked to multiple business applications, we have identified four possible approaches for resolving the mapping. In the upcoming section, we will delve into a comprehensive discussion of these four options. To determine the most suitable resolution method for our framework, we will conduct a brief simulation study.

[1]The simulation was run on Intel Core I7-10850H processor, where 4 cores were used.

Remark 6.3.4. The practical considerations of choosing the estimator (Section 6.3.1) and determining the characteristic time τ (Section 6.3.2) explicitly relate to the estimation of the excitation matrix $\hat{\mathcal{A}} \in \mathbb{R}^{U \times U}$. Conversely, the practical consideration regarding the resolution of the configuration item - business application relates to the construction of the mark space $\mathcal{U} := 1, \ldots, U$ and the determination of which mark $u \in \mathcal{U}$ a specific arrival is assigned to.

Options for resolution

1. Drop the arrival and exclude it from further analysis.
2. Duplicate the arrival for each linked business application, resulting in multiple marks assigned to the arrival.
3. Randomly assign the arrival to one of the linked business applications.
4. Create a new synthetic business application labeled `business application group {BA set}`, where `{BA set}` encompasses all the linked business applications.

Certainly, the first two options result in bias. Even more so, the first option leads to loss of information, as we discard observed arrivals. The second option, where the arrival is associated with multiple marks, might inadvertently establish a misleading Granger causal relationship among the assigned marks. In contrast, randomly assigning the arrival to one of the business applications could be one of the options to reduce this bias. Finally, creating a new, synthetic `business application group` might especially be useful when numerous distinct configuration items are linked to the same set of business applications. With the fourth resolution option in place, arrivals on these distinct configuration items are therefore assigned to the same, synthetic `business application group`. The drawback of this method is that it might become difficult to interpret the results, as we will obtain results regarding the interaction between individual business applications and `business application group`'s.

Simulation setting

In this simulation study, we make the assumption that the mark is solely composed of the configuration item and business application. Furthermore, we consider a service architecture consisting of five configuration items and three business applications. The corresponding mapping is illustrated in Figure 6.4. Specifically, the configuration item - business application relationship includes three scenarios:

- One configuration item is linked to one business application.
- One configuration item is linked to two distinct business applications.
- Three configuration items are linked to the same business application.

In particular, the simulation study is performed in the framework of the two-level hierarchical model as constructed in Section 6.1. We capture the relation between configuration items (resp. business applications) through means of matrix $\mathcal{E}$ (resp. matrix $\mathcal{K}$). Each of the excitation values is now calculated as:

$$\alpha_{kl \leftarrow k'l'} := \kappa_{0l \leftarrow 0l'} + e_{kl \leftarrow k'l'} \text{ for } k, k' \in CI, l, l' \in BA,$$

for $\kappa_{0l \leftarrow 0l'} \in \mathcal{K}$ and $e_{kl \leftarrow k'l'} \in \mathcal{E}$.
We assume that the influence between the three configuration items is as displayed in $\mathcal{K}$,

$$\mathcal{K} = \begin{bmatrix} 10 & 2 & 0 \\ 0 & 8 & 1 \\ 1.5 & 1 & 15 \end{bmatrix}.$$

The interplay between business applications is encapsulated by matrix $\mathcal{E}$. We set the diagonal elements as $e_{k,k'} = 10$ for $k = k'$, while the off-diagonal elements are sampled from a uniform distribution [2] $e_{k,k'} = \mathcal{U}[0, 2]$ for $k \neq k'$. We sample $\mathcal{E}$ only once and keep the value fixed throughout the rest of the simulation. This results in

[2] $\mathcal{U}$ should not be confused with the mark space. Here, and only here, $\mathcal{U}$ indicates the uniform distribution.

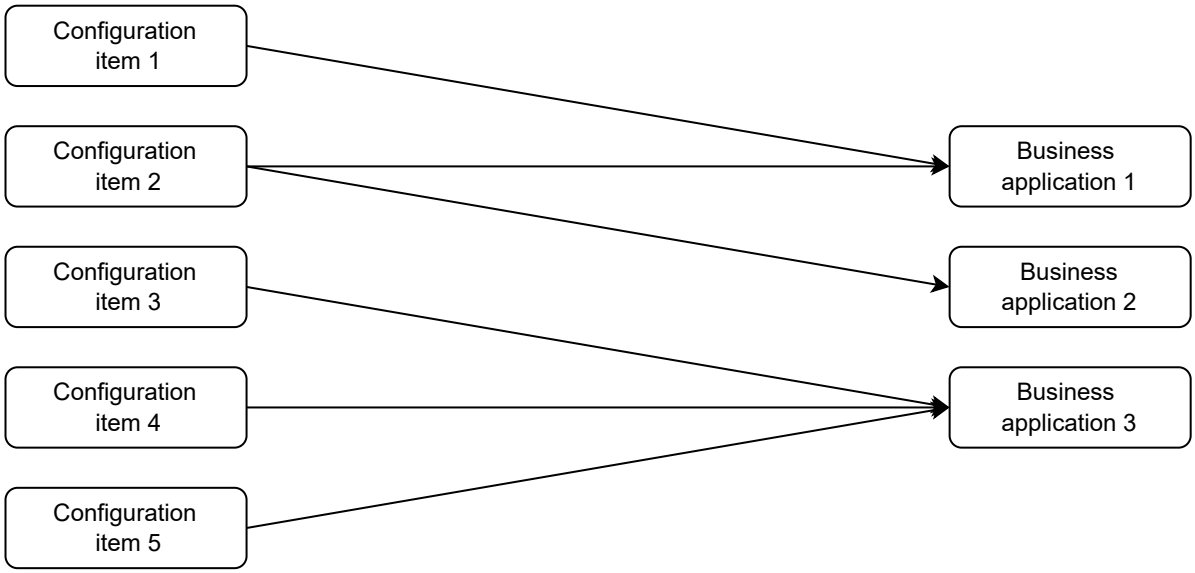

Figure 6.4: Simulation study configuration item - business application mapping for five configuration items and three business applications.

$$\mathcal{E} = \begin{bmatrix} 10 & 0.042 & 1.3 & 1.5 & 1.0 \\ 0.45 & 10 & 1.5 & 0.34 & 0.18 \\ 1.4 & 1.9 & 10 & 1.05 & 1.6 \\ 1.2 & 1.4 & 0.58 & 10 & 1.4 \\ 1.1 & 0.28 & 0.75 & 1.3 & 10 \end{bmatrix}.$$

Finally, the excitation matrix was divided by a factor 100.

$$\mathcal{A} = \frac{1}{100} \begin{bmatrix} 20 & 10 & 0.042 & 2.8 & 3.0 & 2.5 \\ 10 & 20 & 10 & 3.0 & 1.8 & 1.7 \\ 2.4 & 12 & 18 & 1.5 & 0.34 & 0.18 \\ 1.4 & 1.9 & 2.9 & 25 & 16 & 17 \\ 1.2 & 1.4 & 2.4 & 16 & 25 & 16 \\ 1.1 & 0.28 & 1.3 & 16 & 16 & 25 \end{bmatrix}.$$

The baseline intensity and characteristic time matrix are, respectively, defined as

$$\mathcal{M} = 0.03 \cdot \mathbf{1}^T \text{ for } \mathbf{1} \in \mathbb{R}^6,$$
$$\mathcal{T} = \tau \cdot \mathbf{1}\mathbf{1}^T = 3.0 \cdot \mathbf{1}\mathbf{1}^T,$$

for $\mathbf{1} \in \mathbb{R}^6$. The choices for $\mathcal{A}$, $\mathcal{M}$ and $\mathcal{T}$, as well as dividing the excitation matrix by 100, were made such that the simulation roughly represented the excitation behavior as estimated from the IT monitoring data stream.

Simulation procedure

In order to perform the simulation study for each of the four resolution options, we perform Algorithm 2. In particular, we repeat 1000 experiments and obtain the mean and standard deviation for each of the resolution options. Additionally, we run a benchmark. This benchmark estimates the excitation matrix from the original simulated data $\mathbb{T}_{MPP}$ instead of the latent data $\hat{\mathbb{T}}_{MPP}$. By comparing the results of the estimated benchmark excitation matrix and the true excitation matrix, we obtain a feel for the natural deviation of the estimator. Additionally, we set the probability of being assigned to the business application (resp. configuration item) to 20% (resp. 80%). This ratio was selected to mirror the proportion of arrivals registered on a business application as observed from the IT monitoring data stream at ING.

Algorithm 2 Simulation study for four resolution options

Establish mark setting (see Figure 6.4),
Fix true parameters $\mathcal{A}$, $\mathcal{M}$ and $\mathcal{T}$,
for $i = 1, \dots, 1000$ **do**
  Sample timestamps $\mathbb{T}_i$ using $\mathcal{A}$ and $\mathcal{T}$,
  Create $\hat{\mathbb{T}}_i$ by randomly assigning 20% of arrivals to the business application and 80% of the arrivals to the configuration item,
  for Resolvement method j **do**
    Estimate excitation matrix $\hat{\mathcal{A}}_{i,j}$ using $\mathcal{T}$, $\hat{\mathbb{T}}_i$, and resolution option j,
  end for
  Estimate excitation matrix $\hat{\mathcal{A}}_{i,baseline}$ using $\mathcal{T}$ and $\hat{T}_i$,
end for
for Resolvement method j and baseline **do**
  Compute mean and standard deviation for $\left[err\left(\hat{\mathcal{A}}_{1,j}\right), err\left(\hat{\mathcal{A}}_{2,j}\right), \dots, err\left(\hat{\mathcal{A}}_{1000,j}\right)\right]$.
end for

Remark 6.3.5. Using $\mathcal{T}$ and $\mathbb{T}_{MPP}$, we can estimate $\hat{\mathcal{M}}$ and compare it to $\mathcal{M}$. However, since our primary concern is the impact of the configuration item - business application mapping on the estimated excitation matrix, we will not delve into interpreting the results regarding the baseline intensity.

In order to calculate the error between the estimated excitation matrix $\hat{\mathcal{A}}$ and the true excitation matrix $\mathcal{A}$, we calculate the following error metric

$$err(\hat{\mathcal{A}}) = \frac{\|\hat{\mathcal{A}} - \mathcal{A}\|_F}{\|\mathcal{A}\|_F}, \tag{6.2}$$

for each of the 1000 runs. Once we obtain the 1000-dimensional error vector for each of the resolution options, we can compute its mean and standard deviation.

For the last option of creating a business application group, we create a new mark, namely `{Configuration item 2, Business application group 1}`. This can be found in Figure 6.5. Therefore $\hat{\mathcal{A}} \in \mathbb{R}^7$ whereas $\mathcal{A} \in \mathbb{R}^6$. As the dimensions do not correspond, we can not directly compare the two matrices. In order to resolve this issue, we only look at the submatrix $\hat{\mathcal{A}}_{sub}$, where we drop the column and row with regards to mark between {configuration item 2, `business application group {1,2}`}.

Simulation results

The results from the simulation study as performed in Algorithm 2 are depicted in Table 6.1. The second row displays the increase in mean error as compared to the unmodified benchmark scenario.
It was found that indeed the duplication scenario resulted in the highest mean error. Interestingly enough, dropping arrivals results in a smaller mean error as compared to randomly assigning the arrival to a business application.

Table 6.1: Results for the mean and standard deviation for $err(\hat{\mathcal{A}})$ for four recovering methods as well as an unmodified benchmark. The second row shows the increase as compared to the mean benchmark error.

	Benchmark	**Drop**	**Duplicate**	**Randomly assign**	**Create `BA group`**
Mean of $err(\hat{\mathcal{A}})$	0.36	0.44	0.52	0.47	0.46
Increase from benchmark		22%	44%	31%	28%
Sd of $err(\hat{\mathcal{A}})$	0.073	0.084	0.069	0.10	0.080

Establishing on resolution choice

Considering the outcomes presented in Table 6.1, the most straightforward choice would be to drop arrivals associated with configuration items linked to multiple business applications. However, there

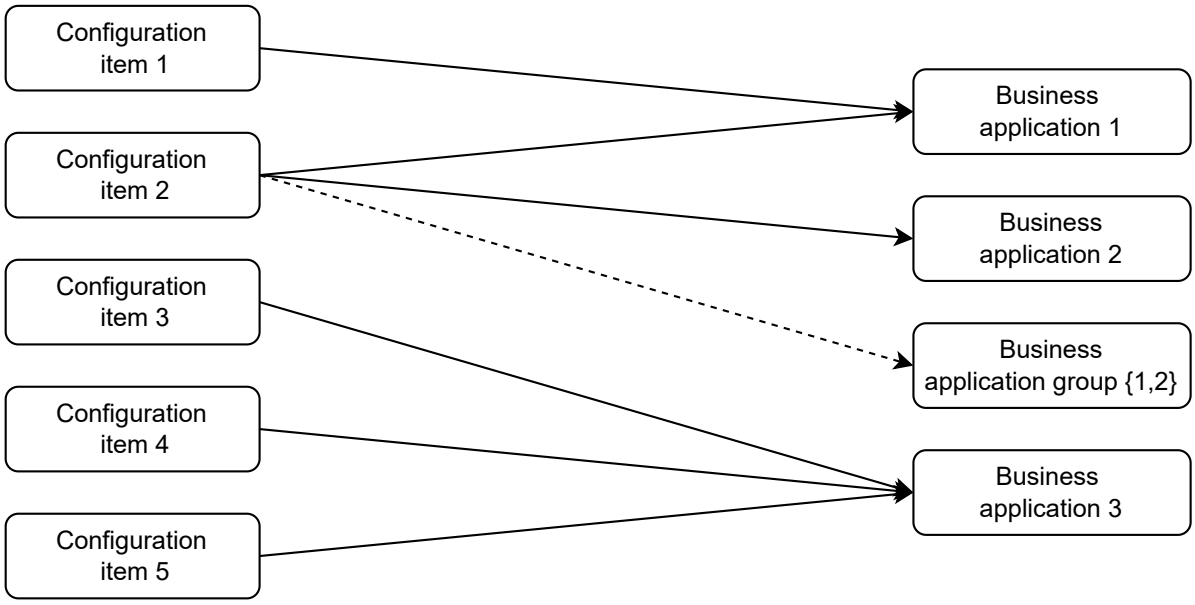

Figure 6.5: Simulation study configuration item - business application mapping for five configuration items, three business applications, and one artificial business application group.

exists an unexamined advantage associated with the choice of crafting a new business application group. In scenarios where two separate configuration items are linked to an identical set of business applications, they will consequently be allocated to the same synthetic `business application group {·}`. This feature permits arrivals registered on distinct configuration items to be aggregated under the same (synthetic) business application.

Authors note: This paragraph is removed due to confidentiality.

Table 6.2: Number of configuration items linked to the same `Business application group{·}`

Authors note: This table is removed due to confidentiality.

6.4. Main results: Hierarchical linear model

In Section 6.3, we established all practical considerations in order to obtain the estimated excitation matrix $\hat{\mathcal{A}}$. This allows for following the procedure in Algorithm 1 from Section 1.6 to obtain the estimated excitation matrix $\hat{\mathcal{A}}$ for one business unit $m \in BU$. Furthermore, in Sections 6.1 and 6.2 we established how each of the elements $\hat{\alpha}_{hjkl \leftarrow h'j'k'l'} \in \hat{\mathcal{A}}$ can be decomposed into different levels of service using a hierarchical linear model.

We now evaluate the results from the estimated excitation matrix, focusing on the business unit discussed in Section 4.3. We recall the estimated excitation matrix was obtained by using the optimal clustering section from 4.3, using the ADM4 estimator with a characteristic time of $\tau \approx 18.700\ s$ (i.e. a vanishing time $\xi = 24$ hours) and a configuration item - business application resolution of creating `business application group`'s

In order to evaluate the hierarchical model, we have transformed the estimated excitation matrix $\hat{\mathcal{A}} \in \mathbb{R}^{U \times U}$ to a *long* format. This long format creates a table that is U^2 rows by 9 columns, where each row represents a combination of service level variables $h, j, k, l, h', j', k', l'$ along with the corresponding estimated excitation value $\hat{\alpha} hjkl \leftarrow h'j'k'l'$. An example of the long format table showing the highest estimated excitation values $\hat{\alpha}_{hjkl \leftarrow h'j'k'l'}$ can be found in Table 6.9.

Establishing on simpler models
Rather than modeling the complete five-level model as defined in Definition 6.2.1, we will explore a set of simpler linear hierarchical models and analyze their outcomes. The rationale behind selecting these simpler models is to reduce the complexity of the estimated components and enhance interpretability. Additionally, as outlined in Section 4.1.3, we model the estimated excitation behavior on a logarithmic scale with base 10, and we substitute zero-excitation values with 1e-20.

We distinguish between the *transferring* excitation level, represented by levels h', j', k', and l', and the *receiving* excitation level, represented by levels h, j, k, and l, for each excitation value $\alpha_{hjkl \leftarrow h'j'k'l'}$.

6.4.1. Model 1: Business application

In order to obtain an overview of the excitation behavior between business applications, we consider the following model. This model only accounts for the transferring and receiving business applications l' and l, respectively.

$$\log_{10}(\alpha_{hjkl \leftarrow h'j'k'l'}) = \beta_{000l \leftarrow 000l'} + \epsilon_{hjkl \leftarrow h'j'k'l'},$$
$$\beta_{000l \leftarrow 000l'} = \gamma_{0000} + u_{000l \leftarrow 000l'},$$

where,

$$\epsilon_{hjkl \leftarrow h'j'k'l'} \sim \mathcal{N}(0, \sigma^2),$$
$$u_{000l \leftarrow 000l'} \sim \mathcal{N}(0, \tau_\beta).$$

It can be found from Table 6.3 that the business application, in this simple model, already explains over 29% of the total variation. To compare how similar the transferring excitation behavior between business applications is, we can compare the Euclidean distance between $\beta_{000l \leftarrow 000l'}$ for different values of $l' \in BA$. The results for this can be found in Figure 6.6. Here, we have applied hierarchical clustering and created a dendrogram using complete linkage. Note how the behavior of the synthetically created `business application group`'s does not differ from the behavior of genuine business applications. Additionally, we provide a more in-depth analysis of Figure 6.6 in Section 6.5.1.

Table 6.3: Model 1: a two-level model using only the business application.

	Variance	**Perc. of total variance**	**Standard deviation**
$\hat{\tau}_\beta$	3.56 e-3	29.1 %	5.97 e-2
$\hat{\sigma}^2$	8.65 e-3		9.30 e-2

	Estimate	**Std. Error**	**t-value**
$\hat{\gamma}_{0000}$	4.18 e-3	9.67 e-4	4.32

Authors note: This figure is removed due to confidentiality.

Figure 6.6: Logarithmically scaled heatmap of the excitation effect between different business applications. Additionally, a dendrogram showing the results of complete linkage using an Euclidean distance is visualized. Furthermore, seven block regarding the transferring business application are created.

Remark 6.4.1. Moreover, we can cluster the receiving business applications $l \in BA$. This resulted in similarly behaving *chunks* of business applications. The results can be found in Figure C.4.

6.4.2. Model 2: Configuration item

Similarly as in Section 6.4.1, we can obtain an overview of the behavior between configuration items. We therefore consider the following model:

$$\log_{10}(\alpha_{hjkl\leftarrow h'j'k'l'}) = \kappa_{00k0\leftarrow 00k'0} + \epsilon_{hjkl\leftarrow h'j'k'l'},$$
$$\kappa_{00k0\leftarrow 00k'l} = \gamma_{0000} + u_{00k0\leftarrow 00k0},$$
where,
$$\epsilon_{hjkl\leftarrow h'j'k'l'} \sim \mathcal{N}(0, \sigma^2),$$
$$u_{00k0\leftarrow 00k'0} \sim \mathcal{N}(0, \tau_\kappa).$$

It can be found from Table 6.4 that the configuration item explains for over 31% of the total variance. We once again use hierarchical clustering, however now we consider $\kappa_{00k0\leftarrow 00k'0}$ and cluster similar behaving configuration item blocks. The results can be found in Figure 6.7. Interestingly, it can be found that the configuration item `UNKNOWN` accounts for distinct behavior as compared to the genuine configuration items. Furthermore, its excitation behavior seems to be rather low. Additionally, we provide a more in-depth analysis of Figure 6.7 in Section 6.5.2.

Table 6.4: Model 2: a two-level model using only configuration item.

	Variance	**Perc. of total variance**	**Standard deviation**
$\hat{\tau}_\kappa$	4.04 e-3	31.8 %	6.35 e-2
$\hat{\sigma}^2$	8.66 e-3		9.31 e-2

	Estimate	**Std. Error**	**t-value**
$\hat{\gamma}_{0000}$	5.04 e-3	1.19 e-4	4.24

Authors note: This figure is removed due to confidentiality.

Figure 6.7: Logarithmically scaled heatmap of the excitation effect between different configuration items. Additionally, a dendrogram showing the results of complete linkage using an Euclidean distance is visualized. Furthermore, seven block regarding the transferring configuration item are created.

Remark 6.4.2. Moreover, we can cluster the receiving configuration items $k \in CI$. This resulted in similarly behaving *chunks* of configuration items. The results can be found in Figure C.3.

6.4.3. Model 3: CI nested in BA

We can once again decompose the behavior of business applications with respect to other business applications. However, in this model, we will account for the variation within each business application caused by the configuration item. This results in the following model:

$$\log_{10}(\alpha_{hjkl\leftarrow h'j'k'l}) = \kappa_{00kl\leftarrow 00k'l'} + \epsilon_{hjkl\leftarrow h'j'k'l'},$$
$$\kappa_{00kl\leftarrow 00k'l'} = \beta_{000l\leftarrow 000l'} + z_{00kl\leftarrow 00k'l'},$$
$$\beta_{000l\leftarrow 000l'} = \gamma_{0000} + u_{000l\leftarrow 000l'},$$
where,
$$\epsilon_{hjkl\leftarrow h'j'k'l'} \sim \mathcal{N}(0, \sigma^2),$$
$$z_{00kl\leftarrow 00k'l'} \sim \mathcal{N}(0, \tau_\kappa),$$
$$u_{000l\leftarrow 000l'} \sim \mathcal{N}(0, \tau_\beta).$$

From Table 6.5, we can see that the business application explains 25.7% of the total variation. Furthermore, knowing the configuration item given the business application explains an additional 3.1% of the variation.

Table 6.5: Model 3: a three-level model where the configuration item is nested in the business application.

	Variance	Perc. of total variance	Standard deviation
$\hat{\tau}_\kappa$	3.78 e-4	3.1%	1.95 e-2
$\hat{\tau}_\beta$	3.11 e-3	25.7%	5.57 e-2
$\hat{\sigma}^2$	8.63 e-3		9.29 e-2

	Estimate	Std. Error	t-value
$\hat{\gamma}_{0000}$	4.27 e-3	9.44 e-4	4.53

6.4.4. Model 4: Nature nested in CI, CI nested in BA

If we additionally include the arrival nature, we obtain the full model

$$\begin{aligned}
\log_{10}(\alpha_{hjkl\leftarrow h'j'k'l}) &= \pi_{0jkl\leftarrow 0j'k'l'} + e_{hjkl\leftarrow h'j'k'l'} \\
\pi_{0jkl\leftarrow 0j'k'l'} &= \kappa_{00kl\leftarrow 00k'l'} + r_{0jkl\leftarrow 0j'k'l'} \\
\kappa_{00kl\leftarrow 00k'l'} &= \beta_{000l\leftarrow 000l'} + z_{00kl\leftarrow 00k'l'}, \\
\beta_{000l\leftarrow 000l'} &= \gamma_{0000} + u_{000l\leftarrow 000l'}, \\
&\text{where,} \\
\epsilon_{hjkl\leftarrow h'j'k'l'} &\sim \mathcal{N}(0,\, \sigma^2), \\
z_{00kl\leftarrow 00k'l'} &\sim \mathcal{N}(0,\, \tau_\kappa), \\
r_{0jkl\leftarrow 0j'k'l'} &\sim \mathcal{N}(0,\, \tau_\pi) \\
u_{000l\leftarrow 000l'} &\sim \mathcal{N}(0,\, \tau_\beta).
\end{aligned}$$

In this model, the arrival nature is modeled as a random effect. It can be observed that the knowledge of the configuration item accounts for 0% of the total variation. In Section 6.5.3, we will elaborate on the implications this model poses.

Table 6.6: Model 4: a four-level model where the arrival nature is nested in the configuration item, and the configuration item is nested in the business application. This arrival nature is assumed to be a random effect.

	Variance	Perc. of total variance	Standard deviation
$\hat{\tau}_\pi$	1.95 e-3	13.9%	4.42 e-2
$\hat{\tau}_\kappa$	0	0.0%	0
$\hat{\tau}_\beta$	3.57 e-3	25.4%	5.97 e-2
$\hat{\sigma}^2$	8.51 e-3		9.22 e-2

	Estimate	Std. Error	t-value
$\hat{\gamma}_{0000}$	4.45 e-3	1.03 e-3	4.33

6.4.5. Model 5: Fixed nature and CI

So far we have considered all variables of interest to be of random effect. However, in Section 6.1, we additionally described the fixed effect. Here we saw that we model a variable as a random effect if we can consider the variable as a sample of the entire set of possibilities. This is very intuitive for both the configuration items and business applications: we can see them as coming from a distribution of configuration items or business applications, respectively. Might we observe a new, previously unobserved, business application, then we generalize observation from results as from the model in Section 6.4.4.

This analogy does not hold for the arrival nature. There are only two arrival natures, namely events and incidents. Moreover, there is no distribution of arrival natures. Therefore it is more natural to model the arrival nature as a fixed effect and assume it has a systematic and constant influence on the excitation value. This results in the following model, where the intercept is no longer a fixed γ, but dependent on the *transferring* and *receiving* arrival nature k and k', respectively. This re-

sults in four fixed effects, one for each combination of arrival natures. The respective model now becomes

$$
\begin{aligned}
\log_{10}(\alpha_{hjkl\leftarrow h'j'k'l}) &= \kappa_{00kl\leftarrow 00k'l'} + e_{hjkl\leftarrow h'j'k'l'} \\
\kappa_{00kl\leftarrow 00k'l'} &= \beta_{000l\leftarrow 000l'} + 1_{(k=I,k'=I)}\gamma_{I\leftarrow I} + 1_{(k=E,k'=I)}\gamma_{E\leftarrow I} \\
&\quad + 1_{(k=I,k'=I)}\gamma_{I\leftarrow I} \\
\beta_{000l\leftarrow 000l'} &= \gamma_{E\leftarrow E} + u_{000l\leftarrow 000l'}, \\
&\text{where,} \\
\epsilon_{hjkl\leftarrow h'j'k'l'} &\sim \mathcal{N}(0,\, \sigma^2), \\
u_{000l\leftarrow 000l'} &\sim \mathcal{N}(0,\, \tau_\beta).
\end{aligned}
$$

Here $1_{\{\cdot\}}$ refers to the indicator function. The estimated variables can be found in Table 6.7. Notice the percentage of variation explained by the configuration item is identical to that in Table 6.4. We will further elaborate on these results in Section 6.5.3.

Additionally, it is important to elaborate on the interpretation of the estimated fixed effects in this model. In this context, the fixed effects between events, denoted as $\gamma_{E\hat{\leftarrow}E}$, represent the grand mean. The other three fixed effects, namely $\hat{\gamma}_{I\leftarrow E}$, $\hat{\gamma}_{E\leftarrow I}$ and $\hat{\gamma}_{I\leftarrow I}$, should be understood as the *additional fixed effects with respect to the grand mean* $\gamma_{E\hat{\leftarrow}E}$. For example, the actual fixed effect for incidents on incidents can be calculated as $\hat{\gamma}E \leftarrow E + \hat{\gamma}I \leftarrow I$.

Table 6.7: Model 5: a three-level model where the arrival nature is nested in the configuration item. The arrival nature is modeled as a fixed effect.

	Variance	**Perc. of total variance**	**Standard deviation**
$\hat{\tau}_\kappa$	4.04 e-3	31.8%	6.35 e-2
$\hat{\sigma}^2$	8.63 e-3		9.31 e-2

	Estimate	**Std. Error**	**t-value**
$\hat{\gamma}_{E\leftarrow E}$	4.81e-03	1.19e-03	4.03
$\hat{\gamma}_{I\leftarrow E}$	7.80e-05	1.22e-04	0.64
$\hat{\gamma}_{E\leftarrow I}$	6.62e-04	1.22e-04	5.44
$\hat{\gamma}_{I\leftarrow I}$	1.52e-04	1.23e-04	1.24

6.4.6. Model 6: Fixed nature and CI nested in BA.

Similarly, we can change the random effect for the nature of the model from Section 6.4.4 to a fixed effect. This results in the following model:

$$
\begin{aligned}
\log_{10}(\alpha_{hjkl\leftarrow h'j'k'l}) &= \pi_{0jkl\leftarrow 0j'k'l'} + e_{hjkl\leftarrow h'j'k'l'} \\
\pi_{0jkl\leftarrow 0j'k'l'} &= \kappa_{00kl\leftarrow 00k'l'} + 1_{(k=I,k'=E)}(\gamma_{I\leftarrow E}) + 1_{(k=E,k'=I)}\gamma_{E\leftarrow I} \\
&\quad + 1_{(k=I,k'=I)}\gamma_{I\leftarrow I} \\
\kappa_{00kl\leftarrow 00k'l'} &= \beta_{000l\leftarrow 000l'} + z_{00kl\leftarrow 00k'l'}, \\
\beta_{000l\leftarrow 000l'} &= \gamma_{E\leftarrow E} + u_{000l\leftarrow 000l'}, \\
&\text{where,} \\
\epsilon_{hjkl\leftarrow h'j'k'l'} &\sim \mathcal{N}(0,\, \sigma^2), \\
z_{00kl\leftarrow 00k'l'} &\sim \mathcal{N}(0,\, \tau_\kappa), \\
u_{000l\leftarrow 000l'} &\sim \mathcal{N}(0,\, \tau_\beta).
\end{aligned}
$$

The fixed and random effects are presented in Table 6.8. Once more, we observe a remarkable similarity between the explained variation with the model from Section 6.4.3. Indeed, as with the fixed effect

model in the previous section, it is important to carefully consider the interpretation of the estimated fixed effect variables. Finally, we will further elaborate on the results from this model in Section 6.5.3.

Table 6.8: Model 6: a four-level model where the arrival nature is nested in the configuration item, and the configuration item is nested in the business application. The arrival nature is modeled as a fixed effect.

	Variance	**Perc. of total variance**	**Standard deviation**
$\hat{\tau}_{\kappa}$	3.78 e-4	3.1%	1.95 e-2
$\hat{\tau}_{\beta}$	3.11 e-3	25.7%	5.57 e-2
$\hat{\sigma}^2$	8.63 e-3		9.29 e-2

	Estimate	**Std. Error**	**t-value**
$\hat{\gamma}_{E\leftarrow E}$	4.04 e-03	9.47 e-04	4.26
$\hat{\gamma}_{I\leftarrow E}$	3.01 e-05	1.22 e-04	0.246
$\hat{\gamma}_{E\leftarrow I}$	7.25 e-04	1.22 e-04	5.93
$\hat{\gamma}_{I\leftarrow I}$	1.71 e-04	1.25 e-04	1.38

6.5. Consequences for Software Architecture

So far, our analysis has focused on arrivals from the IT monitoring data stream for one business unit only. Therefore, we are not able to draw conclusions between different business units. However, for the other four levels of service, namely the business application, configuration item, nature, and cluster level, we can indeed illustrate the implications for software architecture.

6.5.1. Business application level consequences

In particular, we estimated the associations between different business applications $l, l' \in BA$ and visualized them in Figure 6.6. After clustering blocks of business applications that behave similarly together, we can draw several conclusions.

Firstly, there are 52 business applications and 13 synthetically created `Business Application Groups`. Interestingly, the synthetic `Business Application Groups` do not exhibit substantially different behavior compared to genuine business applications. This is evident from the fact that the synthetic business application groups are clustered together with genuine business applications in similar blocks. Additionally, it can be found the seven created blocks of similarly behaving business applications are not of equal size. On the left side, we observe two groups consisting of four and three business applications, while on the right side, there is a larger block consisting of 31 business applications. This suggests that the coherence between business applications is distributed unevenly; some business applications interact only with a few others, while others interact with almost half of the total number of business applications.

Authors note: This paragraph is removed due to confidentiality.

Unfortunately, due to time constraints, we were unable to engage in discussions with system engineers to clarify the difference between business applications that share similar names but are distinguished by a numerical suffix. Nevertheless, the noteworthy observation persists that business applications with similar names do not consistently cluster together in the same block.

6.5.2. Configuration item level consequences

In a similar vein, we have estimated associations between the configuration items $k, k' \in CI$ and visualized them in Figure 6.7. As in the case of Figure 6.6, it is evident that the blocks containing configuration items are not equal in size. Specifically, among the 58 configuration items of interest, we find two blocks each comprising two configuration items, alongside two separate blocks, each featuring only one configuration item.

Indeed, a notable discovery pertains to the `UNKNOWN` configuration item. This category encompasses all arrivals that were recorded on the business application but could not be attributed to any specific configuration item. Intriguingly, it appears that arrivals assigned to the `UNKNOWN` configuration item do

not exhibit substantial associations with arrivals assigned to any of the other configuration items. This is evident from their relatively low estimated excitation values.
Conversely, it is worth noting the behavior of the other three smaller blocks. These blocks display notably high excitation values in relation to a specific configuration item *name removed*. However, due to our limited knowledge of the semantics associated with these configuration items, we are unable to draw definitive conclusions regarding the validity or appropriateness of these results. Therefore, engaging in discussions with system engineers to gain insights into these findings would be of considerable interest for a more comprehensive interpretation.

Finally, in our analysis, we initially anticipated observing significant self-excitation effects on the diagonal of the matrix. This would mean that arrivals within the same configuration item would be strongly associated with each other. However, this expectation was not confirmed in Figure 6.7. This suggests that configuration items form more of a connected network, and to a lesser extent, they influence arrivals on the same configuration item. The same conclusion can be drawn for business applications based on Figure 6.6.

6.5.3. Arrival nature level consequences

We found that the percentage of variation explained by the configuration item is identical (31.8 %) regardless of whether we incorporate the arrival nature or not (see Tables 6.4 and 6.7). This can be interpreted in the following way: it seems that knowledge of the arrival nature does not substitute for knowledge of the configuration item at all. Otherwise, including the arrival nature would reduce the percentage of variation explained by the configuration item. This is coherent with the thought that knowing the arrival nature for a specific arrival provides us with no additional knowledge regarding the configuration items on which that arrival occurred, and conversely. In this sense, we can say that the information between the configuration item and the arrival nature is *orthogonal*. Additionally, the same conclusion can be drawn with respect to the business application (See Tables 6.5 and 6.8).

6.5.4. Message cluster level consequences

From the models in Section 6.4.1 to 6.4.6, we elucidated the variation explained by each of the levels of service. This facilitated the clustering of the business applications and configuration items in different blocks. Additionally, we demonstrated that the information provided by each of these service levels cannot be replaced by relying on knowledge about the arrival nature S. However, instead of decomposing the estimated excitation matrix $\hat{\mathcal{A}}$ in each of these levels of service, we can instead directly identify the top-K highest estimated associations by ascendingly ordering each of the excitation values $\alpha_{hjkl \leftarrow h'j'k'l'}$. Furthermore, we can specifically examine the top-K' excitation values $\alpha_{ijkl \leftarrow i'j'k'l'}$ where $k = E$ and $k' = I$, which represents events associated to incidents.

Authors note: This paragraph is removed due to confidentiality.

Table 6.9: Top K = 3 estimated excitation values $\hat{\alpha}_{hjkl \leftarrow h'j'k'l'}$.

Authors note: This table is removed due to confidentiality.

7

Conclusion & Discussion

In this thesis, we unveiled to what extent marked Hawkes processes can contribute to providing a comprehensive overview of the complex relations between different levels of service in large-scale service systems. In summary, we have seen that the IT monitoring data stream consists of two arrival natures, namely events and incidents. We have seen how we can construct clusters for each of the two natures by making use of the message feature. Subsequently, we assumed independence between business units. Therefore clustering could be performed *per business unit*. For each business unit $m \in BU$, this resulted in mark space $\mathcal{U}^m = BA \times CI \times S \times CLU^m$. For each of the arrivals, a unique five-level mark could therefore be assigned, consisting of the *(1) message cluster h, (2) arrival nature j, (3) configuration item k, (4) business application l, (5) business unit m*. Given the marked monitoring data stream, we considered the arrivals on one specific business unit only. Subsequently, we considered the marked Hawkes process framework with an exponential memory kernel. This enabled us to estimate excitation matrix $\hat{\mathcal{A}}$, which captures the estimated excitation between all marks $u, u' \in \mathcal{U}^m$. By substituting back the mark, we obtained $\hat{\alpha}_{hjkl \leftarrow h'j'k'l'}$. From here on, we have established a hierarchical Hawkes model on the service architecture of a large commercial bank.

Finally, this thesis was structured through means of four research questions. In this section, we summarize the answers to the research questions that were tackled throughout this thesis and discuss future work.

Research Question 1: How can we design a hierarchical architecture that resembles the operations of large-scale service systems?

First, we have established four distinct levels of service: *business unit*, *business application*, *configuration item* and *arrival nature*. Through their definition, these levels inherently form a hierarchical architecture. However, it became evident that using "arrival nature" as the lowest-level hierarchy lacks the necessary granularity for our problem analysis. Consequently, we introduced a new, lower level of service. This new level was obtained by grouping arrivals together based on their messages. This led to the development of Research Question 2.

Research Question 2: How can IT messages with similar semantics be grouped together?

We demonstrated that relying solely on template mining, such as demonstrated by the use of *Drain*, is insufficient for effectively clustering semantically similar messages. Consequently, we explored an approach that combines message embedding, followed by UMAP dimension reduction and HDBSCAN clustering. A comparative analysis unveiled that TF-IDF embedding outperforms Doc2vec embedding, as evidenced by higher DBCV clustering scores and lower percentages of messages with low certainty regarding their assigned cluster. Moreover, outcomes from an extensive grid search indicated almost identical hyperparameter results for both event and incident records. Additionally, we discussed how a low number of clusters leads to high variance, while a high number of clusters introduces bias.

Research Question 3: How can the marked Hawkes process be employed to capture interactions among arrivals? And how can the interactions be estimated?

We have demonstrated that we can model the cross-exciting behavior observed in large-scale service systems by constructing a mark space $\mathcal{U}^m$ for each business unit $m \in BU$. This mark space effectively integrates message clusters, arrival nature, and three additional levels of service. We have introduced a class of memory kernels that can be separated into two components: one governing the instantaneous increase in probability and another determining the rate of decay. Indeed, the exponential memory kernel belongs to this class, which allows us to derive an estimated excitation matrix denoted as $\hat{\mathbb{A}}$. We demonstrated that the estimated excitation matrix can be obtained through both maximum likelihood and least squares estimators. Furthermore, we described the *Tick* package which has incorporated these functionalities through two estimators, namely the `ExpKern` and `ADM4` estimator. Additionally, it was found both of these estimators can be employed with various regularization techniques to enforce a sparse estimated excitation matrix.

Research Question 4: How can the estimated excitation matrix contribute to understanding the associations within a level of service?

First of all, we have seen that the choice of characteristic time (which determines the duration during which an arrival can have an influence on others) exerts a substantial influence on the estimated excitation matrix and, consequently, on the comprehension of service levels. Secondly, we have demonstrated that each of the estimated excitation values $\hat{\alpha} \in \hat{\mathcal{A}}$ can be decomposed into its constituent components from the mark space, resulting in $\hat{\alpha}_{hjkl \leftarrow h'j'k'l}$. The employed hierarchical linear models elucidate the variation explained by each of these service levels, facilitating the grouping of components at the same level of service into different blocks. Notably, we have highlighted which message clusters are estimated to have resulted in an increase in the probability of observing other message clusters and posted that these associations appear sensible, particularly in light of the `incident ID` feature.

7.1. Discussion and future research

Throughout the process of working with event and incident data for over a year, we have compiled a set of recommendations. These are intended to serve as guidelines for AIOps engineers looking to further expand the application of the hierarchical Hawkes model. we have developed recommendations specifically for system engineers, outlining steps they can take to adapt the IT architecture such that they can harness the full potential of the hierarchical Hawkes model effectively.

Currently, we only use the first arrival time of each event record. It could therefore prove advantageous to divide the timeline into discrete bins and then employ an estimation technique to quantify the number of occurrences in each bin. This would enable us to assign tally counts to these discrete bins.
However, the current set of expiration durations, denoted as $TEXP$, permits a maximum expiration of 1 week. This entails that the minimum bin size must also be set to at least 1 week, which may not be practically feasible. Although we acknowledge that recording all arrivals separately is not feasible from a data storage perspective, we recommend that system engineers consider reducing the allowed expiration durations to a more practical range, for instance, $TEXP := \{0\ s, 1\ m, 15\ m\}$.

Moreover, to derive valuable insights from the excitation matrix, it is crucial that the mark space $\mathcal{U}^m$ accurately represents the levels of service. Therefore it should be evaluated with system engineers if the current mark space $\mathcal{U}^m$ should be augmented to incorporate additional levels of service. Additionally, the current mark space does not encompass parameters like priority ($prio_i$) and level environment ($level\_id_i$). It would be beneficial to explore whether incorporating these features could lead to a more comprehensive and representative mark space.

Furthermore, the current hierarchical Hawkes model assumes marked Hawkes processes with a constant

baseline. However, as demonstrated in Section 3.2.1, patterns like seasonality and declining trends in the number of arrivals can be observed. Therefore, exploring the integration of this information into a baseline function and assessing its influence on the excitation matrix presents an intriguing path for future research. Additionally, it is established by means of conversations with systems engineers that it is likely for arrivals to lead to subsequent arrivals only after a short time interval. Hence, exploring memory kernels capable of incorporating a delay in intensity increase, such as the Gamma distributed memory kernel mentioned in Section 5.2.1, becomes a subject of interest.

Furthermore, it is worth noting that the analysis conducted thus far has been restricted to real-world data from a single business unit. Expanding this investigation to include all n_{bu} business units and comparing the findings across these various units could therefore provide valuable insights for Software Architecture.
In our work, we estimate the excitation matrix using the determined cluster. We then did a post-processing of this estimated excitation matrix by fitting a hierarchical linear mixed model. For future work, it could be nice to establish a formal hierarchical Hawkes model that would encompass directly the hierarchical nature of the data into the fitting of the process. This would bypass our 2-step procedure by fitting directly a well-specified model. Furthermore, one could also study such a statistical model to establish theoretical properties (consistency and the asymptotic law of the estimators) that could be used to construct valid statistical tests and confidence intervals for the estimated parameters.

References

[1] Roozbeh Aghili, Heng Li, and Foutse Khomh. *Studying the Characteristics of AIOps Projects on GitHub*. 2022. arXiv: 2212.13245 [cs.SE].

[2] Muhammad Sidik Asyaky and Rila Mandala. "Improving the Performance of HDBSCAN on Short Text Clustering by Using Word Embedding and UMAP". In: Institute of Electrical and Electronics Engineers Inc., 2021. ISBN: 9781665417433. DOI: 10.1109/ICAICTA53211.2021.9640285.

[3] E. Bacry et al. "Tick: A python library for statistical learning, with an emphasis on hawkes processes and time-dependent Models". In: *Journal of Machine Learning Research* 18 (Apr. 2018), pp. 1–5.

[4] Emmanuel Bacry et al. *Sparse and low-rank multivariate Hawkes processes*. 2020. arXiv: 1501.00725.

[5] European Central Bank. *Sanctions*. Accessed: 2023-08-04. URL: https://www.bankingsupervision.europa.eu/banking/tasks/sanctions/html/index.en.html.

[6] Joeran Beel et al. "Research-paper recommender systems: A literature survey". In: *International Journal on Digital Libraries* (July 2015), pp. 1–34. DOI: 10.1007/s00799-015-0156-0.

[7] Samir Bhatt et al. *Semi-Mechanistic Bayesian Modeling of COVID-19 with Renewal Processes*. Dec. 2020. URL: http://arxiv.org/abs/2012.00394.

[8] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. Beijing: O'Reilly, 2009. ISBN: 978-0-596-51649-9. DOI: http://my.safaribooksonline.com/9780596516499. URL: http://www.nltk.org/book.

[9] Martin Bompaire. "Machine learning based on Hawkes processes and stochastic optimization. (Apprentissage automatique avec les processus de Hawkes et l'optimisation stochastique)". In: 2019.

[10] Alexandre Brouste and · Christian Farinetto. "Fast and asymptotically efficient estimation in the Hawkes processes". In: *Japanese Journal of Statistics and Data Science 2023* (Feb. 2023), pp. 1–19. ISSN: 2520-8764. DOI: 10.1007/S42081-023-00186-2. URL: https://link.springer.com/article/10.1007/s42081-023-00186-2.

[11] Alvaro Cartea, Samuel N. Cohen, and Saad Labyad. "Gradient-based estimation of linear Hawkes processes with general kernels". In: (Nov. 2021). URL: http://arxiv.org/abs/2111.10637.

[12] Junjie Chen et al. "Continuous Incident Triage for Large-Scale Online Service Systems". In: Nov. 2019, pp. 364–375. DOI: 10.1109/ASE.2019.00042.

[13] Junjie Chen et al. "How Incidental are the Incidents? Characterizing and Prioritizing Incidents for Large-Scale Online Service Systems". In: Institute of Electrical and Electronics Engineers Inc., Sept. 2020, pp. 373–384. ISBN: 9781450367684. DOI: 10.1145/3324884.3416624.

[14] Yujun Chen et al. "Outage prediction and diagnosis for cloud service systems". In: Association for Computing Machinery, Inc, May 2019, pp. 2659–2665. ISBN: 9781450366748. DOI: 10.1145/3308558.3313501.

[15] E. S. Chornoboy, L. P. Schramm, and A. F. Karr. "Maximum likelihood identification of neural point process systems". In: *Biological Cybernetics* 59 (4-5 Sept. 1988), pp. 265–275. ISSN: 03401200. DOI: 10.1007/BF00332915.

[16] *Cost of Data Center Outages*. Podemon Institute, Jan. 2016.

[17] D.R. Cox and V. Isham. *Point Processes*. Routledge., 1980. DOI: https://doi.org/10.1201/9780203743034.

[18] Daan van Monsjou. *ING-klanten hebben last van storing waardoor geld overboeken niet mogelijk is*. https://tweakers.net/nieuws/209044/ing-klanten-hebben-last-van-storing-waardoor-geld-overboeken-niet-mogelijk-is.html. Accessed: 2023-04-24, Published: 2023-04-24.

[19] D. J. Daley and D. Vere-Jones. *An introduction to the theory of point processes. Vol. I*. Second. Probability and its Applications (New York). Elementary theory and methods. New York: Springer-Verlag, 2003, pp. xxii+469. ISBN: 0-387-95541-0.

[20] Cedric De Boom et al. "Learning Semantic Similarity for Very Short Texts". In: *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. 2015, pp. 1229–1234. DOI: 10.1109/ICDMW.2015.86.

[21] Alexis Derumigny and Johannes Schmidt-Hieber. *On lower bounds for the bias-variance trade-off*. 2023. arXiv: 2006.00278 [math.ST].

[22] Michael Eichler, Rainer Dahlhaus, and Johannes Dueck. "Graphical Modeling for Multivariate Hawkes Processes with Nonparametric Link Functions". In: (May 2016). URL: http://arxiv.org/abs/1605.06759.

[23] C. W. J. Granger. "Investigating Causal Relations by Econometric Models and Cross-spectral Methods". In: *Econometrica* 37.3 (1969), pp. 424–438. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/1912791 (visited on 05/06/2023).

[24] Asela Gunawardana, Christopher Meek, and Puyang Xu. "A Model for Temporal Dependencies in Event Streams". In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor et al. Vol. 24. Curran Associates, Inc., 2011. URL: https://proceedings.neurips.cc/paper_files/paper/2011/file/c9f95a0a5af052bffce5c89917335f67-Paper.pdf.

[25] Xin Guo et al. "Consistency and Computation of Regularized MLEs for Multivariate Hawkes Processes". In: (2018). URL: https://www.researchgate.net/publication/328118461_Consistency_and_Computation_of_Regularized_MLEs_for_Multivariate_Hawkes_Processes.

[26] Peter Hall and Feng Chen. "Inference for a nonstationary self-exciting point process with an application in ultra-high frequency financial data modeling". In: *Journal of Applied Probability, Vol. 50, No. 4* (Dec. 2013), pp. 1006–1024. URL: https://www.jstor.org/stable/43284141.

[27] Stephen Hardiman, Nicolas Bercot, and Jean-Philippe Bouchaud. "Critical reflexivity in financial markets: A Hawkes process analysis". In: *Physics of Condensed Matter* 86 (Feb. 2013). DOI: 10.2139/ssrn.2221243.

[28] Alan G Hawkes. "Point Spectra of Some Mutually Exciting Point Processes". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 33 (3 Oct. 1971), pp. 438–443. DOI: 10.1111/j.2517-6161.1971.tb01530.x.

[29] Alan G Hawkes. "Spectra of some self-exciting and mutually exciting point processes". In: *Biometrika* 58 (1 1971), p. 83. URL: http://biomet.oxfordjournals.org/.

[30] Alan G Hawkes and David Oakes. "A Cluster Process Representation of a Self-Exciting Process". In: *Journal of Applied Probability* 11 (3 1974), pp. 493–503. URL: https://about.jstor.org/terms.

[31] Martin Hilbert. "Scale-free power-laws as interaction between progress and diffusion". In: *Complexity* 19.4 (2014), pp. 56–65. DOI: https://doi.org/10.1002/cplx.21485. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cplx.21485. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cplx.21485.

[32] C. Huyen. *Designing Machine Learning Systems: An Iterative Process for Production-ready Applications*. O'Reilly Media, Incorporated, 2022. ISBN: 9781098107963.

[33] IBM. *Business Applications*. https://www.ibm.com/docs/en/taddm/7.3.0?topic=using-business-applications. Accessed: 2023-17-06, Published: 2022-29-11.

[34] IBM. *Configuration items*. https://www.ibm.com/docs/en/cdfsp/7.6.1.x?topic=overview-configuration-items. Accessed: 2023-03-22, Published: 2022-19-05.

[35] IBM. *Organizational business units*. https://www.ibm.com/docs/el/license-metric-tool?topic=scenarios-organizational-business-units. Accessed: 2023-17-06, Published: 2023-29-03.

[36] IBM. *What is AIOps?* https://www.ibm.com/topics/aiops. Accessed: 2023-15-05. Apr. 2022.

[37] Tsuyoshi Idé et al. *Cardinality-Regularized Hawkes-Granger Model*. 2022. eprint: 2208.10671.

[38] Gartner Inc. *AIOps (Artificial Intelligence for IT Operations)*. Accessed: 2023-08-04. URL: https://www.gartner.com/en/information-technology/glossary/aiops-artificial-intelligence-operations.

[39] Guofei Jiang et al. "Ranking the importance of alerts for problem determination in large computer systems". In: *Cluster Computing* 14 (June 2009), pp. 213–227. DOI: 10.1007/s10586-010-0120-0.

[40] Julia Kagan. *What Is Wholesale Banking? Types of Services and Example*. https://www.investopedia.com/terms/w/wholesalebanking.asp. Accessed: 2023-13-06, Published: 2020-04-05.

[41] Matthias Kirchner. "An estimation procedure for the Hawkes process". In: *Quantitative Finance* 17.4 (Sept. 2016), pp. 571–595. DOI: 10.1080/14697688.2016.1211312. URL: https://doi.org/10.1080%2F14697688.2016.1211312.

[42] P.J. Laub, Y. Lee, and T. Taimre. *The Elements of Hawkes Processes*. Springer International Publishing, 2022. ISBN: 9783030846398. URL: https://books.google.nl/books?id=HJtXEAAAQBAJ.

[43] Patrick J Laub, Thomas Taimre, and Philip K Pollett. "Hawkes Processes". In: (2015).

[44] Laurent Lesage et al. "Hawkes Processes Framework With a Gamma Density As Excitation Function: Application to Natural Disasters for Insurance". In: *Methodology and Computing in Applied Probability* 24 (Mar. 2022), pp. 1–29. DOI: 10.1007/s11009-022-09938-1.

[45] Zhenmin Li et al. *UCLog: A unified, correlated logging architecture for intrusion detection Log Anonymization and Information Management (LAIM) View project MyProxy View project UCLog: A Unified, Correlated Logging Architecture for Intrusion Detection*. 2005. URL: https://www.researchgate.net/publication/228746403.

[46] Elizabeth D Liddy. "Natural language processing". In: (2001).

[47] Derek Lin et al. "Unveiling clusters of events for alert and incident management in large-scale enterprise it". In: Association for Computing Machinery, 2014, pp. 1630–1639. ISBN: 9781450329569. DOI: 10.1145/2623330.2623360.

[48] Qingwei Lin et al. "Log clustering based problem identification for online service systems". In: IEEE Computer Society, May 2016, pp. 102–111. ISBN: 9781450341615. DOI: 10.1145/2889160.2889232.

[49] Jinyang Liu et al. "Logzip: Extracting Hidden Structures via Iterative Clustering for Log Compression". In: (Sept. 2019). URL: http://arxiv.org/abs/1910.00409.

[50] Jian Guang Lou et al. "Experience report on applying software analytics in incident management of online service". In: *Automated Software Engineering* 24 (4 Dec. 2017), pp. 905–941. ISSN: 15737535. DOI: 10.1007/s10515-017-0218-1.

[51] Jian Guang Lou et al. "Software analytics for incident management of online services: An experience report". In: 2013, pp. 475–485. ISBN: 9781479902156. DOI: 10.1109/ASE.2013.6693105.

[52] Yingzhe Lyu et al. "Towards a Consistent Interpretation of AIOps Models". In: *ACM Transactions on Software Engineering and Methodology* 31 (1 Jan. 2022), pp. 1–38. ISSN: 1049-331X. DOI: 10.1145/3488269.

[53] Leland McInnes, John Healy, and Steve Astels. "hdbscan: Hierarchical density based clustering". In: *The Journal of Open Source Software* 2 (Mar. 2017). DOI: 10.21105/joss.00205.

[54] Christopher Meek. "Toward Learning Graphical and Causal Process Models". In: *CIUAI*. 2014.

[55] Seyed Ali Mirheidari, Sajjad Arshad, and Rasool Jalili. *Alert Correlation Algorithms: A Survey and Taxonomy*. 2013.

[56] Davoud Moulavi et al. "Density-based clustering validation". In: vol. 2. Society for Industrial and Applied Mathematics Publications, 2014, pp. 839–847. ISBN: 9781510811515. DOI: 10.1137/1.9781611973440.96.

[57] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. "Natural language processing: an introduction". In: *Journal of the American Medical Informatics Association* 18.5 (Sept. 2011), pp. 544–551. ISSN: 1067-5027. DOI: 10.1136/amiajnl-2011-000464. eprint: https://academic.oup.com/jamia/article-pdf/18/5/544/5962687/18-5-544.pdf. URL: https://doi.org/10.1136/amiajnl-2011-000464.

[58] Y Ogata. "On Lewis' simulation method for point processes". In: (1981). DOI: 10.1109/TIT.1981.1056305.

[59] Yosihiko Ogata. "The asymptotic behaviour of maximum likelihood estimators for stationary point processes". In: *Ann. Inst. Statist. Math* 30 (1978), pp. 243–261.

[60] Yosihiko Ogata and Yoslhlko Ogata. "Statistical Models for Earthquake Occurrences and Residual Analysis for Point Processes". In: *Journal of the American Statistical Association* 83 (1988), pp. 9–27. ISSN: 0162-1459. DOI: 10.1080/01621459.1988.10478560. URL: https://www.tandfonline.com/action/journalInformation?journalCode=uasa20.

[61] T Ozaki. "Maximum likelihood estimation of Hawkes' self-exciting point processes". In: *Ann. Inst. Statist. Math* 31 (1979), pp. 145–155.

[62] Jakob Gulddahl Rasmussen. "Lecture Notes: Temporal Point Processes and the Conditional Intensity Function". In: (2018).

[63] Patricia Reynaud-Bouret and Sophie Schbath. "Adaptive estimation for hawkes processes; Application to genome analysis". In: *Annals of Statistics* 38 (5 Oct. 2010), pp. 2781–2822. ISSN: 00905364. DOI: 10.1214/10-AOS806.

[64] Marian-Andrei Rizoiu, Young Lee, and Swapnil Mishra. "A Tutorial on Hawkes Processes for Events in Social Media". In: Dec. 2017, pp. 191–218. ISBN: 9781970001075.

[65] Marian-Andrei Rizoiu, Young Lee, and Swapnil Mishra. "A Tutorial on Hawkes Processes for Events in Social Media". In: Dec. 2017, pp. 191–218. ISBN: 9781970001075.

[66] Sabin Roman and Francesco Bertolotti. "A master equation for power laws". In: *Royal Society Open Science* 9.12 (Dec. 2022). DOI: 10.1098/rsos.220531. URL: https://doi.org/10.1098%2Frsos.220531.

[67] Paula Rooney. *Microsoft's CEO: 80-20 Rule Applies To Bugs, Not Just Features*. https://www.crn.com/news/security/18821726/microsofts-ceo-80-20-rule-applies-to-bugs-not-just-features.htm. Oct. 2002.

[68] Peter J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7. URL: https://www.sciencedirect.com/science/article/pii/0377042787901257.

[69] Tiago Santos, Florian Lemmerich, and Denis Helic. *Surfacing Estimation Uncertainty in the Decay Parameters of Hawkes Processes with Exponential Kernels*. 2021. arXiv: 2104.01029 [cs.LG].

[70] Ruben Sipos et al. "Log-based predictive maintenance". In: Association for Computing Machinery, 2014, pp. 1867–1876. ISBN: 9781450329569. DOI: 10.1145/2623330.2623340.

[71] Dylan Slack et al. "Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods". In: Association for Computing Machinery, Inc, Feb. 2020, pp. 180–186. ISBN: 9781450371100. DOI: 10.1145/3375627.3375830.

[72] John Spacey. *11 Examples of a Configuration Item*. https://simplicable.com/IT/configuration-item. Accessed: 2023-01-08.

[73] Tanya Tsui et al. "Spatial clustering of waste reuse in a circular economy: A spatial autocorrelation analysis on locations of waste reuse in the Netherlands using global and local Moran's I". In: *Frontiers in Built Environment* 8 (2022). ISSN: 2297-3362. DOI: 10.3389/fbuil.2022.954642. URL: https://www.frontiersin.org/articles/10.3389/fbuil.2022.954642.

[74] Piet Van Mieghem. *Performance Analysis of Complex Networks and Systems*. Cambridge University Press, 2014. DOI: 10.1017/CBO9781107415874.

[75] Wallstreet Mojo. *Strategic Business Unit*. https://www.wallstreetmojo.com/strategic-business-unit/. Accessed: 2023-17-06.

[76] Song Wei et al. "Causal Graph Discovery from Self and Mutually Exciting Time Series". In: (Jan. 2023). URL: http://arxiv.org/abs/2301.11197.

[77] *What is iDEAL?* https://www.ideal.nl/en/consumers/what-is-ideal/. Accessed: 2023-08-03.

[78] Hongteng Xu, Mehrdad Farajtabar, and Hongyuan Zha. “Learning Granger Causality for Hawkes Processes”. In: (2016).

[79] William Yurcik et al. “UCLog+: A Security Data Management System for Correlating Alerts, Incidents, and Raw Data From Remote Logs”. In: (2005).

[80] Ke Zhang et al. “Automated IT system failure prediction: A deep learning approach”. In: Institute of Electrical and Electronics Engineers Inc., 2016, pp. 1291–1300. ISBN: 9781467390040. DOI: 10.1109/BigData.2016.7840733.

[81] Xu Zhang et al. “Robust log-based anomaly detection on unstable log data”. In: Association for Computing Machinery, Inc, Aug. 2019, pp. 807–817. ISBN: 9781450355728. DOI: 10.1145/3338906.3338931.

[82] Nengwen Zhao et al. “Real-time incident prediction for online service systems”. In: Association for Computing Machinery, Inc, Nov. 2020, pp. 315–326. ISBN: 9781450370431. DOI: 10.1145/3368089.3409672.

[83] Nengwen Zhao et al. “Understanding and handling alert storm for online service systems”. In: IEEE Computer Society, June 2020, pp. 162–171. ISBN: 9781450371230. DOI: 10.1145/3377813.3381363.

[84] Xiang Zhou et al. “Latent error prediction and fault localization for microservice applications by learning from system trace logs”. In: Association for Computing Machinery, Inc, Aug. 2019, pp. 683–694. ISBN: 9781450355728. DOI: 10.1145/3338906.3338961.

A

Source Code

Plotly does not provide a logarithmic scale for options for heatmaps. We have defined a function for this and shared the results on the *Plotly Community form* [1]

```
"""
Custom code for generating color bar with logarithmic scaled bar.
"""
def colorbar(n):
    """
    Args:
        n: Maximum value to display in bar
    return:
        Dictionary item with heatmap colorbar items
    """

    adjusted_colorbar = dict(
        tick0 = 0,
        title = "Log color scale",
        tickmode = "array",
        tickvals = np.linspace(0, n, n+1),
        ticktext = 10**np.linspace(0, n, n+1))
    return adjusted_colorbar


fig = go.Figure(data = go.Heatmap(
    z = np.log10(window['size']),
    x = window['time'],
    y = window['day'],
    text = window['size'],
    colorscale='Inferno',
    hovertemplate = "Date: %{y} %{x} <br>" + "Count: %{text}",
    colorbar = colorbar(n),
    reversescale = True))
```

[1] https://community.plotly.com/t/how-to-set-log-scale-for-z-axis-on-a-heatmap/292

B

Additional theorems

B.1. Kullback-Leibler divergence

The Kullback-Leibler (KL) divergence, also known as relative entropy, is a measure of how one probability distribution diverges from another.

Definition B.1.1: Kullback-Leibler divergence

Given two discrete probability distributions $P(x)$ and $Q(x)$ over a discrete random variable x, the KL divergence from P to Q is defined as:

$$D_{\mathrm{KL}}(P\|Q) = \sum_x P(x) \log\left(\frac{P(x)}{Q(x)}\right) \tag{B.1}$$

where the sum is taken over all possible values of x, and $P(x)$ and $Q(x)$ represent the probabilities of the corresponding events under distributions P and Q, respectively.

1. Non-negativity: $D_{\mathrm{KL}}(P\|Q) \geq 0$, with equality if and only if $P(x) = Q(x)$ for all x.
2. Non-commutativity: $D_{\mathrm{KL}}(P\|Q) \neq D_{\mathrm{KL}}(Q\|P)$ in general.
3. Invariance under Re-parametrization: $D_{\mathrm{KL}}(P\|Q)$ is invariant under re-parametrization of the random variable x.
4. Additivity: For independent random variables x and y, $D_{\mathrm{KL}}(P(x,y)\|Q(x,y)) = D_{\mathrm{KL}}(P(x)\|Q(x)) + D_{\mathrm{KL}}(P(y)\|Q(y))$.

C

Parameter settings

C.1. Hyperparameter grid HDBSCAN & UMAP

The total hyperparameter grid is as follows

$$\begin{cases} \text{n\_neighbors :} & [12, 13, 14, 15], \\ \text{n\_components :} & [3, 4, 5, 6], \\ \text{min\_cluster\_size :} & [2, 3, 4, 5, 6, 7, 8, 9, 10, 11], \\ \text{cluster\_selection\_epsilon :} & [0.5, 0.6, 0.7], \\ \text{cluster\_selection\_method :} & [\text{eom}, \text{leaf}], \\ \text{metric :} & [\text{euclidean}, \text{manhattan}] \end{cases} \tag{C.1}$$

C.2. Periodogram

A periodogram for the event and incident data stream for a two-month period, as used in Section 3.2.1 can be found in Figures C.1 and C.2. Both highest peaks are attained at approximately $(24 \cdot 60 \cdot 60)^{-1} Hz$, resulting in a seasonal period of approximately 24 hours.

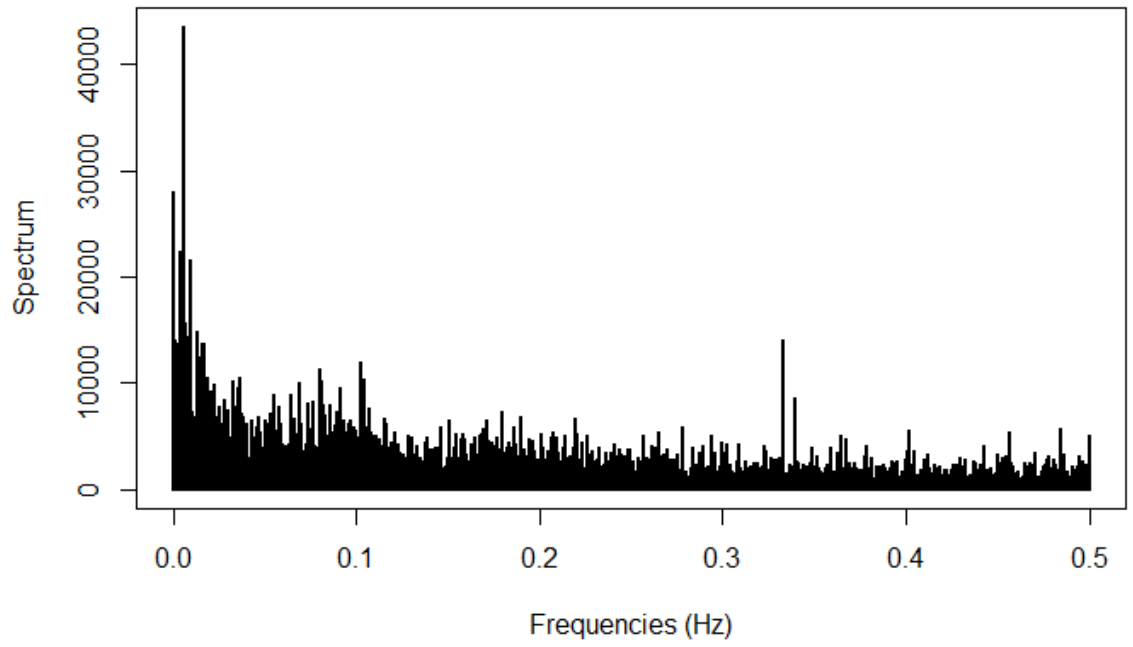

Figure C.1: Periodogram for the arrival time of events.

Authors note: This figure is removed due to confidentiality.

Figure C.2: Periodogram for the arrival time of incidents.

C.3. Additional excitation figures

Authors note: This figure is removed due to confidentiality.

Figure C.3: Logarithmically scaled heatmap of the excitation effect between different business applications. Additionally, a dendrogram showing the results of complete linkage using an Euclidean distance is visualized. Furthermore, seven blocks regarding the transferring business application, as well as seven blocks regarding the receiving business application are created.

Authors note: This figure is removed due to confidentiality.

Figure C.4: Logarithmically scaled heatmap of the excitation effect between different configuration items. Additionally, a dendrogram showing the results of complete linkage using an Euclidean distance is visualized. Furthermore, seven blocks regarding the transferring configuration item, as well as seven blocks regarding the receiving configuration item are created.