UNIVERSITÀ DEGLI STUDI DI MILANO

PhD School in
Computer Science

Computer Science Department
"Giovanni Degli Antoni"

PhD in
Computer Science
XXXVII Cycle

Process Mining for Reengineering Circular and Resilient Production Processes

INF/01

PhD candidate:
Rafael Seidi OYAMADA

Advisor:
Prof. Paolo CERAVOLO

School Director:
Prof. Roberto SASSI

Academic Year 2023/2024

Abstract

This thesis explores the integration of *process mining* techniques with *deep learning* models to enhance the understanding, analysis, and optimization of complex business processes. **Process mining** bridges the gap between model-driven and data-driven approaches through a set of techniques that extract insights from **event logs**. Meanwhile, **deep learning** has the potential to model and learn intrinsic patterns and dependencies from data. By combining these fields, the research carried out throughout this thesis addresses key challenges in a specific subfield of process mining called *predictive process monitoring*. In this subfield, the goal is to leverage all the historical data to predict future behaviors within a process. Relevant predictive tasks include, for example, predicting the **next activity**, the **remaining time** of a current instance, or the **outcome** of a current instance.

In the past few years, several papers have been published on **predictive process monitoring** regarding different tasks. Despite the great success and convincing results in terms of predictive accuracy, researchers overlook some principles that are crucial for the advancement of the field. For instance, transforming business and semantic rules from process data into a **latent space** is a key step for training machine learning models. Usually, most proposals in the literature rely on naive techniques to encode categorical attributes from events, such as the well-known **one-hot encoding**. Furthermore, a second level of encoding is required to aggregate a **trace of events** into a single vector array, which is commonly done by simply averaging the attribute values of events. These approaches naturally result in loss of information and fail to properly capture all the intrinsic complexity of process data.

This often-overlooked step introduces significant limitations to modern predictive process monitoring applications. For example, while deep learning models excel at tasks such as predicting the **next activity** or suffixes in a process, these tasks often fall short of addressing more meaningful, practical needs. In real-world scenarios, the usefulness of next-activity predictors is often questionable. A more impactful application lies in **process simulation**, where the goal is to model potential new behaviors within a process before actual changes are imple-

mented, ensuring safety and minimizing risk. Unfortunately, current techniques for representing process data are inadequate, making deep learning models inflexible to effectively support this type of simulation. For example, process data is often encoded as sequences of discrete events without capturing the **contextual relationships** or dependencies between them. This limitation means that a deep learning model trained on such data may accurately predict the next event in a process but cannot simulate complex scenarios, such as how changes in resource allocation or timing constraints might affect the overall workflow.

Therefore, in this thesis, we aim to better understand how to encode **process data** in latent spaces to improve the performance of learned models. We begin this work by reviewing several encoding techniques from other research areas to assess their applicability in **process mining**. We also investigate and motivate the development of feature engineering techniques tailored to this particular type of data. To this end, we systematically design and empirically evaluate extensive experiments for different tasks. In addition, we propose the use of declarative languages that capture *linear temporal logic* over finite traces to improve the data representation of process data for training deep learning models, thus overcoming the inflexibility of these applications for process simulation. Finally, we use all the knowledge and contributions developed in this thesis to propose meaningful solutions in the **industrial sector**. More specifically, we develop three relevant use cases at an Italian company called Avio Aero: (i) a process mining pipeline to extract insights from their data, (ii) an application of **large language models** to enrich this data, and (iii) a combination of both research areas to propose improvements and design a more **resilient** and robust process model.

Contents

Abstract **2**

1 Introduction **14**
1.1 State-of-the-art 16
1.2 Problem Definition 18
1.3 Research Questions 20
1.4 Contribution 21
1.5 Outline 22

2 Background **23**
2.1 Process Mining 23
2.2 Traditional Machine Learning 25
2.3 Basic Concepts of Deep Learning for Generative Tasks 27
2.4 Predictive Process Monitoring 29
2.4.1 Tasks 29
2.4.2 Event Data Encoding 30

3 Process Data Representation **33**
3.1 Present Context in Predictive Process Monitoring 34
3.1.1 Motivation and Related Works 36
3.2 Survey on Encoding Methods 38
3.2.1 Experimental Setup 38
3.2.2 The Encoding Methods 40
3.2.3 On the Impact of Hyperparameters and Logs' Characteristics 47
3.3 Advanced Feature Engineering to Compete against Deep Learning 54
3.3.1 Our Proposal 54
3.3.2 Experimental Setup 55
3.3.3 Evaluation 56
3.4 Final Remarks 59

4 Striking a Balance with CoSMo: Controlling Randomness in Process Simulation 61
4.1 Current State and Limitations of Process Simulation 62
4.2 Fundamental Concepts 64
4.2.1 DECLARE Model 66
4.3 A Framework to Condition Process Simulation Models 67
4.3.1 Conditioned Recurrent Architecture 68
4.3.2 CoSMo Instantiation 69
4.3.3 Conditioned Simulation 70
4.4 Evaluation 71
4.4.1 Experimental Setup 71
4.4.2 Conditioned Simulation of Process Behaviors 75
4.5 Final Remarks 78

5 Sustainable Process Mining: Enhancing Environmental Impact whilst Preserving Predictive Performances 81
5.1 Problem Motivation and Methodology 83
5.2 Finding a Balance between Computational Resources and Predictive Performances 84
5.2.1 Experimental Setup 85
5.2.2 Scalability Analysis 85
5.3 Speeding Pipelines Up to Reduce Energy Consumption 88
5.3.1 Experiments 89
5.4 Analysis of Execution Times for Trace Encoding Methods 90
5.4.1 Fundamentals 91
5.4.2 Speeding Up Deep Learning Training by Efficiently Managing Sequences 91
5.4.3 Experiments 93
5.5 Final Remarks 93

6 Predictive Process Monitoring and Future Directions 96
6.1 Challenges in the Literature 97
6.2 Centralized Open-source Tool for Robust Research 99
6.2.1 Motivation 99
6.2.2 Design and Implementation 100
6.3 SkPM Usage 101
6.3.1 A High-level Overview 102
6.3.2 Remaining Time Prediction Example 103
6.4 Conclusion and Discussion 105

7 Use Case at Avio Aero **106**
7.1 Business Frameworks . . . 107
7.1.1 Information Technology Infrastructure Library . . . 107
7.1.2 Application Management Services . . . 111
7.1.3 ServiceNow . . . 112
7.1.4 Business Frameworks in Big Industries . . . 113
7.2 Process Mining with Business Management Frameworks: Literature Review . . . 114
7.2.1 Quantitative Overview . . . 115
7.2.2 Qualitative Overview . . . 116
7.2.3 An Intersection with Process Mining . . . 117
7.3 Fundamentals . . . 118
7.3.1 ServiceNow . . . 118
7.3.2 Large Language Models . . . 120
7.4 Integrating Process Mining and Large Language Models at Avio Aero . . . 122
7.5 Experiments . . . 124
7.5.1 Experimental Setup . . . 124
7.5.2 Event Log Extraction . . . 125
7.5.3 PM Analytics . . . 128
7.5.4 LLMs . . . 134
7.5.5 Combining PM and LLMs . . . 140
7.6 Final Remarks . . . 147

8 Conclusion **149**

Acknowledgements **177**

Author's Publications **178**

List of Figures

3.1 An example of how event logs can be encoded. Although the illustration uses specific techniques, such as one-hot encoding at the event-level and average aggregation at the trace-level, other techniques can be employed. For instance, learning embedded representations at the trace-level using deep neural networks. The structure of the resulting encoded event log would remain the same: each trace (or often subtrace, a.k.a., prefix) is represented by a new numerical vector. 35

3.2 A brief example of performances achieved for two different datasets regarding the anomaly detection problem. For both datasets, we fixed two graph-based encoding methods with different parametrizations regarding dimensionality. 37

3.3 Performance metrics (one for row) for different vector sizes throughout all the employed scenarios (one for column). 48

3.4 (a) Overall aggregation of edges and nodes. (b) Specific aggregation of edges and nodes. (c) Aggregation strategies for edges. . . . 50

3.5 Distribution of metrics for aggregations of word-based encoding methods. 51

3.6 Distribution of performances for each anomaly type. 52

3.7 Overall performances for each anomaly type and grouped by algorithm families. 52

3.8 Performances by scenario. 53

3.9 Performances (rows) for each scenario (columns) according to different event log sizes. 54

3.10 Predictive performance of remaining time according to different combinations of preprocessing methods for each learning algorithm (row) and event log (column). 57

3.11 Top 10 averaged feature (x-axis) importance (y-axis) for all event logs. *n2v* refers to the nth-dimension of node2vec, *ac.time* to accumulated time, and *exec.time* to execution time. The numerical suffixes for time features refer to the index of a statistical measure (described in our repository). 58
3.12 Getting insights from the most important time-related feature: (a) number of events that occurred within each hour of a day; and (b) remaining time (RT) distribution for each hour of the day. 59

4.1 Proposed CoSMo framework (left) and the proposed recurrent architecture (right) tailored to capture the relation between user-based constraints and processes. 68
4.2 Weighted precision for the simulated logs under the different approaches and DECLARE templates. 76
4.3 Analysis of event log constraints from two perspectives: (a) count per log across templates, (b) distribution per template. 78
4.4 Data-flow of the proposed CoSMo framework. 78

5.1 Time and memory costs across different event log sizes (1k, 5k, and 10k) for both encoding families. 86
5.2 Ranking of Time Scalability, Memory Scalability, Time Consumption, and Memory Consumption. The better approaches are positioned in the first ranking position, colored by white. The most costly and least scalable are the last potions with dark colors. . . 87
5.3 Overall performances for each anomaly type and grouped by algorithm families. 89
5.4 Execution time using *n2v+ALL* pipeline against the complete LSTM pipeline. 90
5.5 Post-hoc Nemenyi test of both preprocessing and training model. Groups that are not significantly different (p=0.05 and CD=1.62) are connected. 90
5.6 Figure showing both modeling approaches (prefix and continuous) converge but the runtime for continuous modeling is much faster. 94

6.1 An example of our SkPM pipeline implemented extending the traditional Scikit-learn library. 102

7.1 Number of publications retrieved over the past 10 years. 116
7.2 Flowchart example of a ServiceNow incident. An incident is composed of one or more tasks, and the solution might be reached by combining solutions from several support groups. Illustration adapted from the ServiceNow documentation. 120

7.3 The raw data fetched from ServiceNow (left) and the preprocessed data (right). 126
7.4 A tiny piece (for better visualization) from the discovered spaghetti process model. 128
7.5 Execution time of intents. 129
7.6 Group falling within loops 130
7.7 Elapsed time with and without loops. 131
7.8 Variant distribution. 131
7.9 Await time according to each variant frequency group. 132
7.10 Enhanced process model. 133
7.11 Intent labels originally inferred by the LLM. Similar intents were obtained, such as 'Network connectivity issue' and 'Network access issue'. 138
7.12 Intent labels after clustering their embeddings. 139
7.13 Conclusions derived by the LLM. 139
7.14 t-SNE plot of conclusion embeddings. 140
7.15 Top 8 largest ETI values. The intents on y-axis are sorted by their average execution time. 142
7.16 Top support groups and their most frequent intents. The frequency of assignments for each support group (y-axis) according to the intent (x-axis). 143
7.17 The two most frequent variants derived from our event log. 147

List of Tables

3.1 Overview of five process models. For each scenario, we generated event logs by combining three different cardinalities, injecting seven different anomalies, at four different rates of injection (5%, 10%, 15%, and 20%). This resulted in 84 event logs for each scenario and 420 event logs in total. *gw*, *acts*, *evts*, and *vars* stand for the number of gateways, activities, events, and variants, respectively. The increasing complexity from scenario 1 to scenario is intuitively thought as the number of gateways and activities present in the log. 39
3.2 Anomalies used to simulate the real-life event logs. 40
3.3 Encoding methods and related details. 41
3.4 Employed configurations for word- and graph-based encoding methods. Refer to our paper [153] for a detailed description of each encoding technique. 48
3.5 Acronym and a brief description of each statistical measure employed in this work. 55
3.6 RMSE scores, in days, for each model and each event log. 57

4.1 LTLf semantics and textual description of the declare templates. . 67
4.2 Event logs and their properties: number of unique activities, number of events, number of cases, and average case length. 72
4.3 DECLARE templates and their respective activities employed to perform the conditioned simulation. 74
4.4 Precision for each class of rules. 77

7.1 Versions of ITIL. 109
7.2 The ServiceNow terminology aligned with PM terminology. Incidents can be seen as cases and Incident Tasks as events. 119
7.3 Incident task attributes. Translating into PM terminology, this refers to event attributes. 121
7.4 Two real incidents after all the preprocessing steps. 127

7.5 Conclusion count regarding the most frequent support groups and intents. 143
7.6 The most frequent support groups and their ratio of wrong assignments. SG stands for support group and WR for wrong reassignments. 144
7.7 An example of an incident where the support group "AvioAero CAE/CAT" is involved. This support group has the highest ratio of wrong (re)assignments. The incident's intent is the 'ANSYS Tool Suite installation issue'. 145
7.8 The top three variants, each accompanied by a list of their associated intents and the corresponding frequency of occurrence. . . . 146

Acronyms

LTL$_f$ Linear Temporal Logic over finite traces

AI Artificial Intelligence

AMS Application Management Services

API Application Programming Interface

AutoML Automated machine learning

BPM Business Process Management

BPMN Business Process Model Notation

CNN Convolutional Neural Network

DD-S Data-driven Simulation

DL Deep Learning

DL-S Deep Learning-based Simulation

EKG Event Knowledge Graph

FNN Feedforward Neural Network

GRU Gated Recurrent Units

IPA Intelligent Process Automation

IT Information Technology

ITIL Information Technology Infrastructure Library

ITSM IT Service Management

KNN k-Nearest Neighbors

KPI Key Performance Indicators

LLM Large Language Model

LSTM Long Short-Term Memory

MAE Mean Absolute Error

ML Machine Learning

MLP Multi-Layer Perceptron

NLP Natural Language Processing

OCEL Object-Centric Event Log

PCA Principal Component Analysis

PD Process Discovery

PM Process Mining

PPM Predictive Process Monitoring

PrPM Prescriptive Process Monitoring

PSM Process Simulation Models

RF Random Forest

RMSE Root Mean Squared Error

RNN Recurrent Neural Network

RPA Robotic Process Automation

SLA Service Level Agreements

SOTA state-of-the-art

SVM Support Vector Machine

XAI Explainable Artificial Intelligence

Chapter 1

Introduction

Process Mining (PM) is a research field that emerged in the late 1990s [164]. Its main goal is to leverage data logged by information systems to draw insights and propose enhancements in the organization's process flow. It is essentially composed of tools developed in a data-driven way to analyze and understand business processes. PM focuses on analyzing these logs that record transactions within an organization, thereby exposing how processes are truly executed. This is especially beneficial in complex environments where business rules may not be clearly documented, allowing PM to highlight discrepancies between what is documented and what is actually practiced. In practice, process data is represented by traces, which record sequences of events executed over time that are composed of alternative attributes, such as activity labels and timestamps.

Originally, the overall idea behind PM could be summarized as a pipeline for (i) discovering process models that reflect true process behaviors; (ii) evaluating the accuracy of these models against actual data to check if they are conformant; and (iii) using all the gathered insights to enhance and optimize the processes. These steps aim to characterize processes to identify and overcome limitations, eliminate redundant transactions, and suggest improvements to enhance overall efficiency. By providing this robust set of analytical tools, PM enables researchers and practitioners to gain deep insights into organizational processes, diagnose inefficiencies, and propose improvements.

This field has gone through significant advances in recent years, with the emergence of new branches that extend beyond this traditional PM pipeline. Among these emerging areas, Predictive Process Monitoring (PPM) has gained considerable attention within the academic and industrial communities due to the extensive advancements in Artificial Intelligence (AI)-based solutions. The large amount of research in this domain has led to a huge increase in publications, reflecting the growing interest and potential impact of PPM on process analysis and optimization. PPM applications aim to predict the future behaviors of on-

going process executions by, for instance, forecasting the next steps or classifying the final process outcome [127, 158].

Another branch example besides PPM includes the related field Prescriptive Process Monitoring (PrPM) [157]. While both share the common goal of anticipating future process states, they differ in their specific approaches. PrPM seeks to identify potential issues before they occur and provide suitable recommendations to mitigate risks. In contrast, PPM focuses on predicting future behaviors and outcomes without necessarily prescribing specific actions.

Accordingly, Process Simulation Models (PSM) represents another crucial extension of process analysis that bridges the gap between process understanding and improvement [161]. Unlike PPM and PrPM, which work primarily with historical data, simulation allows organizations to test 'what-if' scenarios in a risk-free environment before implementing changes in real-world processes. By creating simulated representations of business processes, companies can experiment with different process designs, resource allocations, and handling strategies to identify bottlenecks and optimize performance. This approach is particularly valuable when historical data is limited or when testing process changes in real environments would be too costly or disruptive.

All the mentioned branches so far share the common characteristic of being able to use AI-based technologies. Additionally, the intersection of these approaches with Explainable Artificial Intelligence (XAI) has become an area of particular interest. As the complexity of predictive models increases, the need for transparency and interpretability in their decision-making processes becomes fundamental. XAI techniques clarify the reasoning behind predictions, which enables stakeholders to understand and trust the insights generated by these PPM systems. This explanatory capability not only enhances the confidence of predictions but also facilitates the development and enhancement of preventive measures.

The growth of PM and its branches has brought attention to an often overlooked aspect in this field: computational costs. Datasets in PM, also known as event logs, grow larger and more complex (e.g., BPIC19 [1] having 20x more process instances than BPIC12 [2]), making the required resources to process them increase accordingly. This presents a significant challenge for organizations implementing these techniques, especially those with limited computing infrastructure. Moreover, real-time analysis, which many businesses now require for timely decision-making, further intensifies these computational challenges. Researchers and practitioners must, therefore balance the desire for more sophisticated and accurate models with the practical constraints of available computing resources.

[1] https://doi.org/10.4121/uuid:d06aff4b-79f0-45e6-8ec8-e19730c248f1

[2] 10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f

As the field continues to evolve, finding ways to reduce computational overhead while maintaining or improving analytical capabilities will tend to become a key focus area.

It is worth noting that PM by itself is a relatively young research field, which makes PPM and related fields even younger. This youthful aspect has led to several AI-based proposals characterized by extensive empirical experimentation. Researchers and practitioners strive to identify the most effective analytical approaches for diverse process datasets but always focus on effective outcomes for decision-making. The natural heterogeneity of processes across different organizations presents a unique challenge, as each dataset captures distinct business rules and operational patterns. Consequently, there is no superior learning technique applicable to all process mining scenarios, making a broader exploration of artificial intelligence methodologies from various problem domains necessary, such as Natural Language Processing (NLP), computer vision, and time series analysis.

1.1 State-of-the-art

The majority of proposals from the literature in PPM focus mainly on predictive accuracy [127] and explainability [57], often leveraging sequence modeling and encoding techniques from other domains. These models have demonstrated remarkable performance for tasks such as next activity prediction [155] and suffix prediction [182]. Regarding sequential modeling, techniques including recurrent networks and transformer-based architectures have been widely employed to develop learned-based solutions tailored to PM. Additionally, the literature presents strategies for building training datasets from event logs. The most common approach consists of creating prefixes [155, 54, 102, 27], where shorter sequences of events are extracted from their respective traces. Alternatively, the continuous sequence approach [52, 71] consists of flattening the whole event log, treating all the processes as a long piece of text, and training models on chunks extracted from it. This approach is inspired by natural language processing techniques, arguing to simplify the integration and implementation of PM and NLP solutions. In practice, prefixes and chunks are identical, but the latter disregards the trace perspective and allows the overlapping of different process executions.

Building these datasets is an important preprocessing step, and recent research has shown that the prefix approach works better in general [127]. Additionally, alternative preprocessing steps include choosing a suitable encoding technique to represent process data in latent spaces. These encoded representations are essential for applying traditional Machine Learning (ML) algorithms [173, 127] and clustering methods [68, 184].

Process data can be encoded at different levels: **event-level** encoding focuses on representing event attributes using methods such as one-hot encoding [52], frequency-based approaches [55, 90], or techniques tailored to process data [79, 29]; **trace-level** encoding captures information at the intra-case level, where traces are individually encoded [76, 184, 97, 143, 122]; **relationship-level** encoding considers relationships at an inter-case level between traces executed concurrently, such as resource sharing [144, 145]; finally, the **log-level** and **process model-level** encoding transforms entire event logs or process models into vector representations [83].

Most existing research simplifies the problem by focusing on event- and trace-level encoding, although the inter-case aspect is crucial in real-life scenarios. The dominant techniques include one-hot encoding for categorical attributes at the event level and simple aggregation methods, such as averaging or summing traces of encoded events, at the trace level [157]. While these methods are popular, they are inherently limited: they lead to significant information loss and high dimensionality when dealing with large categorical variables, and fail to account for parallel processes common in real-world scenarios.

Recent advances have addressed these drawbacks. For example, Kim et al. [79] introduces a framework for resource-aware feature generation that derives numerical metrics, including task frequencies, performance ratios, specialization levels, and workload intensity, from event logs. This approach moves beyond simplistic one-hot encoding, providing meaningful insights into resource behavior. Similarly, feature generation or engineering tailored specifically to process mining has begun to gain attention [29, 111]. Furthermore, Koninck et al. [83] present neural network-based representation learning techniques, such as act2vec, trace2vec, log2vec, and model2vec. These methods aim to generate informative embeddings for process elements at different levels, enabling their use in advanced tasks like clustering and predictive monitoring. Trace-level encoding offers a wider range of solutions and has been the focus of more extensive exploration in recent surveys [158, 153].

The data representation aspect in PM is crucial, and it significantly affects the performances of predictive monitoring tasks [158, 173, 153]. Although the next activity and suffix prediction are important tasks, a more realistic and impactful task regards an extension of both: the process simulation. Originally, process simulation was performed by heuristic-based approaches in a data-driven fashion [161, 28]. Recently, learned-based solutions have been proposed due to the outstanding performances of Deep Learning (DL) models for the aforementioned tasks. However, these solutions involve fully sampling the next events' attributes from the trained model to simulate a process, resulting in a fully stochastic process [28]. This inflexibility of deep learning models is the major limitation for developing learned-based process simulation models.

This drawback is being overcome in the literature by exploring representation and encoding techniques for processing data. Three very similar papers have been recently proposed in the literature to mitigate the natural stochasticity and inflexibility for what-if scenarios via learned-based simulation models. Barón-Espitia et al. [9] present a declarative method for scenario generation by modifying control-flow rules through DECLARE constraints. This approach facilitates targeted what-if analysis while minimizing deadlock risks using LSTM-based trace generation. Similarly, in [112], we propose CoSMo, a framework that designs a new conditioned recurrent neural network architecture to incorporate any user-defined constraints. CoSMo demonstrates its effectiveness by integrating DECLARE constraints as well and offers flexibility across both control-flow and data-flow perspectives. In contrast, Graziosi et al. [64] adopt conditional variational autoencoders to produce traces conditioned on binary constraints. This ensures diversity in the generated traces while adhering to process rules. While Graziosi et al. [64] focuses on enhancing generative control using binary constraints and Barón-Espitia et al. [9] emphasizes constraint flexibility using DECLARE constraints, Oyamada et al. [112] cover all these aspects and additionally provides adaptability to any user-defined constraint, including those specified via DECLARE constraints, achieving broader flexibility across constraint types and process dimensions. These contributions collectively advance the field by improving the adaptability and robustness of simulation models through different data and constraint representation approaches, reinforcing the importance of encoding process data aspects for practical and effective applications.

1.2 Problem Definition

In this thesis, we address several open challenges at the intersection of PM and AI, with a particular focus on advancing the field of data representation, PPM, and process simulation. We aim to not only contribute with novel theoretical frameworks and methodological approaches but also demonstrate their practical applicability through a real-world implementation at Avio Aero, an industrial partner who partially funded this work.

The rapid convergence of PM and AI has created both opportunities and challenges in recent years. While this intersection has created numerous innovations, it has also revealed fundamental questions about the appropriate adaptation of AI techniques to process-oriented domains. The field of AI is composed of various other disciplines, fundamentally grounded in statistics, mathematics, and computer science. Specialized domains within AI, such as computer vision and natural language processing, have developed domain-specific architectures and methodologies that explicitly account for their unique data nature. For instance,

computer vision has evolved to incorporate deep learning architectures specifically designed for spatial data structures (e.g., convolutional networks), while natural language processing has developed models optimized for sequential linguistic patterns (e.g., recurrent networks).

However, the relatively embryonic field of PM has been fundamentally developed by importing AI solutions from these other domains without sufficient adaptation to process-specific characteristics [78]. Although the community has witnessed great achievements, PM has faced extensive empirical research, which might slow down and undermine the progress in this field [70]. By process-specific characteristics, we essentially mean:

- **Temporal Dimension**: Process data exhibits complex temporal patterns, including both sequential dependencies and parallel execution paths [164]. Unlike simple time series data, process events often display intricate temporal relationships that require specialized modeling approaches.

- **Relational Dimension**: Processes involve resource dependencies and constraints that create complex relationships between events. These relationships are grounded by operational system principles, where event execution is contingent upon resource availability and proper synchronization [90, 59]. Classical challenges in this domain, such as deadlock and livelock conditions, should require explicit consideration in any AI-based solution.

- **Semantic Dimension**: Business processes are driven by domain-specific rules and constraints that define valid execution patterns [121, 56]. These semantic constraints, such as mandatory activity sequences or mutual exclusivity rules, form an essential part of the process context that must be incorporated into any analytical framework.

The overreliance on empirical research, leading to the arbitrary application of AI techniques originally developed for other domains, often fails to adequately address these three dimensions. For example, while convolutional neural networks excel at processing spatially structured data, and recurrent neural networks are designed for sequential data, neither architecture inherently takes into consideration the complex interplay of temporal, relational, and semantic aspects that characterize process data. Attention mechanisms are, theoretically, very promising in this regard due to their capability of capturing inner relationships from sequences [170] or between sequences and external data [92]. Graph neural networks recently emerged as a promising solution since process data is naturally modeled as complex networks [167, 59, 148, 114]. However, there are several strategies to model process data as graphs, and the current literature lack a systematic review to compare which graph modeling technique works better with

graph neural networks. Overall, many proposed applications in the PM literature have achieved remarkable results so far, but there is no consensus of which architecture is better.

In parallel to the reliance on well-established learning methods to cover all the aforementioned process-specific characteristics, researchers have also put efforts on the preprocessing phase. The idea is to transform the data in a way that simple learning models could capture all the particularities of process data. For instance, the well-known index encoding [90] encodes business process traces as complex symbolic sequences. It integrates static case attributes and dynamic event attributes, using index-based and Hidden Markov Model (HMM)-based encodings to preserve control-flow and data-flow relationships. However, while the HMM-based encoding captures attribute evolution over time, it assumes that each attribute follows its own independent sequence, missing potential interdependencies between different attributes that could influence the prediction. On the other hand, the aggregation technique [55] that simply averages all the features, without making the relational dependencies as well, tends to perform better than any other encoding technique from the literature [158]. This demonstrates that although efforts have been put into addressing all these challenges, there is still room for improvement and development.

Alternative challenges in PM research include the limited efforts on reproducibility, which limits the ability to verify and build upon results. Reproducibility in research requires more than just public repositories; it involves clear documentation, accessible data, strict dependency management to replicate environments, and open-source packages centralizing state-of-the-art (SOTA) implementations. This practice makes it easier for others to replicate and validate findings [123]. Furthermore, the high computational demands of AI-based solutions raise environmental concerns since extensive training processes consume significant energy and resources. Addressing these challenges requires not only technical refinement but also a commitment to sustainability and open practices within the PM community.

1.3 Research Questions

Considering the aforementioned challenges, this thesis aims to advance the field of PPM by developing novel AI architectures and methodologies that address the unique characteristics of process data and the challenges and concerns derived from it. Three critical challenges exemplify this need: Firstly, the fundamental challenge of representing event logs as numerical vectors for AI model training still exists, and there is room for innovation and improvement. Current approaches often default to one-hot encoding or general sequence modeling without

adequately capturing the temporal, relational, and semantic dimensions of process data discussed above. Moreover, feature engineering techniques for PM have been studied, but we show that there is still a lot to be explored and expanded. Secondly, process simulation, despite its importance for business process optimization, has struggled to effectively incorporate AI techniques due to a lack of balance between the deterministic requirements and the stochastic nature of simulation models. Such determinism is covered by the existing heuristic-based methods, whereas AI-based methods struggle with such flexibility. This limitation significantly impacts stakeholders who need to accurately estimate the impact of proposed process modifications while keeping control over specific simulation parameters. Lastly, we provide an in-depth discussion of the current literature, indicating limitations and flaws, and propose an open-source package to support the development of future PPM applications. By centralizing existing methods through a user-friendly tool based on a well-known machine learning library, we intend to accelerate the development of new methodologies.

In summary, we aim to answer the following research questions:

- **RQ1**: *Given the complex temporal, relational, and semantic dimensions of process data, what are the most effective methods for encoding and representing this data in latent spaces, and how do these methods perform across different process mining tasks?*
- **RQ2**: *How can modern deep learning techniques enhance traditional heuristic based process simulation to improve the accuracy and reliability of what-if analysis?*
- **RQ3**: *How do the complex temporal, relational, and semantic patterns in process data impact an organization's ability to leverage Process Mining solutions for optimizing processes and addressing environmental challenges, such as energy consumption?*
- **RQ4**: *What are the advantages, disadvantages, and practical challenges of integrating process mining and deep learning in an industrial context at Avio Aero, and how can these technologies be effectively combined to address real-world process optimisation problems?*

1.4 Contribution

This thesis addresses these fundamental challenges through several novel contributions, including: (i) an extensive and systematic evaluation of encoding

techniques from other literature to showcase that arbitrary choices lead to suboptimal results; (ii) adaptive architectures that bridge the gap between stochastic AI models and deterministic simulation requirements; (iii) a methodology and new proposals for different PPM tasks that take into consideration environmental concerns; and (iv) a practical framework that demonstrates the real-world applicability of these theoretical advances. Together, these contributions advance the state of the art in PPM while establishing new methodological foundations for future research at the intersection of process mining and artificial intelligence.

Furthermore, in order to reinforce our achievements, our solutions are developed through a systematic methodology that aligns academic research with Avio Aero requirements. This ensures that our theoretical contributions maintain practical relevance while advancing the fundamental understanding of process mining challenges. We start by addressing the foundational challenge of process data representation, conducting a comprehensive evaluation of existing methods for mapping process data into numerical latent spaces. This evaluation systematically evaluates existing embedding techniques through the lens of process-specific requirements, considering how well they preserve domain-specific relationships and encode semantic business rules. Additionally, we also explore how to enhance the process data representation by implementing novel feature engineering techniques for process mining. Moreover, we cover the environmental impact of learned solutions in our experiments in order to align with large industries' concerns. This work forms the basis for our subsequent contributions to new architectures for process simulation and practical frameworks for PPM pipelines, ensuring that our solutions maintain both theoretical rigor and practical utility throughout the research pipeline.

1.5 Outline

Regarding the remainder of this document, chapter 2 presents all the necessary technical details for the understanding of this thesis. In some Chapters, some more nuanced aspects are also introduced when necessary for the understanding of the specific Chapter. chapter 3 presents a discussion on process representation for two different tasks. chapter 4 introduces a new architecture tailored to learn process constraints in order to perform reliable process simulation. chapter 5 extends the findings so far by addressing environmental concerns on the development of AI-based solutions in PM. chapter 6 presents foundations of AI for PM in order to enhance the development of new solutions and technologies. Finally, chapter 7 presents the industrial results accomplished at Avio Aero during the one-year internship, and we conclude the thesis in chapter 8.

Chapter 2

Background

As initially introduced, the nature of PM research covers a wide range of techniques and methodologies: from the application of traditional automata theory in discovering process models based on Petri nets [167] to the application of cutting-edge deep learning approaches such as a Large Language Model (LLM) [15]. Therefore, in this Section, we provide a comprehensive and narrowed presentation of essential techniques for the understanding of this thesis. This roughly includes an introduction of Process Discovery (PD) algorithms, with a particular focus on declarative models, and a review of various PPM tasks.

2.1 Process Mining

PM [162] is a powerful approach to analyzing and improving business processes by leveraging event data generated by information systems. Essentially, PM relies on **event logs**, which record sequences of activities performed during process executions and capture all the actual behavior of processes. Process discovery algorithms form a crucial component in this research field, aiming to automatically discover **process models** from event logs. These algorithms aim to abstract and represent the underlying structure of processes [164]. Together, these techniques provide a data-driven foundation for understanding, monitoring, and enhancing complex business processes across various domains [48].

Definition 1 (Event log, case, trace.)**.** *An event log L consists of a set of cases Q (a.k.a. process executions). Each case $q_i \in Q$ is composed of, essentially, a trace of events t_i and a case identifier i. A trace is composed of a finite ordered sequence of s events and each event $e_{i,j}$ refers to the j-th execution, for $0 \leq j \leq s$, of a system activity in the trace t_i and is characterized by a set of alternative attributes.*

Definition 2 (Events.). *The j-th event from the i-th case can be represented as a tuple* $e_{i,j} = (a, time, \mathcal{F})$*, where* $a \in \mathcal{A}$ *is the activity label from the set of activities present in L, time is the timestamp denoting when the activity was executed, and* $\mathcal{F} = \langle f_0, \ldots, f_z \rangle$ *are alternative feature values, with* $z \geq 0$*.*

In some applications, the notion of case and event features is disregarded and used as defined above. In this thesis, we take into consideration both versions. Thus, let us define this notion subsequently.

Definition 3 (Alternative case and event features). *Given a trace t, a case feature* $cf \in \mathcal{CF}$ *is a static feature as it maintains the same value for all events* $e \in t$*. On the other hand, an event feature* $ef \in \mathcal{EF}$ *is dynamic, varying its value across events in the trace. Thus, the event definition can be rewritten as the tuple* $e = (i, a, time, \mathcal{CF}, \mathcal{EF})$*, where* $\mathcal{CF} = (cf_1, v_1), \ldots, (cf_{z_1}, v_{z_1})$ *and* $\mathcal{EF} = (ef_1, v_1), \ldots, (ef_{z_2}, v_{z_2})$*, with* $z_1 \geq 0$ *and* $z_2 \geq 0$*. Hence,* $\mathcal{F} = \mathcal{CF} \cup \mathcal{EF}$*.*

An example of a case feature includes the personal data of a client (for instance, their name does not change throughout the process); whereas an event feature can be the cost to execute its respective activity (each activity might have a different cost). Note that the case identifier can be seen as a case feature, whereas the activity label and the time stamp as event features, but all of them are mandatory to define a process event. Finally, a final notion to be presented here is the control- and data-flow: the former regards the flow of activity sequences only, whereas the latter takes into consideration the remaining features composing the events as well.

Recently, several new log standards, including the Object-Centric Event Log (OCEL) [59] and Event Knowledge Graph (EKG) [53], have been proposed for reading and managing process events. However, formal definitions of these standards are not provided in this thesis as they are out of scope here.

A process model is an abstract representation capturing the sequence and interrelation of activities in historical process executions recorded in an event log [167]. The process discovery algorithms that construct such models from event logs can be classified as either imperative or declarative [164, 121]. Imperative models explicitly define the possible paths through a process. In contrast, declarative models focus on specifying the constraints or rules that capture the process behavior rather than explicitly defining all possible paths.

Definition 4 (Process discovery, process model.). *A process discovery algorithm* PD *constructs a process model* N *from an event log and can thus be seen as a function* $\delta : PD(L, \alpha) \rightarrow N$*, where* α *is a set of hyperparameters to configure the corresponding* PD *algorithm.*

Petri nets [167] and Business Process Model Notation (BPMN) [174] are graphical representations for imperative process models. The former offers a mathematically rigorous framework that supports various analysis techniques, enabling verification of properties like reachability, liveness, and soundness [165]. On the other hand, the latter provides a more visually intuitive approach to process modeling and it offers a rich set of graphical elements to represent process flow, including events and gateways [180]. The choice between them often depends on the specific needs of the project, the intended audience, and the level of formal analysis required. This is due to the fact that although both Petri nets and BPMN aim to represent process behavior, they serve different purposes in practice. Petri nets are often favored in academic and research contexts for their formal properties and analytical capabilities. BPMN, with its richer visual vocabulary and industry acceptance, is widely used in business environments for process documentation, communication, and implementation.

Discovery algorithms of imperative process models include, for instance, the α-miner, pioneered by Aalst et al. [164]. It was one of the first to automatically construct Petri nets from event logs. The Inductive miner, a more advanced technique, offers greater flexibility by discovering either Petri nets or BPMN models. It employs a divide-and-conquer approach, recursively breaking down the log to identify key process structures. In this thesis, we do not dive into more details of imperative models, as they will be discussed only at a high level. However, we do provide an in-depth introduction of declarative models in the following.

From a discovered process model and the actual event data, it is possible to design new enhancements. Process enhancement [48] focuses on refining existing operations by using the insights derived from the process model. Accordingly, the original data reveal how processes actually work, often uncovering gaps between the intended and observed behavior. Comparing these insights with the reference discovered model highlights inefficiencies, such as delays, unnecessary transitions, or quality issues. Thus, adjustments can be made to the process logic or the systems supporting it. Furthermore, this model has recently become crucial for the development of learned solutions in PM, as we will describe in Section 4.1a.

In the remaining Chapter, we will introduce PPM tasks. Let us first introduce the basic concepts of machine learning.

2.2 Traditional Machine Learning

ML focuses on developing algorithms capable of learning patterns from data and making predictions. At its core, ML relies on statistical techniques to uncover relationships between input features and target variables [5]. A fundamental

concept in ML is the independent and identically distributed (i.i.d.) random variables. This assumption forms the basis of many ML algorithms and facilitates statistical inference.

Definition 5 (i.i.d.)**.** *A sequence of random variables $X_1, X_2, \ldots, X_n$ is considered i.i.d. if (i) each variable has the same probability distribution as the others and (ii) all variables are mutually independent.*

In supervised learning, the goal is to learn a model (a.k.a., mapping function) $f : X \rightarrow Y$, where X represents the input feature space and Y is the target space. Given a dataset $D = (x_1, y_1), \ldots, (x_n, y_n)$, where $x_i \in X$ and $y_i \in Y$, the objective is to find a function f that minimizes a predefined loss function $L(f(x), y)$ over the entire dataset. The prediction, or inference, involves applying the trained function f to new, unseen data points x' to estimate their corresponding y' values. However, different ML tasks are characterized by the nature of this target variable Y. We describe below the tasks that are employed throughout this thesis:

- Binary classification: $Y \in 0, 1$, the model predicts one of two classes.
- Multi-class classification: $Y \in 1, \ldots, k$, where $k \geq 2$. The model assigns inputs to one of K mutually exclusive classes.
- Regression: $Y \in \mathbb{R}$. The model predicts a continuous numerical value.

Beyond these traditional tasks, there are several ML paradigms that attempt to learn beyond the target Y. For instance, the meta-learning paradigm aims to learn how to learn [169], unsupervised learning learns from unlabeled data [8], self-supervised learning learns one part of the input from another part of the input [42], etc. In this thesis, however, we eventually employ the multi-target learning paradigm so we define it subsequently.

Definition 6 (Multi-target learning)**.** *It can be defined as the function that maps X to Y such as $Y \in \mathbb{R}^d \cup \{\langle 1, \ldots, k_1 \rangle, \ldots, \langle 1, \ldots, k_2 \rangle\}\rangle$.*

In the multi-target setup, the model simultaneously predicts multiple targets, which can be either continuous (multi-output regression) or discrete (multi-class classification). Traditional ML algorithms, such as decision trees, are not suitable for this nature of problem. Usually, one model is trained separately for each of these targets.

Another limitation of traditional ML algorithms is the reliance on i.i.d. data due to its mutual independence expectation. In process mining, for instance, data is sequential so the next step will always rely on the previous ones. There are strategies in ML to overcome this limitation, like aggregating all the previous

steps into one single array vector [158], but this usually leads to information loss [49]. On the other hand, in deep learning there are tailored architectures to capture this sequential relationship by reducing the information loss throughout the sequence.

2.3 Basic Concepts of Deep Learning for Generative Tasks

DL is a subset of ML that employs artificial neural networks with multiple layers to learn hierarchical representations of data [17]. These networks are capable of automatically extracting features from raw input, often outperforming traditional ML algorithms on complex tasks. The main types of deep learning architectures include: Feedforward Neural Network (FNN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Transformer Networks. In practical terms, all these architectures can be designed or adapted to learn from process data. However, in this thesis, only RNNs are employed since they outperform all the other ones, specifically in the process mining domain.

Given the sequential nature of process data, RNNs are particularly relevant for PPM tasks since they are designed to process sequential data by maintaining an internal state that captures information from previous time steps. Formally, an RNN can be defined as follows:

$$h_t = f(W_h h_{t-1} + W_x x_t + b_h) \tag{2.1}$$

$$y_t = g(W_y h_t + b_y) \tag{2.2}$$

Where x_t, h_t, and y_t are the inputs, hidden states, and outputs at time step t, respectively; W_h, W_x, W_y, b_h, b_y are the parameters to be learned in the form of weights and biases; finally, f and g are activation functions. The intuition behind these equations is to capture the sequential relationships throughout the hidden state h_t. Thus, at each time step t, the hidden state will take into consideration all the previous steps in order to generate the next h_{t+1} and to infer y_{t+1} from it.

RNNs can process sequences of varying lengths, making them suitable for analyzing event logs where cases may have different numbers of events. However, basic RNNs often struggle with long-term dependencies due to the vanishing gradient problem. To address this limitation, researchers frequently employ more advanced RNN variants. Examples include the Long Short-Term Memory (LSTM) [72], which better captures such long-term dependencies by adding more parameters to be learned; and Gated Recurrent Units (GRU) [38], which

offers a simplified gating mechanism compared to LSTMs, often achieving similar performance with fewer parameters.

This natural ability to model sequential dependencies makes RNNs and their variants particularly well-suited for PPM tasks [127]. They can effectively capture the complex temporal patterns and inter-event relationships present in process event logs, leading to more accurate predictions of future process behaviors, outcomes, or performance metrics. In contrast, applying traditional ML methods to such sequential data often requires extensive feature engineering to capture temporal aspects, such as extraction of business features [26] or aggregating historical information [158]. While these approaches can be effective to some extent, they often fall short in capturing the full complexity of sequential dependencies that RNNs can model naturally.

Furthermore, although attention mechanisms and Transformer networks have recently revolutionized the field of DL and NLP [170], their outstanding advantages include the capability to parallelize training and to compute how much attention each input x_t should pay to all other inputs in the sequence to output its predictions. Roughly speaking, this ability allows the model to calculate attention scores between each time step and all other steps, enabling it to identify the most important elements for each position individually rather than summarizing all information into a single hidden state. Consequently, a different set of attention scores is generated for each time step.

In practice, such models are often used for generative purposes, which characterizes an autoregressive task. In autoregressive tasks for next element prediction, the goal is to model the probability of each element in a sequence based on all the preceding elements. Formally, let $X = (x_1, x_2, \ldots, x_T)$ be a sequence of elements. The conditional probability for the next element prediction is defined as $P(x_t \mid x_1, x_2, \ldots, x_{t-1})$. This represents the probability of the element x_t occurring at position t given the sequence of preceding elements $x_{<t} = (x_1, x_2, \ldots, x_{t-1})$. The model aims to learn this conditional distribution for all positions t in the sequence.

Definition 7 (Autoregressive task)**.** *The autoregressive task for generative purposes can be defined as the joint probability of the entire sequence X decomposed into the product of the conditional probabilities: $P(X) = \prod_{t=1}^{T} P(x_t \mid x_{<t})$.*

Autoregressively predicting event data is an open challenge in the literature and consists of predicting multiple event attributes at each step [155].

2.4 Predictive Process Monitoring

PPM is a subfield of PM and aims to predict future process behaviors by leveraging advanced artificial intelligence techniques [36]. The field has experienced significant growth in recent years, driven by rapid advancements in AI research [10]. Traditionally, PPM tasks are categorized into next activity prediction, remaining time estimation, and outcome prediction, with suffix prediction often considered as an extension of next activity prediction.

2.4.1 Tasks

The traditional PPM tasks [127] can be categorized into three main types: next activity prediction (potentially extended to suffix prediction), remaining time estimation, and outcome prediction. Each of these tasks aims to estimate different aspects of an ongoing process instance.

Next activity prediction [155] focuses on estimating the activity that will be executed at time step $i + 1$, given an ongoing case at time step i. This task can be extended to suffix prediction [155, 156], which involves recursively predicting multiple future activities until the final stop criteria are reached. In machine learning terminology, this approach is known as autoregressive prediction, where each prediction serves as an input feature for subsequent predictions.

The remaining time prediction [106, 173] estimates the time required to complete a case from a given time step i. Note that the remaining time is an expected outcome at the current time step rather than the next attribute to be predicted. That means at the initial time step ($i = 0$), this value is not available as an input feature, but it can be used as such in autoregressive scenarios: common approaches include initializing it to zero or using other heuristics based on historical data. Moreover, instead of predicting the remaining time, other strategies include the prediction of the next timestamp. Thus, through post-hoc processing, the remaining time at each time step can be derived from the predicted timestamps. This is due to the fact that the remaining time can be defined as the difference between the last timestamp and the current one. The modeling approach here is a design choice and varies from each implementation.

Outcome prediction [158] aims to estimate the final result of a case. This can cover various aspects such as case success or failure, total cost, customer satisfaction ratings, or any other relevant process-specific outcomes. The nature of the outcome variable can vary depending on the specific business context and can be formulated either as a classification or a regression problem. Below, we formally define the traditional tasks based on [127, 158].

- Next activity prediction: $f : t_i \rightarrow \mathcal{A}$, where t_i represents the trace state at time step i, and A is the set of possible activities.

- Remaining time prediction: $f : t_i \rightarrow Y$, for $Y \subseteq \mathbb{R}^+$ being the output of a positive real number representing the estimated remaining time.
- Outcome prediction: $f : t_i \rightarrow Y$, where $Y \subseteq \mathbb{Z}$ is a set of finite discrete (categorical) values, t_i represents the final state of the process.
- Suffix prediction: $f : t_i \rightarrow t_{i+1:k}$, where k is the number of predicted steps to complete the ongoing prefix trace.

The suffix prediction is typically performed autoregressively: the current prediction is used as input for the next step. Although many researchers have used this strategy, few have explicitly defined their tasks as autoregressive [155, 54, 106, 93, 26]. Emphasizing the use of autoregressive multi-target prediction is important because it makes clear that the model is predicting several outputs step by step.

The intention of introducing this definition here is that it covers a broader range of tasks beyond suffix prediction only, such as event log generation [28] and process simulation [112]. Here, we assume the activity is a mandatory event attribute to be predicted since it is the cornerstone of process data. However, differently from the natural language processing domain, which aims to predict one-dimensional sequences (i.e., words or tokens), the autoregressive event prediction aims to predict the activity labels and additional event features. Some solutions in the literature, however, also focus on training one model for each attribute, characterizing a regression or multi-class classification problem, instead of training one model under the multi-target setup.

2.4.2 Event Data Encoding

Encoding is the task of transforming data from one domain to another [35]. For instance, one way of encoding categorical data is to transform categories into numerical values. In PPM, encoding is a term that sometimes is overused due to the fact it refers to encoding any sort of data, such as activity labels, sequences of events, encoding relationships between cases, etc. This overuse might be confusing and misleading sometimes, especially in this thesis since we cover most of these applications at different granularities. Therefore, let us clarify this extensive terminology.

- A **scalar** is a single numerical value, representing a quantity without any directional information.
- A **vector** is an ordered collection of numbers, which can be described as an n-dimensional object. Each element in a vector corresponds to a specific

dimension, allowing it to represent multiple features or properties simultaneously.

- A **tensor** is a generalized mathematical object that extends the concept of scalars, vectors, and matrices (two-dimensional vectors) to higher dimensions. A tensor can have one or more dimensions (or orders), where a 0th-order tensor is a scalar, a 1st-order tensor is a vector, a 2nd-order tensor is a matrix, and higher-order tensors represent more complex multidimensional arrays.

- A **sequence** is a more abstract and generalized concept, defined as an ordered list of elements. Each element in a sequence can be a scalar, vector, tensor, or other structured data type. Sequences are designed to capture the temporal or logical order of elements, making them suitable for representing data that unfolds over time.

Subsequently, we categorize and discuss each encoding technique in the context of process mining as follows.

Definition 8 (Event to vector (evt2vec))**.** *At the event-level, given an arbitrary event e^d containing d attributes, the event to vector is a function $e2v$ that transforms such d attributes into d' new attributes. Thus, $e2v(e^d) = e^{d'}$.*

The *evt2vec* can be thought of as a feature extraction step, and these terms are usually used interchangeably in the literature. For instance, activity labels are encoded using techniques like one-hot encoding to activity labels, and time-related features are extracted from timestamps. Thus, the $e_encoder$ function can be composed of one or more functions that will be applied over specific attributes. For instance, the one-hot for activities and extracting the weekday from a timestamp.

Definition 9 (Trace to sequence (trace2seq))**.** *At trace-level, a function mapping a trace $f : T^n \rightarrow S^m$, where T is a trace of length n and S is a (sub)sequence of length m, with $m \leq n$. If $m = n$ it means the trace is used as it is, otherwise, a prefix is extracted from it.*

The *trace2seq* encoding can be seen as a sequential modeling strategy to build the training dataset. Common approaches include slicing each trace into (sub) sequences, which are also known as prefixes. Another existing approach takes the whole event log as input and flattens it to split it into smaller sequences. However, this approach disregards the case perspective, and recent works have shown this approach does not work well [127]. Thus, the prefix modeling is the current state-of-the-art. On the other hand, encoding a sequence of events into numerical representation requires an extra step.

Definition 10 (Sequence to vector (seq2vec))**.** *A function* $f : S^d \to \mathbb{R}^{d'}$ *that maps a sequence* $S = \{x_0, x_1, \ldots, x_n\}$*, where each* $x_0 \in \mathbb{R}^d$ *represents an* d*-dimensional event, to a* d'*-dimensional vector* $v \in \mathbb{R}^{d'}$ *that encodes the information from the entire sequence.*

Common *seq2vec* approaches in the context of process mining include aggregation (e.g., averaging the whole sequence) and embedding (learned representations using neural nets). This technique links sequences and vectors by transforming the temporal information from a sequence into a static representation (vector). Such representations allow machine learning models to handle temporal data more effectively by summarizing the sequential patterns into fixed-size inputs.

Chapter 3

Process Data Representation

Process mining is an evolving field that bridges the gap between data science and Business Process Management (BPM) by analyzing event logs to gain insights into process performance. Despite significant advances, the representation of process data in a format suitable for machine learning algorithms remains an open challenge. Existing techniques predominantly rely on basic encoding methods, such as one-hot encoding and simple feature extraction from timestamps. A few feature extractors tailored for process mining have been proposed, but the literature also lacks a systematic comparison of how much each of these methods affects the predictive tasks. Nevertheless, all these existing methods, while easy to implement, fail to capture the complex, sequential nature of process data, limiting the predictive power of machine learning models in PPM. As a result, there is a pressing need for more sophisticated techniques to encode process data, especially in dynamic environments where rapid model updates and interpretability are essential.

This Chapter has two contributions. The first one aims to explore alternative encoding techniques drawn from various domains, including text- and graph-based representations, to improve the quality of process data embeddings. Second, it introduces new feature extraction methods tailored for time-related features and enhances the traditional *seq2vec* aggregation method to better represent sequential process data. By leveraging these innovations, we demonstrate in the first contribution that existing solutions can be leveraged in the PM domain; and second, we show how shallow learning algorithms can be made more efficient and effective, providing faster and more interpretable models for PPM tasks without sacrificing predictive performance.

We validate the proposed methodology using two different tasks: anomaly detection and remaining time prediction. For anomaly detection, we systematically evaluate several surveyed encoding techniques, aiming to identify which data characteristics and encoding hyperparameters have the greatest influence on

predictive performance. This comprehensive evaluation provides critical insights into how different representations impact model accuracy. For the remaining time prediction, we focus on developing and exploring more robust preprocessing techniques, demonstrating that these steps can significantly enhance performance without the computational overhead associated with deep learning models. This approach underscores the potential of efficient, shallow learning methods in predictive process monitoring tasks. Furthermore, the methods proposed in this Chapter were described in our previous works. The survey of encoding methods employed for anomaly detection was introduced in [153] and presented in section 3.2; whereas for the remaining time prediction task, we designed new feature engineering techniques tailored to process data in [111] and it is presented in section 3.3.

3.1 Present Context in Predictive Process Monitoring

As explained in subsection 2.4.2, encoding involves transforming data from one domain into another to make it suitable for specific computation, such as training ML models or clustering applications [68, 158]. This step is essential when working with process data because the data has multiple levels of granularity. Refer to Figure 3.1 for an illustration. In the example, we start with an event log consisting of traces, where each trace is a sequence of events. Although in real-life scenarios, events might contain several attributes, each event in this example contains three attributes: case identifier, activity, and timestamp. At the event-level encoding, we convert categorical values such as the activity label "A" into a numerical representation using one-hot encoding. For instance, "A" becomes the vector $\langle 1, 0, 0 \rangle$. Next, we handle timestamp data like "*2024.12.31 12:35*" by applying feature engineering to extract meaningful features such as the day of the week and the accumulated time over the ongoing trace. This transforms the timestamp into a new vector of numerical features. Encoding each event individually in this way produces a matrix (or tensor) where the rows represent events and are ordered by their timestamps. After encoding the events, we move to trace-level encoding. In this example, we aggregate the event-level tensor by computing the average of its rows [158], which results in a single vector representing the entire trace. This sequence of encoding steps ensures the process data is structured for machine learning or process mining tasks. This illustration shows one possible approach to encoding traces, but the literature also offers other methods to similarly encode traces either individually [153] or by considering their relationships [144].

The example presented, which once was the leading approach in the litera-

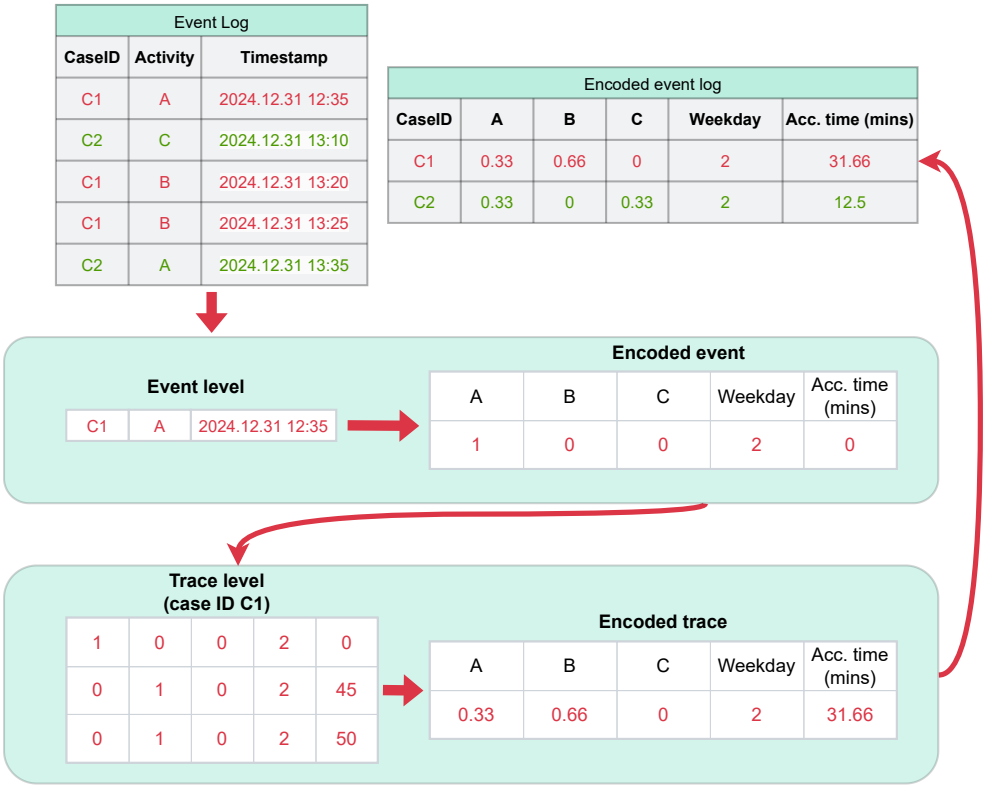


Figure 3.1: An example of how event logs can be encoded. Although the illustration uses specific techniques, such as one-hot encoding at the event-level and average aggregation at the trace-level, other techniques can be employed. For instance, learning embedded representations at the trace-level using deep neural networks. The structure of the resulting encoded event log would remain the same: each trace (or often subtrace, a.k.a., prefix) is represented by a new numerical vector.

ture [158], has limitations. Intuitively, consider the day of the week in the provided example: if an arbitrary trace had the following values, $\langle 4, 6, 0, 6, 1 \rangle$, where these numerical values illustrate Monday (0) to Sunday (6); simply averaging this array would not capture the circular aspect of days of the week. As a result, recent evidence has shown [83, 153] that learning-based methods for encoding process data often outperform approaches that rely on aggregating one-hot encoded data. This advantage comes from the ability of certain neural networks, such as recurrent models, to naturally capture the sequential relationships inherent in process data. In contrast, the intuition behind the static aggregation of process data into a single vector says it could result in information loss. Nevertheless, despite

the better performances of learned-based solutions, recent findings suggest that static aggregation can sometimes be competitive in terms of predictive accuracy while offering greater interpretability [57] and less computational costs [111].

Learning embeddings of traces also allows to transfer knowledge to other tasks [168], avoiding the costly and challenging process of training deep networks from scratch. Traditionally, these embeddings are learned from the flattened sequences of events ordered by timestamps. However, process data is better represented as graphs, a structure that recent works have started to explore in order to better capture aspects like parallelism of events and relationships among traces [53, 148, 1, 14, 114]. Alternatively, to enhance predictive performance and explainability, researchers have proposed event-level feature engineering techniques to capture the mentioned relationships as well [144, 145]. Ideally, managing and encoding all levels (i.e., event and trace) simultaneously would tend to result in richer insights, as each level offers unique interpretive value [179].

3.1.1 Motivation and Related Works

Due to the fact that process mining is an emerging research field, there remains a vast amount of unexplored topics to be studied. A key challenge and a promising research direction is the development of better methods to represent process data as numerical representations (also known as latent spaces). As previously discussed, before this thesis, the dominant approach to representing traces involved (i) encoding activity labels using one-hot encoding, where a d-dimensional vector is filled with zeroes except for a single one that identifies the category being encoded; (ii) extracting basic features from timestamps, such as the day of the week; and (iii) aggregating the sequential data (e.g., by averaging or summing) into a single vector to train machine learning models. In this Chapter, we advance the field by critically examining and enhancing these encoding techniques.

Although encoding methods are crucial in predictive process monitoring, there is no standardized approach for selecting and evaluating these methods. Outside the process mining domain, the literature on representational learning has made significant advances, using neural networks to create embeddings for various tasks, such as graph-based embedding techniques [62] and text encoding methods [61]. In PPM, approaches like *act2vec*, *trace2vec*, *log2vec*, and *model2vec* [83] have been inspired by natural language processing techniques such as *word2vec* [104, 105]. Other techniques rely on graph convolutional networks [171] and image-like structures [113]. Despite these options, most PM studies still rely on basic techniques like one-hot encoding or simple frequency counts, with more sophisticated embeddings and handcrafted features rarely utilized.

Selecting the right encoding technique is often a complex and non-trivial task. As illustrated in Figure 3.2, different encoding techniques and their hyperparam-

eters can have a significant impact on performance. More specifically, we are illustrating an example of anomaly detection. The techniques *NMF-ADMM* and *Walklets*, which share a common hyperparameter (the number of dimensions), demonstrate different behaviors based on the event log characteristics. For one log (left), *Walklets* performs better with a smaller number of dimensions, while *NMF-ADMM* outperforms when the dimensionality is higher. On another log (right) with different characteristics (number of events, unique activities, etc.), *Walklets* is more stable across hyperparameter choices, consistently outperforming *NMF-ADMM*. This variability emphasizes the challenge of selecting appropriate encoding methods.

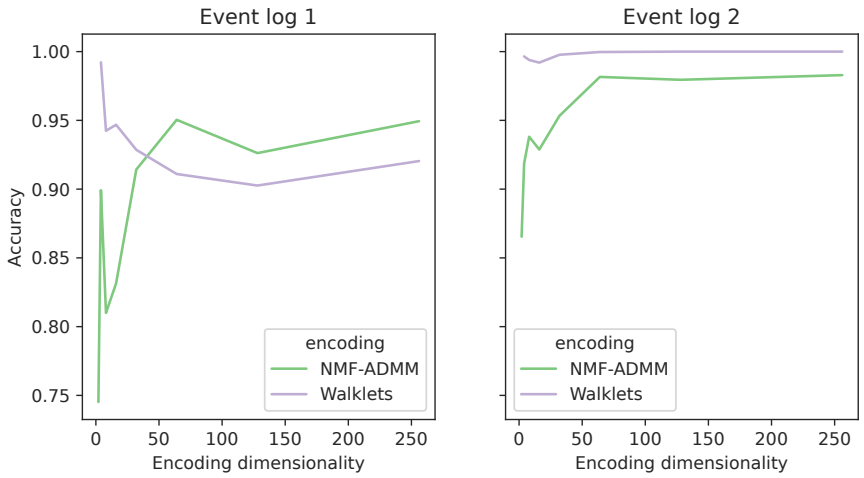


Figure 3.2: A brief example of performances achieved for two different datasets regarding the anomaly detection problem. For both datasets, we fixed two graph-based encoding methods with different parametrizations regarding dimensionality.

We argue that the literature has overly relied on one-hot encoding as a default choice, neglecting other more advanced techniques. This tendency has led to an overuse of deep learning models, which can effectively learn directly from raw data without the need for complex preprocessing. However, predictive process monitoring often operates in dynamic environments that require fast data processing and frequent model updates [34], making deep learning models impractical due to their time-consuming nature. In contrast, shallow learning algorithms, with their computational efficiency, are better suited for such environments. Furthermore, deep learning models face challenges related to explainability, as their complex structure and sensitivity to hyperparameters can cause instability in explainability metrics, as shown by [94]. This issue is especially significant in business contexts, where understanding model decisions is crucial. Despite re-

cent efforts to overcome this limitation [147], shallow algorithms still outperform deep algorithms in this regard [57]. Additionally, [84] found that increased model complexity, such as in neural networks, leads to higher levels of disagreement among different explainability methods.

Therefore, this Chapter makes two key contributions. First, we conducted an extensive survey of encoding techniques from various fields, evaluating their effectiveness in capturing the complex semantics of process data [153]. For this contribution, we use anomaly detection as a predictive task to validate our research. Moreover, we evaluate alternative aggregation strategies from the literature to aggregate the temporal behavior into d-dimensional vectors. Second, we introduced new feature extraction techniques tailored for timestamps, thereby improving the representational power of the models. In this scenario, we use the remaining time prediction as a task. In a nutshell, our main contributions include a comprehensive survey of *seq2vec* methods, the development of a new *evt2vec* method focused on time-related features, and improvements to the *seq2vec* approach. These innovations provide more effective ways to encode process data, as demonstrated through two different predictive tasks in process mining. The goal of validating our proposals through two different tasks is to provide a broader perspective of the issue being addressed through two different perspectives.

3.2 Survey on Encoding Methods

This survey is part of extensive research published in a special issue of AI applications for process mining [153]. Therefore, in this Section, we will abstract the main insights related to the capabilities of encoding methods capturing the complex process structure. These capabilities are measured in terms of three evaluation metrics that we will further described.

3.2.1 Experimental Setup

Our experiments are organized into three steps: (i) dataset preparation, (ii) encoding generation, and (iii) evaluation of the encoding methods from multiple perspectives. First, we generated synthetic logs using the PLG2 tool [23] and injected well-known process mining anomalies into them. Second, we group each encoding method according to families and provide the respective references for original papers and online implementations. In summary, we set families of baselines, process mining-based, text-based, and graph-based methods. The *evt2vec* is performed at the activity label (i.e., control-flow only), then the *tra2seq* employed is the prefix modeling to build the train/test datasets, and finally the *seq2vec* method used to encode the prefixes was the aggregation by average and

Table 3.1: Overview of five process models. For each scenario, we generated event logs by combining three different cardinalities, injecting seven different anomalies, at four different rates of injection (5%, 10%, 15%, and 20%). This resulted in 84 event logs for each scenario and 420 event logs in total. *gw*, *acts*, *evts*, and *vars* stand for the number of gateways, activities, events, and variants, respectively. The increasing complexity from scenario 1 to scenario is intuitively thought as the number of gateways and activities present in the log.

log	**#gw**	**trace size**	**#acts**	**#cases** (10^3)	**#evts** (10^3)	**#vars** (10^3)
scenario 1	8	9-13	22	1	75 ± 1	1 ± 0
				5	378 ± 5	2 ± 0
				10	755 ± 9	2 ± 1
scenario 2	12	26-30	41	1	186 ± 1	1 ± 0
				5	929 ± 5	5 ± 0
				10	1857 ± 10	10 ± 0
scenario 3	22	42-50	64	1	308 ± 1	1 ± 0
				5	1538 ± 5	5 ± 0
				10	3077 ± 10	10 ± 0
scenario 4	30	3-30	83	1	89 ± 3	0 ± 0
				5	439 ± 6	2 ± 0
				10	879 ± 12	4 ± 0
scenario 5	34	4-37	103	1	133 ± 3	1 ± 0
				5	659 ± 7	3 ± 0
				10	1318 ± 13	6 ± 0

summation to obtain the numerical trace representation. Finally, we employ three evaluation metrics from the literature to validate our methodology.

Synthetic event logs generation. PLG2 [23] was used to create five different process models by performing a random generation of a process capable of capturing several behaviors, such as sequential, parallel, and iterative control-flow. The rationale of PLG2 is based on the combination of basic control-flow patterns [137], e.g., sequence, parallel split, and synchronization. In order to simulate real-world scenarios, the patterns are progressively combined according to predetermined rules. Each of the five generated process models defines *five scenarios* of different complexities based on the number of activities and gateways included in the scenario. An overview is presented in Table 3.1.

Anomaly injection. For each scenario, we injected different percentages of anomalies (5%, 10%, 15%, and 20%) by replacing normal traces. A total of *420 event logs* were generated given five process models, six types of anomalies, and four anomaly percentages using labels and descriptions as additional attributes. Labels regard a normal execution or an anomalous one. It is important to note the different scenarios were created with increasing trace lengths and log sizes (1k, 5k, and 10k cases). We intuitively assume the complexity of event logs increases from scenario 1 to scenario 5 since we slowly increase these model properties such as the number of gateways and activities. We injected anomalies, following [16], by perturbing regular traces as proposed by [108], as illustrated in Table 3.2.

Anomaly	Description
skip	A sequence of 3 or fewer necessary events are skipped
insert	3 or fewer random activities are inserted in the case
rework	A sequence of 3 or fewer necessary events are executed twice
early	A sequence of 2 or fewer events executed too early, which is then skipped later in the case
late	A sequence of 2 or fewer events executed too late
all	Scenario where the event log is affected by all anomalies listed above

Table 3.2: Anomalies used to simulate the real-life event logs.

Evaluation metrics. We employ three evaluation metrics. First, the *N2* metric is a classification complexity metric [96] to measure how well samples, i.e., encoded traces, are distributed within classes. This measure is sensitive to how data are distributed within classes and labeling noise in the data. Low values are indicative of simple problems. Second, we employ the *F1-score* [141] to observe the impact of encoding methods on the accuracy, representing the predictive performance delivered by an encoding method to detect anomalies. Finally, the *T4* metric is the ratio of the Principal Component Analysis (PCA) dimension to the original dimension. This measure is related to the proportion of relevant dimensions that the coded feature vector is composed of. A larger T4 value means more encoded features are needed to describe data variability. Thus, the two first metrics aim to measure the capacity of the encoding method to improve the original space, by making classes more distinguishable; whereas the latter can be thought of as an information loss measure, to measure how well the encode method converts event data into a new latent space.

We subsequently introduce all the encoding methods employed in this work.

3.2.2 The Encoding Methods

We now present the encoding methods found in the literature, and we are including only the ones with open-source implementation available. These methods are introduced in this section and summarized in Table 3.3 along with their open-source libraries in Python, which include Sklearn[1], Karate Club[2], PM4PY[3], NLTK[4], Gensim[5], GloVe[6], and the *position profile* implementation on github[7].

[1]https://github.com/scikit-learn/scikit-learn
[2]https://github.com/benedekrozemberczki/karateclub
[3]https://github.com/pm4py/pm4py-core
[4]https://github.com/nltk/nltk
[5]https://github.com/RaRe-Technologies/gensim
[6]https://github.com/maciejkula/glove-python
[7]https://github.com/gbrltv/meta_trace_clustering/blob/main/clustering.py#L64

We organize each method according to families and provide the respective references for original papers and online implementations. Moreover, we set as baselines the methods *count2vec*, *one-hot*, *n-grams*, which implement the most simple transformations and are commonly employed in process mining papers.

Algorithm	Year	Family	Implementation
n-grams [58]	-	Baseline	NLTK
one-hot [176]	-	Baseline	Sklearn
count2vec [176]	-	Baseline	Sklearn
token-replay [162]	2016	PM	PM4PY
alignment [162]	2016	PM	PM4PY
Log skeleton [172]	2018	PM	PM4PY
position profile [33]	2017	PM	GitHub
hash2vec [176]	-	Text	Sklearn
TF-IDF [98]	1958	Text	Sklearn
word2vec (CBOW) [105]	2013	Text	Gensim
word2vec (skip-gram) [104]	2013	Text	Gensim
doc2vec [89]	2014	Text	Gensim
GloVe [116]	2014	Text	GloVe
DeepWalk [119]	2014	Graph	Karate Club
node2vec [66]	2016	Graph	Karate Club
Walklets [120]	2017	Graph	Karate Club
role2vec [2]	2018	Graph	Karate Club
Laplacian Eigenmaps [11]	2001	Graph	Karate Club
GraRep [31]	2015	Graph	Karate Club
Hope [110]	2016	Graph	Karate Club
BoostNE [91]	2019	Graph	Karate Club
diff2vec [135]	2018	Graph	Karate Club
GLEE [160]	2020	Graph	Karate Club
NetMF [126]	2018	Graph	Karate Club
NMF-ADMM [151]	2014	Graph	Karate Club
GraphWave [44]	2018	Graph	Karate Club
NodeSketch [183]	2019	Graph	Karate Club

Table 3.3: Encoding methods and related details.

PM-based Encoding

Given an event log, we discover its process model and perform conformance-checking techniques to measure its adherence to the model. Each trace in the

event log is evaluated. The results produced are employed as the encoded representation of the trace. The methods considered in our survey are illustrated below.

Token-replay [12]: given a process model, traces are replayed in it to obtain values that measure its conformance. More specifically, the values accumulated at each step are the number of tokens correctly consumed (c), the number of tokens correctly produced (p), the number of missing tokens to execute the event in the next step (m), and the number of unconsumed tokens after the last event execution (r). Thus, the final measure defined by the token-replay method is given by $fitness = \frac{1}{2}(1 - \frac{m}{c}) + \frac{1}{2}(1 - \frac{r}{p})$. All the values produced, $\langle c, p, m, r, fitness \rangle$, are used as the feature vector of a given trace.

Alignment [32]: performs a comparison between the process model and a trace and relates the trace to valid execution sequences, i.e., allowed by the model. An alignment is a sequence of moves that can be synchronous, model-dependent, or log-dependent. It is also important to note that more than one alignment between the log and model is possible, and techniques aim at finding the optimal one. The final feature vector is composed of the cost of the alignment, the number of visited states, the number of queued states, the number of traversed arcs, and the fitness value produced.

Log skeleton [172]: this technique aims at summarizing activity traces by capturing a set of constraints that apply to activities throughout the log. For example, the R_L^{eq} captures the equivalence relation between two activities, which exists if both activities have the same frequency of occurrence in every trace. On the other hand, the C_L^{df} counts the number of directly-follows occurrences for every pair of activities. Other examples of measures to capture relations include the always-after and never-together; examples of countermeasures include the sum of occurrences of a given activity in the entire log and the min and max numbers of occurrences of an activity in any trace. In the implementation used for this paper, six different constraints are used.

Position profile [33]: this technique represents an event log through a matrix, where each position refers to the $activity \times position$ regarding all traces. It can be formally defined as a triple $apf = (a, p, f) \in E$, where a is the activity, p is the position of the activity, f is the frequency of occurrence of the given activity, and E is the universe of events.

Text-inspired Encoding

Many solutions used for trace encoding in PM are adapted from methods used in NLP. Exploiting the fact that words in sentences are ordered in sequence and are constrained by dependencies, encoding methods applied to text capture that information. Because traces are composed of sequences of activities the same

information appears relevant to characterize them. In particular, in our survey, we consider the following methods.

N-grams [58]: this method represents a given sequence of elements through sub-sequences of n items. Thus, considering a sequence $\mathbf{s} = \{s_1, \ldots, s_i\}$, the *n-grams* representation of these sequences is given by $\textit{n-grams} = \{(s_1, \ldots, s_n), (s_2, \ldots, s_{n+1}), \ldots (s_{i-n}, \ldots, s_i)\}$.

One-hot [176]: given a variable containing n different values, the variable is transformed into an array where each unique value is represented as a binary vector with the *i-th* position set to one and the rest set to zero. Clearly, the dimension of the vector depends on the size n of the unique values in the vector space, easily reaching high dimensional spaces.

CountVectorizer (count2vec) [176]: given a collection of categorical documents, this method produces a matrix of token occurrences, where each line in the matrix represents a document and each column a token. The size of the vector space depends on the n unique values in the vector space.

HashVectorizer (hash2vec) [176]: it does the same as *count2vec*. However, instead of storing tokens, it directly maps each token to a column position in the matrix of occurrences. It is mainly useful for large datasets and unlike *one-hot* and *count2vec*, which have the same dimensionality as the vocabulary length, this method has the flexibility to hash tokens in any dimensionality.

TF-IDF [98]: the term frequency (TF) captures the frequency of a particular token w.r.t. to a given document, whereas the inverse document frequency (IDF) measures how common the token is in the corpus. TF can be simply the number of times the token appears and the IDF is calculated as follows: $idf(t, D) = log(\frac{N}{count(d \in D : t \in d)})$, where t is the token and N is the number of documents d in the corpus D. Thus, the *TF-IDF* is obtained by multiplying both $\text{TF-IDF}(t, d, D) = tf(t, d) \times idf(t, D)$.

Word2vec [104, 105]: the main contribution behind *word2vec* was learning distributed representations of words and reducing the computational cost compared to the state-of-the-art at the time. Although there are two original model architectures for learning the word vectors, Continuous Bag-of-Words (*CBOW*) and Continuous Skip-gram Model (*skip-gram*), the core characteristic of *word2vec* is the removal of the hidden layer of a simple Neural Net Language Model. *CBOW* predicts the current word based on the t words around it, i.e., it predicts w_t given $(w_{t-i}, ..., w_t - 1, w_{t+1}, ...w_{t+i})$. On the other hand, given w_t, the *skip-gram* predicts the surrounding words $(w_{t-i}, ..., w_t - 1, w_{t+1}, ...w_{t+i})$. The parameter i in both cases is a parameter representing a range surrounding the current word w_t.

Doc2vec [89]: this algorithm is an extension of *word2vec* and learns the embeddings of documents (sentence, paragraph, essay, etc.). The difference w.r.t. *word2vec* is given by the learning which is performed via the distributed memory

and distributed bag of words models and by adding another vector (document ID) to the input.

GloVe [116]: this is an unsupervised learning algorithm for obtaining vector representations for words. The main intuition behind this model is the capturing ratios of word-word co-occurrence probabilities in order to capture both local and global dependencies. This is expressed by $F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$, where P_{ik} and P_{jk} are the probabilities that the word k appears in the context of words i and j respectively.

Graph-based Encoding

Process models are based on graphs, therefore, representing event data as graphs where nodes are activities and edges are control-flow relationships becomes natural. Considering that graph embedding methods offer promising encoding capabilities, it is of interest to study how such algorithms behave in the PM domain [7]. Furthermore, graph embeddings open new possibilities for PM analysis, such as capturing graph structures and finding similarities across different process models. The intuition behind graph embedding methods is to represent nodes of a graph as low dimensional vectors, where such vectors are representative enough to keep its original relations (edges) intact. We can formally define the general idea as follows. A graph can be described as $G = (V, E)$, where $V = \{v_1, ..., v_n\}$ is a set of vertices (nodes) and E is a set of edges $e = (u, v)$ that connect a pair of vertices $u, v \in V$. Given a graph G, a graph embedding is a mapping function $f : v_i \rightarrow y_i \in R^d$, such that $d << |v|$ and f preserves the original structure of their local neighborhood and minimizes the information loss. In this section, we describe graph embedding methods for event log encoding.

DeepWalk [119]: it can be seen as a two-stage algorithm. First, a discovery of the local structure is performed through random walks. There are two parameters here, the number of random walks α and the number of vertices to visit t for each random walk. Second, similar to the *word2vec*, the *skip-gram* is performed to learn the embeddings. The intuition behind this algorithm is learning embeddings close to each other if they often occur in a similar structural context.

Node2vec [66]: this algorithm is similar to *DeepWalk*, where the difference is a biased-random walk that aims at employing a trade-off between breadth-first and depth-first searches. In practice, such balance is capable of providing more informative embeddings than *DeepWalk*.

Walklets [120]: while *DeepWalk* and *node2vec* implicitly capture a certain level of local dependencies by generating multiple random walks from a starting point node, this algorithm combines factorization approaches with random walks to capture both local and global information. It preserves dependencies by subsampling short random walks on the vertices and by skipping over steps in each

random walk. This results in paths of fixed lengths composing sets of pairs of vertices. Thus, these sets are used to learn the latent representations.

role2vec [2]: this is a framework that uses random walks to approximate the pointwise mutual information matrix, which is obtained by multiplying a matrix of structural features with the pooled adjacency power matrix.

Laplacian Eigenmaps [11]: the algorithm aims to find a low-dimensional this algorithm intuitively keeps the embedding of two nodes close when the weight W_{ij} is high. Given a graph G, this algorithm computes eigenvalues and eigenvectors $Ly = \lambda Dy$, where D is a diagonal weight matrix $D_{ii} = \sum_j W_{ji}$, and W is the weight matrix. Thus, $L = D - W$ is the Laplacian matrix that can be used to minimize the function $\rho(Y) = \frac{1}{2}\sum |Y_i - Y_j|^2 W_{ij} = tr(Y^T LY)$, where $Y \in \mathbf{R}^{N\times\hat{d}}$ is a matrix of $\hat{d}$ eigenvectors (desired dimensionality of the low-dimensional space) associated with the smallest eigenvalues.

GraRep [31]: it aims to learn node embeddings by utilizing higher-order structural information in a graph. It computes k-step transition matrices, performs Singular Value Decomposition (SVD) on these matrices, concatenates the resulting singular vector matrices, and normalizes the concatenated matrix to obtain the final node embeddings.

Hope [110]: this embedding algorithm is similar to *GraRep*, but instead of using the transition probability matrix, it employs a similarity matrix S. Thus, S can be obtained by using different similarity measures and consequently preserves higher-order dependencies.

BoostNE [91]: this algorithm performs a non-negative matrix factorization to calculate the residuals generated by previous embedding models. It assumes the same idea as the gradient boosting method in ensemble learning, where multiple weak learners lead to a better one when aggregated. Given a connectivity matrix obtained through the adjacency matrix of the graph, the algorithm calculates k residual matrices and uses each one as input to the next one using the following equation:

$$R_i = \begin{cases} X, & \text{if } i = 1 \\ max(R_{i-1} - U_{i-1}V_{i-1}, 0), & \text{if } i \geq 2 \end{cases} \tag{3.1}$$

where $U_i \in R_+^{n\times d_s}$ and $V_i \in R_+^{n\times d_s}$ intuitively act like the embedding representation of the center node and the context node in the $i-th$ level, respectively. Assuming the defined residual matrix, the embedding representation at the $i-th$ level is obtained by minimizing the loss function $L = \min_{U_i,V_i,\geq 0} ||R_i - U_iV_i||_F^2$, for $1 <= i <= k$.

Diff2vec [135]: the overall idea of this algorithm is sub-sampling diffusion graphs for each node in a graph and generating sequences of vertices through an Euler tour. Given a graph G, a graph G' of l vertices is sub-sampled in a

diffusion-like random process. Then, from G', sequences of vertices are generated by performing an Euler walk. In this process, G' is first converted to a multigraph by doubling each edge. Thus, the Euler walk is employed instead of the random walk since this algorithm can capture a more complete view in graphs with this characteristic. The generated sequences of vertices are then used to create the graph embedding.

GLEE [160]: unlike most graph embedding algorithms that expect similar nodes to have their embeddings close to each other, this algorithm uses the Laplacian matrix of a given graph to find an embedding with geometric properties. Examples of such properties are dot product (angle), length (area or volume) of a line segment (or polygon), the convex hull of a set of vectors, etc. Thus, given a graph G and its Laplacian matrix L, this procedure extracts eigenvectors corresponding to the largest eigenvalues in L. These vectors are used as node embeddings.

NetMF [126]: this method is built on a theoretical analysis that shows the equivalence of different graph embedding algorithms based on *DeepWalk*. In the original paper, the authors show that methods that use negative samplings, such as *DeepWalk* and *node2vec*, implicitly perform matrix factorization. Thus, the framework *NetMF* is proposed to unify existing methods and perform an explicit factorization.

NMF-ADMM [151]: given an adjacency matrix, the *NMF-ADMM* algorithm learns the embeddings by using the alternating direction method of multipliers to solve the negative matrix factorization problem.

GraphWave [44]: given an undirected graph $G = (V, E)$, an adjacency matrix A (binary or weighted), and a diagonal matrix $D_{ii} = \sum_j A_{ij}$ representing the degree of node i, this method learns a structural embedding of every vertex $v \in V$. The resulting GraphWave represents the compact node embeddings, where each row corresponds to a node in the graph and each column represents a dimension in the embedding space. These embeddings encode the structural similarities and local connectivity patterns of nodes.

NodeSketch [183]: this method recursively generates k-order node embeddings in a recursive manner. These embeddings are categorized into low-order ($k = 2$) and high-order ($k > 2$). At each step, k, a Self-Loop-Augmented (SLA) adjacency matrix is generated to obtain the embeddings. Low-order SLA is obtained by simply adding the identity matrix to the original adjacency matrix $M' = M + I$. On the other hand, high-order embeddings first sketch an approximate k-order SLA adjacency of the current nodes and merge it with the $(k-1)$-order SLA adjacency matrix in a weighted manner.

3.2.3 On the Impact of Hyperparameters and Logs' Characteristics

We now dive into the algorithms' behaviors according to their hyperparameters and the event logs' characteristics. We present the impact of hyperparameters for each family of algorithms.

Hyperparameter Impact. Employing synthetic data is beneficial for this evaluation since it allows us to have more control over different properties and obtain insights regarding which algorithm family and configurations can achieve better results according to the user's preferences. Thus, we discuss the employed configurations for parametric methods (i.e., graph- and word-based methods, see Table 3.4) across the employed scenarios. Furthermore, all the results in this Section are filtered and only the performance values lying between the first and third quartiles are included to avoid outliers and bad visualizations.

Table 3.4 presents two important configurations that guide embedding generation: feature vector size and aggregation methods. A key aspect that we aimed to capture with experiments is scalability. Moreover, by analyzing different vector sizes we can assess how well the encoding method distributes the generated embeddings. This way, we fixed feature vector sizes to $\{2^n\}_{n=1}^{8}$, i.e., ranging from vectors of size 2 to 256. Due to vocabulary size, some embeddings were limited in terms of possible size configurations (e.g., *Laplacian Eigenmaps*, *GraRep*, *Walklets*, *NodeSketch*, *GLEE*, and *Hope*). *BoostNE* required a particular configuration that only allowed multiples of 17, this way, we tried to obtain vector sizes as close as possible to powers of 2. It is important to note that word and graph embeddings create an embedding representation for words and nodes, respectively. Therefore, aggregation techniques are required to obtain a trace representation, i.e., an aggregation of the word or node representations. As for the trace, we selected two options: average and max pooling. For the graph embeddings, we can obtain representations for both the nodes and edges, increasing the possible aggregation options. Considering the proposed edge aggregations in [66], the edges embedding can be obtained following four binary operators: Average, Hadamard, Weighted L1, and Weighted L2. Given $f(x)$ as the function that returns the node embedding and u and v as two nodes belonging to a graph, the edge embedding is computed as described in Equations 3.2, 3.3, 3.4, 3.5.

$$Average = \frac{f(u) + f(v)}{2} \tag{3.2}$$

$$Hadamard = f(u) \times f(v) \tag{3.3}$$

$$WeightedL1 = |f(u) - f(v)| \tag{3.4}$$

Table 3.4: Employed configurations for word- and graph-based encoding methods. Refer to our paper [153] for a detailed description of each encoding technique.

Encoding	Vector Size	Trace Agg.	Graph Agg.	Edge Agg.
GraphWave	$\{2^n\}_{n=1}^{8}$	{avg, max}	{node, edge}	{avg, had, w1, w2}
Laplacian Eigenmaps	$\{2^n\}_{n=1}^{4}$	{avg, max}	{node, edge}	{avg, had, w1, w2}
NMF-ADMM	$\{2^n\}_{n=1}^{8}$	{avg, max}	{node, edge}	{avg, had, w1, w2}
DeepWalk	$\{2^n\}_{n=1}^{8}$	{avg, max}	{node, edge}	{avg, had, w1, w2}
GraRep	$\{2^n\}_{n=1}^{4}$	{avg, max}	{node, edge}	{avg, had, w1, w2}
node2vec	$\{2^n\}_{n=1}^{8}$	{avg, max}	{node, edge}	{avg, had, w1, w2}
Walklets	$\{2^n\}_{n=2}^{8}$	{avg, max}	{node, edge}	{avg, had, w1, w2}
role2vec	$\{2^n\}_{n=1}^{8}$	{avg, max}	{node, edge}	{avg, had, w1, w2}
NetMF	$\{2^n\}_{n=1}^{8}$	{avg, max}	{node, edge}	{avg, had, w1, w2}
NodeSketch	$\{2^n\}_{n=2}^{8}$	{avg, max}	{node, edge}	{avg, had, w1, w2}
BoostNE	$\{17, 34, 68, 136, 255\}$	{avg, max}	{node, edge}	{avg, had, w1, w2}
GLEE	$\{2^n\}_{n=1}^{4}$	{avg, max}	{node, edge}	{avg, had, w1, w2}
Hope	$\{2^n\}_{n=1}^{5}$	{avg, max}	{node, edge}	{avg, had, w1, w2}
diff2vec	$\{2^n\}_{n=1}^{8}$	{avg, max}	-	-
word2vec	$\{2^n\}_{n=1}^{8}$	{avg, max}	-	-
hash2vec	$\{2^n\}_{n=1}^{8}$	{avg, max}	-	-
GloVe	$\{2^n\}_{n=1}^{8}$	{avg, max}	-	-
doc2vec	$\{2^n\}_{n=1}^{8}$	{avg, max}	-	-

$$WeightedL2 = |f(u) - f(v)|^2 \tag{3.5}$$

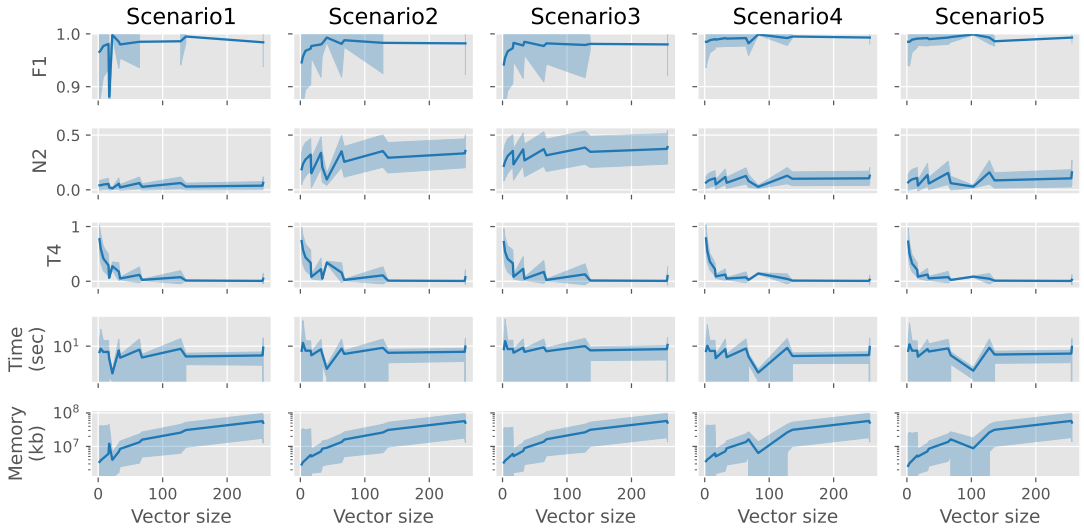


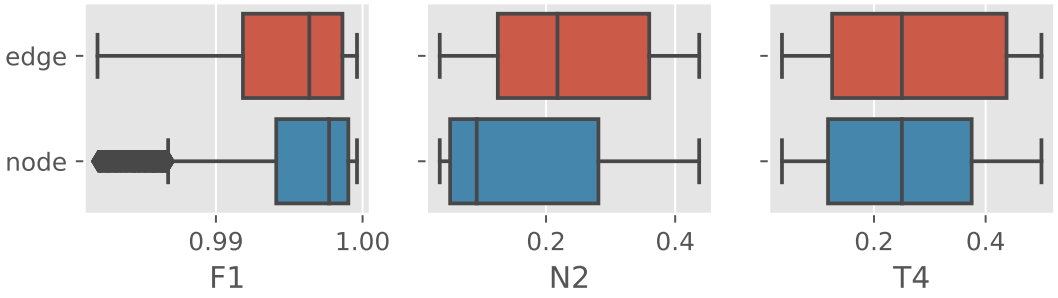
Figure 3.3: Performance metrics (one for row) for different vector sizes throughout all the employed scenarios (one for column).

Figure 3.3 shows the behaviors of different hyperparameter vector sizes (see Table 3.4) across the five employed scenarios. We can see that for T4, although the hyperparameters affect this metric, they behave similarly across all the scenarios. The behavior of this hyperparameter for the mentioned metric is intuitive since T4 measures the ratio of the PCA dimension to the original data dimension. Furthermore, the elapsed time for encoding methods across scenarios does not change and we can see in the figure a similar behavior for them. Thus, if this is the metric of interest of a user, the choice is intuitive and straightforward.

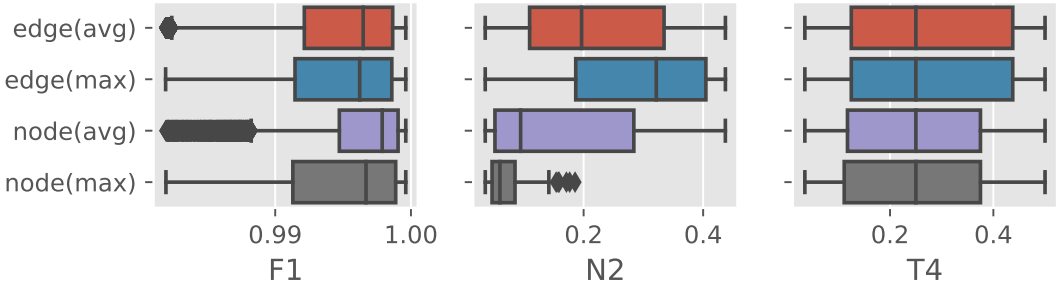
On the other hand, regarding the classification complexity metrics F1 and N2, the hyperparameter choice depends on the characteristics of the event log. For example, for scenarios 1, 4, and 5, N2 has slight differences, whereas it considerably changes for scenarios 2 and 3. Considering the F1 metric, although the average behavior is similar across the scenarios, we can see a high variation of performances in the first three scenarios for small vector sizes.

This evaluation highlights the difficulty of choosing the right hyperparameter values according to the user's requirements. For instance, we can see that high F1 scores are achievable by employing high and low vector sizes. However, if only a limited amount of memory is available, an algorithm selection procedure is needed to find the right algorithm that achieves the desired score of F1 and fits the available storage. Nevertheless, if the user faces characteristics similar to scenarios 2 and 3, other hyperparameters and algorithms should be evaluated to optimize N2.

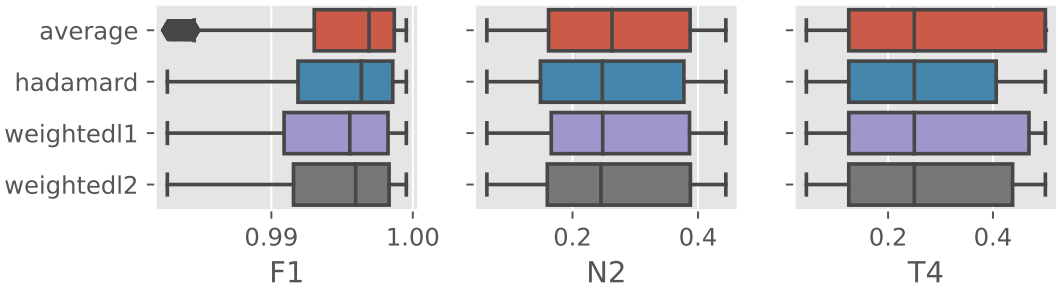
Another hyperparameter to be evaluated is the aggregation strategy (note: this is not the same aggregation from trace encoding) for algorithms from the graph and word families. Figure 3.4 shows the overall distributions of the employed methods for each type of aggregation. First, Figure 3.4(a) illustrates a high-level overview regarding the aggregation of edges and nodes all at once. We can see that, in general, aggregating nodes for the final encoded trace significantly overcome the edge aggregation regarding N2, whereas for the remaining ones it slightly overcomes or performs equally. This is due to the fact the full information of a trace is retained in the node representation and not in the edges that connect the encoded trace to similar or related (depending on the graph algorithm) neighbors. Second, the overall aggregations for edges and nodes according to each strategy (average or max) are presented in Figure 3.4(b). We can see that for edges the overall behavior is the same across all metrics except for N2, which performs slightly better in general for average aggregation. Moreover, there is no best aggregation regarding nodes, since for the F1 and T2 metrics the behavior is quite similar, whereas for F1 the average is slightly better (higher average score and lower variation), and for N2 is the opposite. Lastly, in Figure 3.4(c) we also show the individual aggregation strategies for the edges, although there is no significant difference in performances among these.



(a)



(b)



(c)

Figure 3.4: (a) Overall aggregation of edges and nodes. (b) Specific aggregation of edges and nodes. (c) Aggregation strategies for edges.

Finally, the aggregations for methods from the word family are illustrated in Figure 3.5. T4 metric behaves similarly to the Figure 3.3 and Figure 3.4. In this case, however, F1 and N2 scores also behave similarly in general, although the average aggregation presents a slightly lower variation for the former one.

It is evident the wide distribution of possible performances regarding all metrics that might be achieved by the same hyperparameter value across different event logs. Always selecting the same feature vector size and aggregation strategy

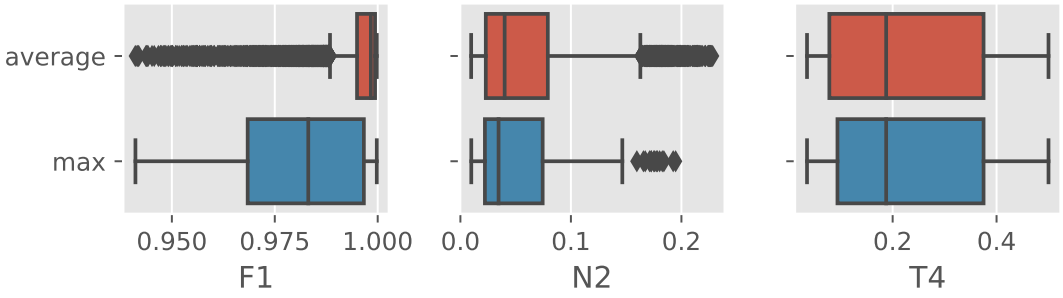


Figure 3.5: Distribution of metrics for aggregations of word-based encoding methods.

might lead to drastic differences among metrics. Thus, in the remaining Chapter, we evaluate which underlying characteristics in the event logs might affect the performances.

Anomaly Analysis. We now evaluate how each anomaly type affects the encoding methods in general. The performances regarding each type of anomaly are aggregated throughout all the families to visualize the general behaviors in Figure 3.6. We employ the violin plots for this visualization since it better highlights the density curves and allows us to deeply see where the main differences among the anomalies lie. For instance, similar F1 scores are achieved by all methods and hyperparameters regarding the *all* anomaly. We can assume the *attribute* anomaly as the easiest one for F1 since most performances tend to be almost 1, whereas, for the remaining anomalies, we see a small concentration close to 1 although the overall behavior is the same. On the other hand, N2 is significantly affected by the different anomalies. Regarding the *all*, *attribute*, and *late* anomalies, some extra effort should be considered to find the right algorithm and its hyperparameters since the values of the performance are more concentrated above their averages. Finally, the T metric behaves similarly for this evaluation as well.

In order to go deeper into the anomaly analysis, we present in Figure 3.7 the same idea but this time separated by families and via bars. Since we are including all the results without excluding outliers as in the previous discussion, we can highlight overall differences among the families. Regarding the F1 metric we can notice a significant poor performance by the PM encoding methods, except for the *attribute* anomaly. We can also notice slight variations in performances among anomalies with respect to the text-based methods, whereas this does not occur for the graph-based methods. Contradictorily, we can see a better performance of PM methods regarding N2, which emphasizes the difficulty of selecting a suitable algorithm for each task. Furthermore, the graph-based methods were overcome

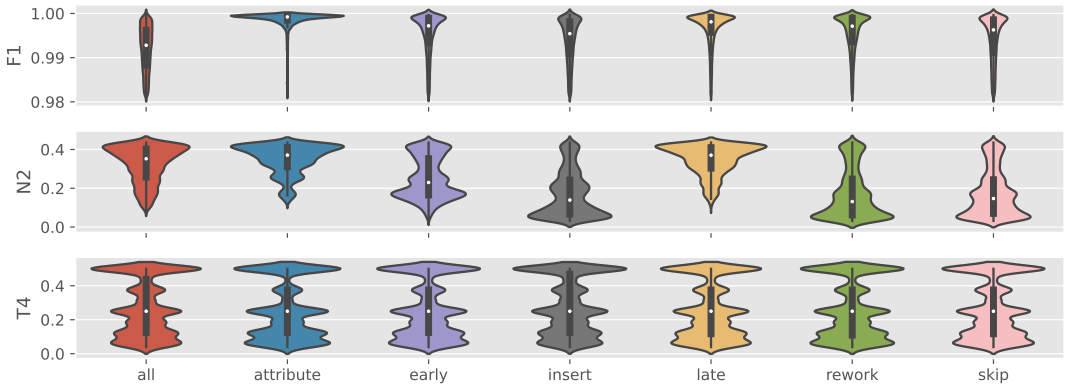


Figure 3.6: Distribution of performances for each anomaly type.

by the other ones in this metric.

More specifically, methods from the baseline and PM families present the lowest T4 metrics since they are non-parametric, which means they have the same encoded dimension regardless of the event log. Furthermore, the PM methods are drastically affected by the anomalies because some anomalies are easier to detect (e.g., early) than others (e.g., rework). Encoding methods from the graph and word families are not affected by the anomalies and they present the same patterns in general.

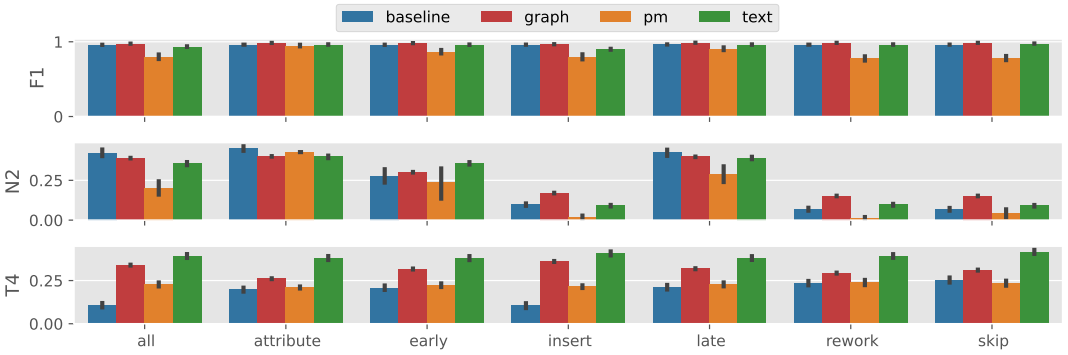


Figure 3.7: Overall performances for each anomaly type and grouped by algorithm families.

Log Behavior. Besides the anomaly type affecting the PM family algorithms, in this Section, we will analyze further properties of event logs that might also affect the performance metrics.

In Figure 3.8, we show the overall performances achieved by each family of algorithms over the scenarios. Regarding the F1 metric, the first three scenarios

present a slightly higher variation in performance, suggesting more difficulty in finding the right algorithms and hyperparameters. However, this variation is due to the fact that simple data (i.e., a small number of activities, short traces, etc.) do not require exploring such a wide range of hyperparameters, and the higher the hyperparameter values (e.g., vector size) the higher the information loss. Regarding the remaining metrics, we can see some disturbance with respect to scenarios 2 and 3. There is a high variation and high average values for N2, which emphasizes that the event properties affect the algorithms' performances. The PM family performs better for T4. Overall, the graph methods perform better or are similar with respect to most of the other ones across all scenarios.

This analysis indicates that event log properties do not drastically affect the employed encoding methods, except for the PM family. This is intuitive since this family has an underlying bias to handle this specific data domain, i.e., process data.

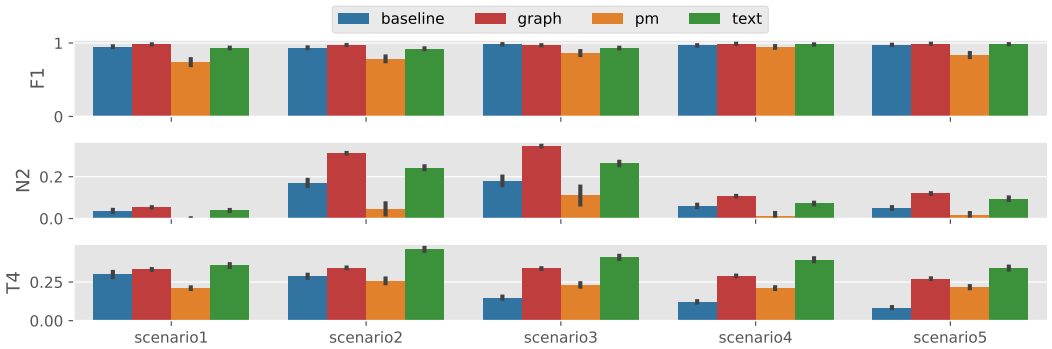


Figure 3.8: Performances by scenario.

Since the specific event properties do not present many behavior changes, in Figure 3.9 we perform the last evaluation regarding the log characteristics but with respect to their sizes for each performance metric. For F1, we can notice that the log size affects only the PM methods, indicating they are not scalable. Again, scenarios 2 and 3 seem problematic since the families' behaviors significantly change for the N2 although the remaining behavior remains the same as discussed in the previous figure. T4 has the expected behavior since this metric should not be affected by the event log sizes.

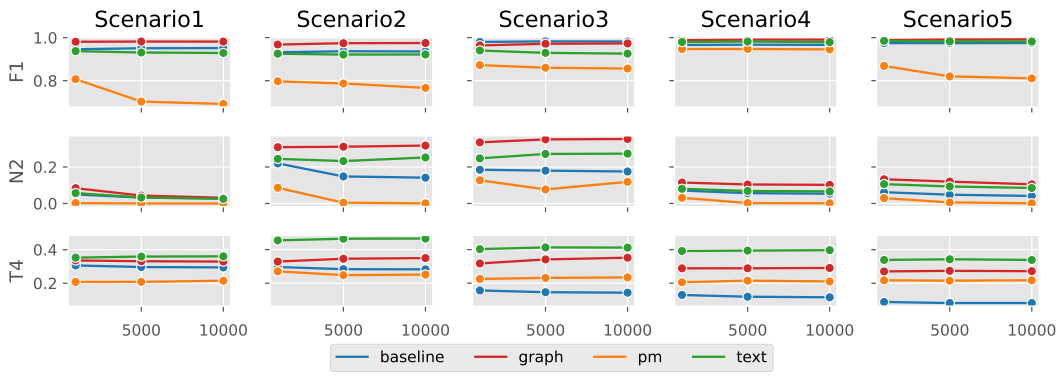


Figure 3.9: Performances (rows) for each scenario (columns) according to different event log sizes.

3.3 Advanced Feature Engineering to Compete against Deep Learning

We shall now present a contribution on how to make shallow learning models competitive against deep learning models. This is accomplished by proposing advanced feature engineering techniques to extract more meaningful and representative information from timestamps. We also motivate the benefits of our proposal by providing a discussion on interpretability based on the proposed features.

3.3.1 Our Proposal

In this Section, we aim to assess how preprocessing techniques can enhance predictive accuracy and interpretability in the remaining time prediction task. We review existing features from the literature and introduce new ones developed in this study.

We start by utilizing three features defined by Tax et al. [155]: *execution time* (the difference between the current and previous event timestamps, zero for the first event), *time within the day* (time from midnight), and *time within the week* (time from Sunday midnight). Additionally, we propose two new features: *accumulated time* (difference between the current timestamp and the initial event) and *weekday* as a categorical feature. These are referred to as *TF* (time-related features) in our experiments.

To improve predictive performance, it is common to extend raw data with statistical measures [3]. While [154] applied this idea to control-flow aspects, they did not explore time-related features. Building on their approach, we incorporate

statistical, distributional, and dispersion metrics to better capture temporal patterns. After extracting temporal features at each trace position t_i, we compute statistical measures like average execution time ($avg_{exec\_time} = \frac{\sum_{i=0}^{n} \#_{exec\_time}(e_i)}{n}$), variance, entropy, and kurtosis. A detailed list of these metrics is in Table 3.5. We then aggregate these measures using the method from Teinemaa et al. [158] to form vector representations of traces. The combined set of TF and these engineered features are referred to as *ALL* features in our experiments. In the following Sections, we describe the experimental setup and present the results.

Table 3.5: Acronym and a brief description of each statistical measure employed in this work.

Acronym	**Description**	**Acronym**	**Description**
min	Minimum value	*geometric_mean*	Geometric mean
max	Maximum value	*geometric_std*	Geometric Std.
mean	Mean value	*harmonic_mean*	Harmonic mean
median	Median value	*skewness*	Skewness
mode	Mode	*kurtosis*	Kurtosis
std	Standard deviation	*coefficient_variation*	Coefficient of variation
variance	Variance	*entropy*	Entropy
q1	25th percentile	*hist*	Histogram (10 bins)
q3	75th percentile	*skewness_hist*	Skewness histogram
iqr	Interquartile range	*kurtosis_hist*	Kurtosis histogram

3.3.2 Experimental Setup

In these experiments, we compare shallow learning algorithms against a deep learning baseline in different scenarios. Essentially, each scenario has a different preprocessing step to train the shallow algorithms. For the deep learning baseline, no preprocessing is made, i.e., the model is trained with the raw data. These scenarios, more specifically, are designed as follows: the first one uses raw data as well, the second one uses preprocessing techniques from the literature, and the third one uses our proposal. Each scenario is combined with two different *evt2vec* methods to encode the activity labels. We will further describe in more detail in the following.

We selected three shallow baseline algorithms from different families: kernel-based (Support Vector Machine (SVM)), similarity-based (k-Nearest Neighbors (KNN)), and tree-based (Random Forest (RF)), to assess how event feature engineering affects their performance. Python's Scikit-learn was used for these models, with initial hyperparameter tuning via Grid Search. Due to low variance in predictive performances, default settings were reported for fairness. For deep learning, we adopted the LSTM from [26], adjusting the output for time

prediction. Hyperparameters were optimized with Bayesian methods, resulting in $batch\_size = 32$, $epochs = 50$, $learning\_rate = 0.0005$, and $prefix\_len = 5$.

We evaluated performance using Root Mean Squared Error (RMSE), given by $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$, as it aligns with the original target unit (days). Feature importance was analyzed using RF and SHAP to explain model predictions, focusing on their popularity in PM. Our methodology included two *evt2vec* encoding methods (*one-hot* and *node2vec*) and an aggregation strategy for feature extraction. We tested six pipelines by incrementally adding time features and proposed features to the encodings. More specifically, we have the scenarios with an (i) encoding method only (*OH* or *node2vec*), (ii) encoding method plus the time-related features *TF* (e.g., *OH+TF*), and (iii) encoding methods plus *ALL* (e.g., *OH* + ALL) features (i.e., from the literature and ours)

We used BPI12 and BPI20 event logs and their variations, representing loan applications and expense claims, respectively, to test seven scenarios. Our study focused on timestamps, not covering attributes like resources and costs, which might be absent in some logs.

3.3.3 Evaluation

The Impact of Time-related Features. Figure 3.10 illustrates the significant progress made in improving predictive performance by incorporating time-related features, particularly concerning the RMSE metric. The inclusion of all features consistently yields the lowest error rates, despite the resulting increase in dimensionality. In the context of RF, this result aligns seamlessly with expectations given the inherent robustness of the algorithm, which prevents a decrease in predictive performance with the inclusion of additional features. Notably, the KNN algorithm also shows improvements when using all features, despite its traditional susceptibility to the curse of dimensionality. In contrast, the SVM, despite achieving competitive results in predictive monitoring as reported by Polato et al. [124], encounters challenges with high dimensionality and doesn't necessarily exhibit improvements through various preprocessing techniques, as evidenced by the *RequestForPayment* event log. The SVM algorithm's task of identifying a hyperplane for class separation becomes increasingly difficult as the dimensionality of the feature space increases. This analysis highlights that the robustness of machine learning predictions greatly benefits from carefully integrating comprehensive time-related features. In the next Section, we introduce a discussion on computational complexity and the trade-off between complexity and predictive performance.

Among the shallow algorithms, the SVM exhibits consistently poor performance across various metrics, including both elapsed time and predictive accu-

Figure 3.10: Predictive performance of remaining time according to different combinations of preprocessing methods for each learning algorithm (row) and event log (column).

racy. This underscores the necessity for fine-tuning to mitigate its high RMSE, as well as additional preprocessing measures for dimensionality reduction. In this case, these steps are aimed at slightly increasing preprocessing time but significantly reducing the training duration. On the other hand, the KNN algorithm demonstrates the shortest average elapsed time and minimal variation. However, it also exhibits higher error rates in terms of RMSE (as indicated in Table 3.6). This behavior can be attributed to the algorithm's intrinsic susceptibility to the curse of dimensionality. While the new event features contribute to enhancing predictive performance, the KNN algorithm could benefit from a subsequent feature selection stage, particularly considering the impact of dimensionality. Lastly, the RF algorithm showcases a notably low average elapsed time despite considerable variation. Remarkably, it achieves the best predictive performance among the shallow algorithms, with errors below the single time unit (a day). Notably, it surpasses LSTM performance in a specific event log scenario and maintains competitive performance for other event logs.

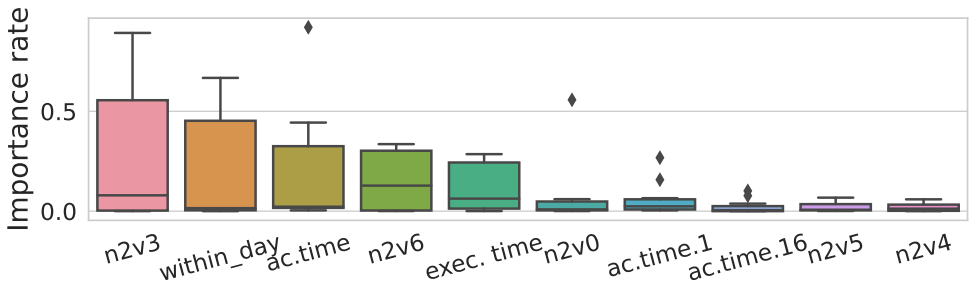
Table 3.6: RMSE scores, in days, for each model and each event log.

	BPI12_A	**BPI12_Complete**	**BPI12_O**	**BPI12_W**	**PrepaidTravelCost**	**RequestForPayment**
KNN	2.7077	4.2951	4.85	4.1232	12.0898	3.4464
LSTM	**0.0785**	**0.4348**	**1.4561**	**1.0097**	4.7497	**0.3839**
RF	0.2817	1.5617	1.6475	1.7889	**4.4131**	0.6338
SVM	3.5933	6.6551	8.8416	4.8422	9.0235	10.3065

Interpretability. We shall now highlight an additional advantage related to transparency offered by the optimal shallow method presented in this study. The

RF, in addition to low inference time and reasonable predictive performance, provides us with the ability to identify which feature has the most significant impact on the predicted value. Thus, the temporal features presented in this study not only demonstrate improvements in predictive performance but also provide transparency and interpretability through the RF. We then examine the overall impact of all the time-based features across all the event logs. We then select one specific event log to gain a deeper understanding of the scenario and conclude by reinforcing the feature explainability using the SHAP method.

Figure 3.11: Top 10 averaged feature (x-axis) importance (y-axis) for all event logs. *n2v* refers to the nth-dimension of node2vec, *ac.time* to accumulated time, and *exec.time* to execution time. The numerical suffixes for time features refer to the index of a statistical measure (described in our repository).

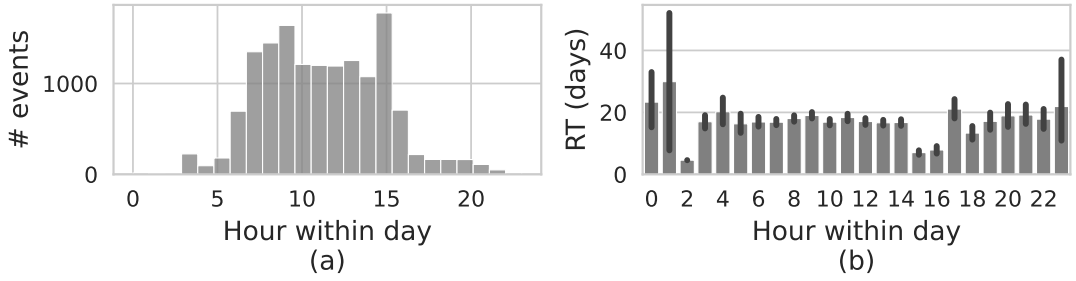


In order to evaluate the impact of the presented features from a global point of view, we select the 10 features with the highest importance averages according to the RF. We show their distributions as box plots in Figure 3.11. The suffix integers in the features' names represent the n-th dimension of the *node2vec* or the i-th statistical measure (see Table 3.5) extracted from the time-related feature, whereas the feature names without those suffixes refer to the time feature itself. The *n2v3* is the third feature (dimension) generated by the *node2vec* encoder and is the most important one. Although the control-flow of a (sub)trace is encoded into a new latent space, we can implicitly save the current activity from the ongoing trace before encoding the respective data, which could further support the user in interpreting the model's decision. On the other hand, the *within_day* is the second most impactful feature across all datasets and it regards the time within a day. With respect to the proposed features, we notice a lower importance rate, which leads us to conclude they have a thorough and refined participation in the final prediction since we showed in the previous Section their significant improvement in predictive accuracy.

Intuitively, the *within_day* feature is important for all datasets since it refers to commercial working time; the *BPI12* dataset describes events related to banks, and the *BPI20* describes events related to a university. We further investigate

the initial intuition and assert it in Figure 3.12. We select the *BPI20 PrepaidTravelCost* to illustrate that most events are indeed executed within the time intervals 7:00 and 17:00.

Figure 3.12: Getting insights from the most important time-related feature: (a) number of events that occurred within each hour of a day; and (b) remaining time (RT) distribution for each hour of the day.



Given the importance of the *time within day* feature, we filtered out activities with noticeable error rates measured by the absolute differences between predictions and actuals by comparing average true and predicted remaining times. Our investigation revolves around identifying, understanding, and explaining the poor performance of these activities despite the importance of the feature. Initial observations highlight the *Payment Handled* activity as having the most pronounced absolute error rates (approximately 135%) in predicting remaining time. Further observations indicate that this activity has minimal variance due to likely automated executions that consistently occur at the same daily time. Consequently, this reduced variance does not add significant information to the prediction model. Subsequent analysis suggests the influence of European summertime on event log timestamps, resulting in a temporal pattern: between October and March, timestamps are consistently recorded around 16:30, deviating from the 15:30 pattern observed in other months.

3.4 Final Remarks

In this Chapter, we successfully addressed our first research question (RQ1): "*Given the complex temporal, relational, and semantic dimensions of process data, what are the most effective methods for encoding and representing this data in latent spaces, and how do these methods perform across different process mining tasks?*". Our methodology showed that there is still room in the field of process data encoding, with significant opportunities for innovation. We highlighted the potential of encoding strategies at different granularity levels, such as at the

event attribute level or the case level. Furthermore, designing domain-specific engineered features tailored to the PM field emerged as a promising direction for research.

Our results demonstrated that even basic encoding techniques based on feature engineering could substantially enhance the predictive performance of shallow machine learning methods. This finding is particularly noteworthy because shallow models are not only computationally cheaper (as investigated in the following Chapter) but also offer greater interpretability compared to deep neural networks. This makes them a practical choice in scenarios where resources are limited or where model transparency is crucial. Furthermore, our findings align with several related works in the field [145, 57, 179], suggesting the potential to develop a systematic comparison of all available solutions in the future.

While deep learning models can achieve superior accuracy, their complexity and resource demands often make them less feasible in real-world settings. In contrast, shallow algorithms, augmented by carefully chosen encoding techniques, offer a compelling alternative that combines solid performance with greater accessibility and ease of use [179]. These insights lead us to future research to explore this trade-off and to develop more effective and adaptable encoding methods tailored to the unique challenges of PM. This promising future direction is already being explored in the literature [57, 179, 14]. Moreover, the computational resources needed for training such models is another concerning topic that is often disregarded. Therefore, in the next Chapter, we introduce a new methodology to tackle this important issue.

Chapter 4

Striking a Balance with CoSMo: Controlling Randomness in Process Simulation

Process simulation models are instrumental in a variety of PM applications including compliance analysis, performance analysis, and what-if analysis. Predominantly, these models are categorized into Data-driven Simulation (DD-S) models [101, 25, 27, 24], Deep Learning-based Simulation (DL-S) [26, 28], and hybrid approaches combining both [30, 103], each offering unique capabilities and facing distinct challenges. DD-S consists of discoverying a process model to guide he simulation, whereas the DL-S involves the training of a deep learning model capable of simulating behaviors, and the hybrid approaches combine both. While DD-S models excel in simulating control-flow patterns, they are often hindered by oversimplified assumptions about resource behavior and are susceptible to biases from event log characteristics. Conversely, DL models, despite their success in predictive process monitoring [127], struggle with the stochastic nature and the challenges of integrating what-if scenario knowledge, limiting their applicability in process simulation [47].

A natural aspect common to both DD-S and DL-S models is the stochastic approach in simulation, relying on randomness to yield realistic outcomes [161]. While this enhances realism, it might limit user control over simulated behavior and hinder practical flexibility (e.g., for what-if analysis). In this sense, DD-S models overcome this issue by providing tunable parameters [25], which serve as knobs to adjust characteristics (e.g., statistical properties and resource usage) of simulated processes. On the other hand, DL-S models typically do not provide this flexibility. These models are often seen as black boxes, with no options for users to influence the simulation results, relying solely on random sampling methods to draw predictions from the probability distributions outputted by the

model.

Therefore, we motivate the use of deep learning for process simulation due to its successful accomplishments in other process mining tasks, and we address the problem associated with the lack of flexibility of current DL-S solutions for process simulations [47]. More specifically, in this Chapter, we describe *CoSMo* (**Co**nditioned Process **S**imulation **Mo**dels), a novel methodology to introduce user-directed controls and guidance within these models. Such control is crucial for analysts, researchers, stakeholders, etc., to address questions like "what is the outcome of my process if a condition C is met?" and "what would be the impact of changing the flow of some processes?". Thus, CoSMo advances the fields by introducing a new bridge between advanced process simulation capabilities using deep learning and practical usability needs.

The contributions of this Chapter are outlined as follows: (i) a novel conditioned recurrent architecture tailored to capture the relationship between user-defined constraints and the sequential nature of events; (ii) the introduction of a new framework, **Co**nditioned Process **S**imulation **Mo**del (CoSMo), which implements the conditioned recurrent architecture using any type of constraint or a-priori knowledge; (iii) the validation of this approach through experiments considering both control-flow and data-flow perspectives. CoSMo has proven its capability to simulate traces in compliance with constraints represented by the DECLARE language model [121], providing users with significant control over their generative process models and effectively overcoming the stochastic limitations inherent in traditional DL-based process simulation models.

The proposed methodology does not fully solve the lack of flexibility in performing what-if analysis using deep learning models, but we see it as a substantial step towards filling this gap. This research has been originally introduced in [112].

4.1 Current State and Limitations of Process Simulation

Process simulation models aim at abstracting details from the event logs in order to simulate reality [161]. They are employed as a tool by the PM community for several applications, such as conformance checking [139], event log generation [22, 28, 64], purposed-oriented event log generation [24], what-if analysis [47, 30, 95, 82], and process monitoring [103].

DD-S models are foundational in process mining, built upon process models discovered from event logs [136, 22, 24]. Notable DD-S methods include PGL_2 [22] and SIMOD [25], which simulate control-flow patterns by adjusting user-defined parameters like the number of gateways and the level of noise. However, these

methods often rely on oversimplified assumptions about resource behavior, neglecting complexities such as multitasking, batching, and resource sharing [51]. This limitation arises from their "control-flow first" approach, which inadequately captures the impact of resource dynamics on simulated processes [82]. To address this, the *AgentSimulator* [82] was proposed as a resource-first approach that enhances DD-S by incorporating realistic resource behaviors.

Alternatively, DL-S models have shown outstanding performance in predictive monitoring tasks like next-event prediction and remaining time estimation [155, 54, 113, 26, 127]. Despite this success, their application in process simulation is limited due to their stochastic nature and difficulty in adapting to what-if scenarios [28]. DL models often generate traces that follow the statistical properties of observed data but may not reflect underlying business rules, leading to unrealistic simulations [28]. This has led to their use primarily as generative models or in hybrid approaches that combine DL with process mining techniques [30, 103]. In the hybrid setup, the usage of deep learning involves using it as components within a larger simulation framework to predict specific event attributes – a task where DL excels due to its ability to capture complex patterns. By integrating deep learning's predictive strengths with the structural insights from process mining, it is possible to create more accurate and realistic simulations that calibrate both statistical properties and business constraints.

A common characteristic among process simulation models is the stochasticity inserted during simulation. The simulation process requires a sampling step to draw information from the probability distributions, and applying a certain level of randomness helps models produce more realistic simulations [161]. For example, DD-S models are supported by statistical measures extracted from the event logs, such as branch probabilities and activity duration time distributions. On the other hand, DL-S approaches consist of learning the underlying data distributions and drawing event attributes from the probability distributions returned by the learned model. Although this stochastic approach makes simulations more realistic [161, 25], the complete dependence on randomness limits the flexibility of the user to control the simulated behaviors [47]. Furthermore, despite the efforts to incorporate a-priori knowledge or inter-case features to improve the predictive performance [54, 43], the incorporation of such information tailored specifically for what-if analysis is still challenging for DL-based solutions [47].

To illustrate this stochastic problem, let us consider the problem of next activity prediction in PM [127]. The goal here is to always determine the most likely next activity in an ongoing process. Conversely, in process generation or simulation, diversity is key because realistic process traces typically vary. Relying solely on the most likely sequence results in monotonous, identical trace variants. Typically, this problem is addressed by sampling the next activity from a probability distribution [128], using techniques such as multinomial sampling [155] or

beam search [54]. However, such stochastic methods reduce the user's ability to precisely control the simulation, especially when performing what-if analyses.

In summary, process simulation is a traditional research field but has gained more attention recently. On one hand, we have DD-S models that outstand at providing flexibility and interpretability but fail at generalizing to unseen trace variants. On the other hand, we have DL-S models that struggle with what-if analysis due to their intrinsic lack of flexibility but excel at generalizing to new, stochastic process behaviors. Thus, to address the inherent stochasticity in deep learning models, we introduce an innovative recurrent architecture specifically designed to recognize the relationship between the sequential nature of events and various forms of a-priori knowledge, articulated as user-provided constraints. More specifically, we use the DECLARE model to extract semantics from event logs and reuse them as constraints for learning and compliance. Our methodology is informed by previous research that has effectively used the DECLARE language to enrich or encode training data [54, 43]. By extending these concepts to the simulation domain, our approach provides users with greater control over their generative process models and effectively mitigates stochastic challenges. This strategy not only aligns with the current literature but also pushes the boundaries of conventional modeling techniques, promising significant advances in the field. In the following Sections, we further describe our proposal as well as the experimental evaluation and validation of our approach.

4.2 Fundamental Concepts

Process simulation aims to experiment with new process scenarios under different settings and configurations without harming actual operations. Thus, **process simulation models** are defined as a set of techniques that aim at predicting possible behaviors of a process [161]. This capability proves invaluable for predicting outcomes, identifying bottlenecks, and testing improvement strategies before implementation (a.k.a., what-if analysis). Advanced simulation techniques incorporate stochastic elements to account for variability and uncertainty, providing a more realistic view of process behavior.

Definition 11 (Process Simulation Model)**.** *It is originally defined as a tuple $M = (N, P)$, where N is a discovered imperative process model (e.g., a BPMN model) and P is a set of parameters needed to define the simulation specifications for the different process perspectives, such as the control-flow, the resource, and time perspectives.*

The definition of parameters P can be manually defined based on expert knowledge [136] or automatically extracted from the event log [25]. Examples of

parameters include resource availability, processing times, and arrival rates. The manipulation of such parameters allows stakeholders to generate simulated event logs and observe how changes might affect process performance. This definition was originally proposed to describe DD-S models but it can be further extended to DL-S as well if we treat P as the set of constraints to be satisfied. In a nutshell, DD-S models see P as parameters derived from the data which will be used to generate a simulated value based on stochasticity or another domain-specific heuristic; whereas DL-S sees P as context to guide the simulation, illustrated in this work as constraints.

Simulation under the DL-S setup is formulated as multi-target (see 6) and autoregressive 7 tasks, where the objective is to generate sequences of events that mimic the execution of real-world business processes. This approach involves three main phases: training, validation, and inference (simulation). During the training and validation phases, the event log data is partitioned into training and validation sets, adhering to standard machine learning practices. The deep learning model is trained to predict the next event in a sequence based on the history of preceding events. Hyperparameters are tuned to optimize performance metrics like prediction accuracy and to ensure convergence of the learning algorithm. In the inference phase, also known as the simulation phase, the trained model generates new traces. Simulation can start either from scratch, using an initial random input, or from an ongoing process execution. The process proceeds in a time-step-wise manner: at each time step t, the model predicts the next event e_t, which comprises an activity label and possibly additional attributes (i.e., multi-target) such as timestamps, resources, or other contextual features. This predicted event is then appended to the input sequence to form the input for the next time step (i.e., autoregression), enabling the model to iteratively generate subsequent events.

The DD-S definition takes into consideration an imperative process model and a set of parameters. In this Chapter, we intorduce a DL-S solution that also relies on discoverying a process model, but in our case, a declaritive model is employed. Such model is used solely to extract rules that will be used as additional features for training a deep learning model. The intuition behind this approach is to train a model that learns how to satisfy these rules, whereas at inference/simulation step these rules/features can be manually manipulated to generate a desired trace. The formal definition can thus be described as below.

Definition 12 (DL-S Model)**.** *Similarlyl, the DL-S model can be defined as a tuple* $M = (N_{learned}, E, P_{rules})$*, where* $N_{learned}$ *is a deep learning model trained on an event log* E *and on a set of rules* P_{rules} *extracted from a declarative process model discovered from* E*.*

Finally, let us introduce the declarative model employed in this worrk in the

following Section.

4.2.1 DECLARE Model

Declarative algorithms identify temporal relationships between activities, expressing them as logical constraints. This results in models that specify what must or must not happen in a process, rather than describing exact sequences of activities. The process modeling language DECLARE [121] is based on Linear Temporal Logic over finite traces (LTL_f) [60], which is used to express process semantics. A DECLARE model is defined by a set of activities and constraints on these activities. These constraints are formalized using LTL_f formulae. DECLARE utilizes a subset LTL_f formulae, embodied in predefined patterns known as DECLARE templates. Thus, each template is an LTL_f formula containing placeholders, which are replaced by specific activities to form a constraint. For example, considering the template *Choice* and its constraint $ExclusiveChoice(a, b)$, where a and b are arbitrary activities, it is stipulated that either a or b must occur at least once, but not both. In the following, we will better describe this model in more details by providing formal definitions.

Definition 13 (DECLARE Model)**.** *A DECLARE model N is defined as a set of constraints $N = \{\phi_0, ..., \phi_m\}$, where each constraint ϕ is a LTL_f formula. The satisfaction of ϕ in t at position i is denoted by $t, i \models \phi$. The notation $t \models \phi$ is used as a shortcut to indicate that the formulae ϕ holds over the entire trace t starting from the beginning. Thus, ϕ is satisfiable if it admits at least one trace t such that $t \models \phi_0 \wedge ... \wedge \phi_m$ for each $t \in L$.*

Thus, a DECLARE model can be formalized as the set of LTL_f formulae N expressing its constraints. Among all the existing LTL_f formulae, DECLARE selects predefined templates, which can be gathered into four main groups according to their semantics. Given the arbitrary activities a and b, the templates can be defined as follows:

- **Existence** ($\mathcal{E}$): given a, this group of templates checks the number of occurrences, its absence, or its position in the trace;
- **Choice** ($\mathcal{C}$): given a and b, this group of templates check if a or b occurs, or if a and b occur together;
- **Positive Relations**($\mathcal{PR}$): given a and b, this group checks the relative position of the activities. For instance, if a occurs, b eventually occurs;
- **Negative Relations**($\mathcal{NR}$): given a and b, this group checks if the activities do not occur together or do not occur in a certain order. For instance, if a occurs, b never occurs.

Following the notations presented by Donadello et al. [43], the complete set of DECLARE templates is defined as follows.

Definition 14 (DECLARE templates.)**.** *The set of templates is defined as the union of every template* $\mathcal{DT} = \mathcal{E} \cup \mathcal{C} \cup \mathcal{PR} \cup \mathcal{NR}$.

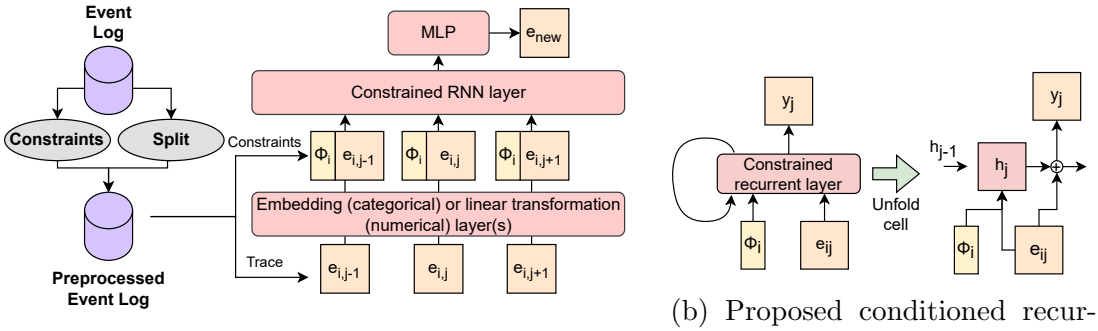
Each of these templates is composed of several rules related to their respective semantics. Table X describes all these rules. We do not explicitly describe the LTL_fsemantics but we refer to Donadello et al. [43] for more details.

Table 4.1: LTLf semantics and textual description of the declare templates.

Family: Template	**LTLf Semantics**	**Description**
$\mathcal{E}$: existence (n, A)	$F(A \wedge X(\text{existence}(n-1, A)))$	A has to occur at least n times.
$\mathcal{E}$: absence (n + 1, A)	$\neg\text{existence}(n+1, A)$	A has to occur at most n times.
$\mathcal{E}$: exactly (n, A)	$\text{existence}(n, A) \wedge \text{absence}(n+1, A)$	A has to occur exactly n times.
$\mathcal{E}$: init (A)	A	Each case has to start with A.
$\mathcal{C}$: choice (A, B)	$FA \vee FB$	A or B have to occur at least once.
$\mathcal{C}$: exclusive choice (A, B)	$(FA \wedge \neg FB) \vee (\neg FA \wedge FB)$	A or B have to occur at least once but not both.
$\mathcal{P}_R$: responded existence (A, B)	$FA \rightarrow FB$	If A occurs, B must occur as well.
$\mathcal{P}_R$: response (A, B)	$G(A \rightarrow FB)$	If A occurs, B must eventually follow.
$\mathcal{P}_R$: alternate response (A, B)	$G(A \rightarrow X(\neg AUB))$	If A occurs, B must eventually follow without any other A in between.
$\mathcal{P}_R$: chain response (A, B)	$G(A \rightarrow XB)$	If A occurs, B must occur next.
$\mathcal{P}_R$: precedence (A, B)	$(\neg BUA) \vee G(\neg B)$	B can occur only if A has occurred before.
$\mathcal{P}_R$: alternate precedence (A, B)	$(\neg BUA) \wedge G(B \rightarrow X((\neg BUA) \vee G(\neg B)))$	B can occur only if A has occurred before, without any other B in between.
$\mathcal{P}_R$: chain precedence (A, B)	$G(XB \rightarrow A)$	B can occur only immediately after A.
$\mathcal{N}_R$: not responded existence (A, B)	$FA \rightarrow \neg FB$	If A occurs, B cannot occur.
$\mathcal{N}_R$: not response (A, B)	$G(A \rightarrow \neg(FB))$	If A occurs, B cannot eventually follow.
$\mathcal{N}_R$: not precedence (A, B)	$G(FB \rightarrow \neg A)$	A cannot occur before B.
$\mathcal{N}_R$: not chain response (A, B)	$G(A \rightarrow X(\neg B))$	If A occurs, B cannot occur next.
$\mathcal{N}_R$: not chain precedence (A, B)	$G(XB \rightarrow \neg A)$	A cannot occur immediately before B.

4.3 A Framework to Condition Process Simulation Models

In this Section, we introduce CoSMo, our proposed framework for implementing process simulation models based on user-based constraints or any other type of a-priori knowledge available. In a nutshell, given an event log, the instantiation pipeline of our framework consists of a preprocessing phase, where the user will split and extract their custom constraints, and train our proposed conditioned recurrent network. This pipeline is depicted in Figure 4.1a and it will be better described in this Section. We first introduce the recurrent network tailored to learn how to generate data based on constraints, then we explain how this network



(a) CoSMo framework overview.

(b) Proposed conditioned recurrent layer.

Figure 4.1: Proposed CoSMo framework (left) and the proposed recurrent architecture (right) tailored to capture the relation between user-based constraints and processes.

can accept any constraint, and finally, we illustrate the example of conditioning the simulation of processes based on LTL_f rules.

In this Chapter, we use declarative language to express user-based constraints under the proposed framework. However, CoSMo can be extended straightforwardly to any sort of constraint. To do so, let us provide a formal definition. The constrained simulation of processes with CoSMo is built upon the definition of autoregression for generative tasks. More specifically, our proposal is an extension of conditional probability under the autoregression setup (see definition 7), which takes into consideration previous observations, previous predictions, and external context illustrated as constraints. Thus, the task of predicting the next event of a trace based on user-defined constraints $\mathcal{C}$ can be formally defined as the conditional probability $P(e_i \mid e_{<i}, \mathcal{C}) = P(e_i \mid e_0, e_1, \ldots, e_{i-1}, \mathcal{C})$. Hence, we generalize the definition as follows.

Definition 15 (Constrained Simulation of processes)**.** *The constrained simulation of a process execution can be defined as the joint conditional probability of an entire trace t, which is seen as the product of the conditional probabilities:* $P(t) = \prod_{i=0}^{s} P(e_t \mid e_{<t}, \mathcal{C})$.

In the following, we further describe how to model this joint conditional probability by designing a new recurrent cell that better captures the relationship between constraints and process executions.

4.3.1 Conditioned Recurrent Architecture

Our proposed architecture starts by taking a sequence of events x as input, where each event element is composed of different attributes, as described in Section 4.2.

Alongside this sequence, the architecture is introduced with external feature vectors c representing a set of constraints. We extend the vanilla RNN by introducing a new learnable parameter K to capture the relation between the condition Φ and the sequential input, followed by a residual connection between each recurrent cell:

$$\begin{aligned} h_j &= f(Ux_j + Wh_{j-1} + K\Phi) \\ y_j &= g(Vh_j) + x_j \end{aligned} \tag{4.1}$$

Where h_j is the hidden state at time step j. This represents the "memory" of the network up to that point in time. On the other hand, U, W, and V are the learnable parameters from the vanilla RNN, and each plays a distinct role in transforming the input and hidden states. The parameter K specifically learns how the constraint feature vector Φ relates to the input sequence at every timestep. The key for the proposed recurrent network is that these condition vectors are external to the primary sequence but highly relevant to the task at hand. In simple terms, a basic RNN learns a hidden state for each event in the sequence. Our model introduces extra parameters, K, which act like an additional hidden state. This extra state learns the relationship between the set of rules at each time step and its corresponding hidden state. Finally, g is a non-linear activation function, essential for enabling the network to capture complex patterns and the residual connection at y_j helps to improve and stabilize the learning phase. This residual technique is common in other deep learning applications and modern recurrent architectures, such as LSTMs, have a similar characteristic (e.g., the forgetting gate mechanisms). The bias terms are omitted for simplicity. We depict the new architecture in Figure 4.1b.

Ultimately, given an input sequence x and a constraint vector Φ, our proposed backbone layer learns the conditioned probability $p(y|x, \Phi) = \hat{y}$ to predict the next attribute of that sequence. Theoretically, these constraint features can take any form, such as generic user-based constraints, intra-case features describing the availability of shared resources at that timestamp, or declarative rules based on LTL_f. Subsequently, we dive into the latter and demonstrate how the simulation can be guided by enforcing the generation of traces that satisfy different DECLARE templates of LTL_f rules.

4.3.2 CoSMo Instantiation

In this Section, we describe how to instantiate CoSMo by taking into consideration the LTL_f rules extracted from the event log using the set of DECLARE templates $\mathcal{DT}$ described in Section 4.2 and the conditioned recurrent architecture previously described. We highlight that any other nature of constraints can be employed in our framework, but we focus on LTL_f rules in this work.

Consider an event log L of n traces and a group of DECLARE templates, where this group can be represented by dt from the set of DECLARE templates of LTL_f rules described in Section 4.2, i.e., $dt \in \mathcal{DT}$. For every trace t_i in the event log L, the DECLARE model discovers a vector $\Phi_i = [\phi_0, \ldots, \phi_m]$ of m LTL_f rules from dt that are fulfilled for t_i. The original log L can be written as $L = \{t_i\}_{i=0}^{n}$. This log is then augmented to form a new log L', which is defined as $L' = \{\langle t_i, \Phi_i \rangle\}_{i=0}^{n}$. Here, Φ is represented as a binary feature vector of m dimensions (constraints) that describes if the $\hat{m} - th$ constraint is fulfilled in t_i or not, for $0 \leq \hat{m} \leq m$.

Subsequently, the event log L' is split into train and test sets (L'_{train} and L'_{test}, respectively). The training phase is performed by feeding the network with a trace t_i containing s events $t_i = [e_0, ..., e_s]$ and its respective constraint vector Φ_i. As depicted in Figure 4.1a, the Φ_i vector is concatenated to each event vector $e_{i,j} \in t_{j=0}^{s}$ in order to enforce the model to learn the relation between the set of rules and the sequential nature of events. The network is trained to predict the next event e_{new} and the loss function is computed by comparing the predicted event attributes with the ground truths. This process constitutes a multitasking framework (see definition 6), especially when there are multiple event attributes to be predicted. Here, we focus on the process simulation by predicting in an autoregressive fashion the next activity label and the remaining time of the trace based on a set of constraints.

4.3.3 Conditioned Simulation

After instantiating and training the CoSMo model, we initiate the conditioned simulation of new traces. This process begins by selecting constraint vectors, Φ_{test} from the test dataset L'_{test}. For clarity, we will drop the 'test' subscript in this explanation. The core idea of our approach is to modify specific LTL_f rules, ϕ, within the set Φ, thereby directing CoSMo to create new traces based on these modifications. The goal is to control the process behavior by specifying which rules are to be fulfilled or not in the generation of a new trace. Thus, from the set Φ we select a subset of rules $\Phi' \subset \Phi$, and invert their values to condition the simulation of process behaviors. This selective modification, as opposed to altering all rules, is crucial to maintaining a diversity of trace behaviors; otherwise, using the same full set Φ would result in repetitive, identical behaviors. For instance, to enforce the existence of an activity a, an arbitrary subset Φ', which is defined as a binary vector, e.g. $\Phi' = [existence(a) = 0]$, has its value inverted. Therefore, after inverting the values from Φ' and given the preserved subset of rules $\Phi^p = \Phi - \Phi'$, we design a new set of constraints $\Phi^s = \Phi^p \cup \Phi'$ to perform our conditioned simulations. Starting with an arbitrary initial event e_0, we autoregressively generate subsequent events, each conditioned on the set Φ^s,

thereby simulating the process according to these newly defined constraints.

Example. Consider an event log composed of traces $L = \{< a, b, c >, < a, a, c >\}$ and the instantiated (trained) CoSMo framework. Considering the usage of the Existence template, assume the DECLARE model identifies the following constraints: $\Phi = \{\text{Existence}(a), \text{Existence}(b), \text{Existence}(c)\}$. In practice, Φ is represented as a matrix $n \times m$, where n is the number of traces and m is the number of found constraints. For example: $\Phi = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$. Here, 1 indicates that the constraint is met, and 0 indicates that it is not. Next, let us define the subset of constraints $\Phi' = \{\text{Existence}(c)\}$ to be inverted. Thus, the final set of constraints Φ^s to be simulated will be: $\Phi^s = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$, where the second column values are inverted. Given an arbitrary first event $e_0 = [a]$, let us assume the model needs to generate the next event e_1 based on the arbitrary constraints $\phi = [1, 1, 0]$ (in simple terms, a and b should exist, whereas c should not). The generated e_1 will be concatenated to the input sequence and this new sequence will be used to generate the next event e_2. Each of these events will always consist of an activity label and its execution time. The model continues this autoregressive process, adding each generated event to the input trace and predicting the next event based on the imposed constraints. The stop criterion consists of defining the maximum trace length or when a specific event label is generated. Examples of valid traces that satisfy the set of constraints include $< a, b >$ and $< a, a, b >$, among others.

4.4 Evaluation

In the following Section, we delineate our experimental framework, illustrate its application through a use case involving real-world event logs for the assessment of our proposed methodology, and conclude with an analytical discourse on the prevailing limitations and strengths of our approach.

4.4.1 Experimental Setup

Event logs. We employ five distinct real-world datasets, each with unique characteristics, sourced from the 4TU.ResearchData repository. To ensure a thorough yet focused evaluation, we attempt to choose datasets of varying characteristics (number of unique activities, number of events, a trace length) – small, medium, and large, illustrated in Table 4.2. Some of these event logs have different versions. In this work, more precisely, we use the complete *BPI12*[1], the problems version

[1] `https://data.4tu.nl/articles/_/12689204/1`

of *BPI13*[2], and the travel permit subset of *BPI20*[3]. The *BPI17*[4] and*SEPSIS*[5] datasets are used in their entirety.

Event Log	Acts.	Evts. (10^3)	Cases (10^3)	C. Length
BPI12	23	104.82	9.49	11.05 ± 9.64
BPI13	6	4.89	1.33	3.69 ± 4.09
BPI17	26	1210.81	31.50	38.44 ± 17.96
BPI20	50	82.22	6.85	12.01 ± 5.46
SEPSIS	8	9.87	0.95	10.37 ± 3.9

Table 4.2: Event logs and their properties: number of unique activities, number of events, number of cases, and average case length.

Preprocessing. Our preprocessing phase is structured into the following stages: initial data cleaning, extraction of a-priori knowledge, and then dividing the data into training and testing sets. This initial stage involves removing traces that are either shorter than three events or longer than the 90th percentile in the trace lengths' distribution, a strategy that ensures a more even distribution of sequence lengths. Following the cleaning process, we use the Declare4Py library [4] to extract the a-priori knowledge by discovering a DECLARE model from the event log, utilizing the discovered LTL_f rules as constraints to feed our proposed deep neural net. For the datasets *BPI12*, *BPI17*, and *BPI20*, we employ the benchmarked splits [178], while for the *BPI13* and *SEPSIS* datasets, the data is split using a publicly available implementation from the popular PM4PY library[6].

Architecture design and training. We introduce a conditioned recurrent network model that integrates categorical and numerical event attributes, enriched with preprocessing constraints, as its input (see Figure 4.1). A Multi-Layer Perceptron (MLP) is employed upon the proposed conditioned backbone, and it operates at the final stage to generate new events, symbolized as e_{new}. The model performs multitasking by autoregressively generating multidimensional events, utilizing previous predictions of activity labels and execution times to forecast subsequent labels and times. Specifically, it predicts the next activity label and the remaining time, where execution time is the interval between the current and previous event timestamps, and the remaining time is the expected duration to complete the process. Model evaluation employs cross-entropy and mean squared error for loss functions, addressing categorical and numerical predictions respectively. Hyperparameter tuning involved grid search

[2]`https://data.4tu.nl/articles/_/12688556/1`
[3]`https://data.4tu.nl/articles/_/12718178/1`
[4]`https://data.4tu.nl/articles/_/12696884/1`
[5]`https://data.4tu.nl/articles/_/12707639/1`
[6]`https://pm4py.fit.fraunhofer.de/`

for optimal learning rates $[5e-3, 1e-3, 5e-4, 1e-4]$, number of epochs $[50, 100]$, batch sizes $[16, 32, 256, 512]$, and recurrent layer configurations including input sizes $[16, 32, 64, 128, 256]$, hidden sizes $[32, 64, 128, 256, 512, 1024]$, and layer counts $[1, 2, 4]$. We do not include the complete discussion on the hyperparameter tuning results, but we report the best configuration. The Python implementation is available in our repository [7].

Simulation. Building upon the methodology outlined in Section 4.3, we initiate by selecting the constraints Φ_{test}, represented in our study as groups of DECLARE templates. To condition the simulation, we then identify a subset Φ' from these constraints to be inverted. Given the frequency distribution of the set of rules, a pair of activities (or a single activity when the template is the Existence) is drawn from it taking into consideration only the rules lying between the 1st and 3rd quantile of that distribution. By setting this boundary, we aim at not selecting rules with low variance in order to accomplish greater generalization. Despite the arbitrary intuition behind this selection, in the sense of business-driven choices, we propose this data-driven approach to ensure a fair evaluation across all event logs and eliminate the need for an expert to define the rules to be simulated for each event log. Although this is a clear limitation in real-world scenarios, it effectively validates our scientific methodology.

Subsequently, in order to ensure consistency in our evaluation across all event logs, we uniformly select the same DECLARE templates. As previously mentioned, these rules are expressed as vectors of binary values, where '1' indicates rule fulfillment, and '0' denotes non-fulfillment. Our primary aim is to invert one specific subset of rules to 'enforce' desired process behavior. However, since we are using all the available DECLARE templates, we must ensure that the rules are not contradictory. For instance, inverting the 'existence' rule while maintaining the 'absence' rule could create contradictory conditions, leading to a conflict in the model's inference process. We perform a careful selection of templates to avoid any inconsistencies: we do not include the $\mathcal{NR}$ template group in our evaluation, as it is inversely related to the $\mathcal{PR}$ group, in general. Such meticulous template selection ensures coherence and efficacy in our model's simulation capabilities. Thus, if we want to ensure the 'existence' rule, the 'absence' must be set as the opposite as well.

Therefore, for each group of templates, we pick the following rules as references: the Existence(a) regarding the group $\mathcal{E}$, the Exclusive choice(a,b) regarding the group $\mathcal{C}$, and the Chain Response(a, b) regarding the group $\mathcal{PR}$. Subsequently, the following rules are also picked to avoid the aforementioned inconsistency issue: (i) for group $\mathcal{E}$, the templates Absence(a) and Exactly(1, a) are chosen; (ii) for group $\mathcal{C}$, no extra template is selected; and (iii) for group

[7] `https://github.com/raseidi/cosmo`

$\mathcal{PR}$, we select the Alternate Response(a, b), Precedence(a, b), and Response(a, b) templates. The remaining rules are preserved (Φ^p) and the picked ones are inverted (Φ') to compose the final set of constraints $\Phi^s = \Phi^p \cup \Phi'$ to be simulated. Each of these templates and its details is described in [43]. We selected the activities based on their frequencies (we picked the 3rd quartile), and the activities incorporated into each group of templates for each event log are described in Table 4.3.

Event Log	Template	Activity A	Activity B
BPI12	Existence	A_DECLINED	-
	Choice	W_Completeren aanvraag	W_Nabellen offertes
	Positive Relations	A_PARTLYSUBMITTED	W_Completeren aanvraag
BPI13	Existence	Queued-Awaiting Assignment_complete	-
	Choice	Accepted-Assigned_complete	Queued-Awaiting Assignment_complete
	Positive Relations	Accepted-In Progress_complete	Queued-Awaiting Assignment_complete
BPI17	Existence	W_Handle leads	-
	Choice	A_Submitted	A_Cancelled
	Positive Relations	O_Accepted	A_Pending
BPI20	Existence	Permit APPROVED by BUDGET OWNER	-
	Choice	Declaration APPROVED by BUDGET OWNER	Declaration SUBMITTED by EMPLOYEE
	Positive Relations	End trip	Send Reminder
SEPSIS	Existence	LacticAcid	-
	Choice	Admission NC	LacticAcid
	Positive Relations	Admission NC	CRP

Table 4.3: DECLARE templates and their respective activities employed to perform the conditioned simulation.

Evaluation and Validation. Upon generating a simulated event log under these inverted constraints, we employ conformance checking to verify the fulfillment of the modified rule set. In this evaluation, we split the traces between two classes to quantify the effectiveness of the simulation: one where the reference rule must be fulfilled and the other where it must not. For instance, a trace where the reference rule must be fulfilled (e.g., Existence(a)=1) will be a true positive if the simulation conforms to the rule, otherwise, it will be a false negative. The performance is then reported through the calculation of weighted precision, a metric chosen specifically for its ability to provide nuanced insight into the degree to which the simulated traces adhere to or deviate from their intended behavior. The use of weighted precision is particularly appropriate in this context because it allows for a nuanced assessment of each trace's adherence to the imposed rules, recognizing the varying importance of different rules within the process, thereby providing a more granular and meaningful measure of CoSMo's effectiveness and ability to replicate or alter specific process behaviors as needed.

In addition, we include in this evaluation two baselines against the aforementioned inverted subset. First, we compare CoSMo with this perturbed set of constraints (Φ^s) as input to CoSMo with the original set of constraints (Φ) as

input, which is the simplest way to assess whether the model actually changes or maintains the input set of rules. Simulating the *Original* and *Inverted subset* of rules is an alternative to performing an ablation study, which is a common practice in the machine learning literature to assess the impact of a particular component in a model. In practical terms, by providing the *Original* set of rules, we expect to validate the model's capability to learn the relationship between those rules and the process semantics. On the other hand, the *Inverted subset* allows us to validate the model's capabilities to generalize. More specifically, this approach could be thought of as the changes that we aim to do in a process, which could characterize a what-if scenario. Second, we compare CoSMo to a *Vanilla* simulation without any constraints, which is the most common way to stochastically generate processes using deep learning in the literature [28]. The intuition behind this comparison is to assess whether the model is actually learning the constraints and whether the constraints are affecting the simulation. We use the architecture introduced in [155] as a vanilla baseline, since it overcomes other approaches from the predictive monitoring literature [127]. We run each simulation 10 times to ensure the robustness of the results since the stochastic nature of the model can lead to different results in each run.

We then discuss the capabilities and limitations of adapting complex LTL_f constraints by taking into account data and process characteristics. Finally, we present the remaining time prediction performance through the Mean Absolute Error (MAE) of the test set obtained during training, a common metric in the predictive monitoring literature [127].

4.4.2 Conditioned Simulation of Process Behaviors

Precision evaluation. Figure 4.2 illustrates the weighted precision of the simulated logs under the different approaches. The *Vanilla* is the fully stochastic model without any constraints, the *Original* is the model with the original set of constraints, and the *Inverted subset* is the model with the inverted set of constraints. The weighted precision is the average of the precision calculated for each class of conformance, i.e., if the rule should be fulfilled or not. The intuition of employing the *Vanilla* model is to check if and how much of the original set of rules is being kept. We can notice that for any template, this model performs quite randomly, since it keeps a median precision of 58% across all the event logs, highlighting its stochastic nature. The *Original* model, on the other hand, illustrates the capability of the model to learn the original set of rules, with a median precision of 99% across all the event logs. This approach can be considered a sanity check since we are basically replaying the original traces at position 0 and their respective set of rules and checking if it can preserve the original semantics of the input processes.

Although the *Original* simulation is capable of fulfilling the imposed rules, we can notice that not all the set of inverted rules are being fully fulfilled by observing the *Inverted Subset*. This is particularly expected since the model is not trained to fulfill the inverted set of rules, but the original set of rules. However, the achieved precision is remarkably high in general, with a median of 81% across all the event logs. This enforces the generalization capability presented by neural networks, which can learn the original set of rules along with the event attributes and apply them to a different set of rules, even if they differ from the original set of observed rules. This is particularly important for the process simulation since it is not always possible to know the set of rules that will be imposed on the model in the future, and the model should be able to generalize the learned set of rules to a different set of rules. However, for some templates and event logs, we can notice a higher difficulty in generalizing. For instance, in the templates, $\mathcal{C}$ and $\mathcal{PR}$ on the *BPI17* event log, the *Original* approach has a slightly lower precision compared to the other logs, which is drastically reflected on the inverted set of rules. This clearly identifies the difficulty of learning the rules and, hence, the inability to generalize to the unseen combination of event attributes and inverted rules.

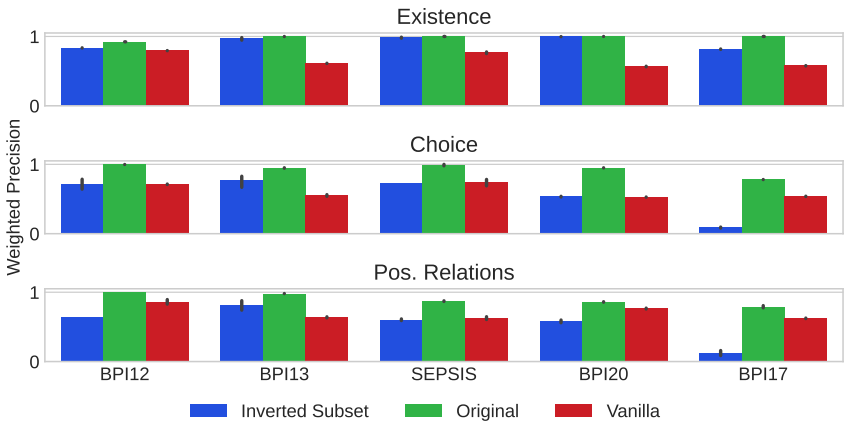


Figure 4.2: Weighted precision for the simulated logs under the different approaches and DECLARE templates.

Furthermore, we can notice that some templates are harder than others. Table 4.4 shows the average precision for each class of conformant rules, i.e., to be or not to be fulfilled, for each template regarding the *Inverted* simulation. We can notice that the precision for rules that should not be fulfilled is higher and this behavior is due to a class imbalance issue, where the not fulfilled corresponds to the majority class in most cases. More precisely, in our experimental setup, across all the event logs and templates, the number of traces corresponding to

the set of rules to be fulfilled is 285 ± 8 total rules, while the set of rules to be not fulfilled corresponds to 1000 ± 200 total traces. Finally, the difference in performances among the templates is intuitive due to the nature of each of those templates, where some of them are more complex than others due to their higher dimensionalities (i.e., more rules ϕ), making it more difficult to be learned by the model. We extend this discussion in the following.

Table 4.4: Precision for each class of rules.

Template	Fulfilled	Not fulfilled
Existence	0.912 ± 0.2	0.917 ± 0.18
Choice	0.807 ± 0.28	0.9048 ± 0.15
Pos. Relations	0.6468 ± 0.29	0.9053 ± 0.12

Log and DECLARE templates characteristics. The difficulty of learning to satisfy imposed constraints can be also reflected in the number of constraints for each event log under a certain template. We illustrate the number of constraints obtained by each template and for each event log in Figure 4.3. More specifically, Figure 4.3a shows the number of constraints for each event log under different templates, whereas Figure 4.3b shows the distribution of the number of constraints for each template. We can notice that the $\mathcal{PR}$ template is the one with the highest number of constraints, which results in a higher dimensional feature vector for the conditioned simulation. Intuitively, the higher the dimensionality of the feature vector, the harder it is for the model to learn the set of rules, and the harder it is for the model to generalize the learned set of rules to a different set of rules. Additionally, in Figure 4.2 the *BPI17* had the poorest performance, which is reflected in the higher number of constraints for the $\mathcal{PR}$ template, as depicted in Figure 4.3a. Furthermore, despite the similar characteristics with the *BPI20*, the *BPI17* is even more complex due to its longer traces. It is well known by the deep learning community that longer sequences are harder to learn and generalize, specifically for recurrent neural networks.

Besides the log characteristics, another factor that impacts the ability of our proposal to generalize to different sets of rules is the selected activities to simulate. Recall that the activities illustrated in Table 4.3 were selected based on the rule frequencies, which can result in a bottleneck. Another characteristic observed from the process models is that the *End Tripe* and *Send Reminder* activities are far away from each other in the *BPI20*, which also indicates a more complex semantic to be captured.

Data-flow. As a final discussion, we illustrate the model capability of autoregressively simulating the remaining time along with the activities under the imposed constraints versus the vanilla approach. The performance of the remaining time (RT) is illustrated as the loss function (MAE) in Figure 4.4. Due to the limited space, we are illustrating only three event logs with varying charac-

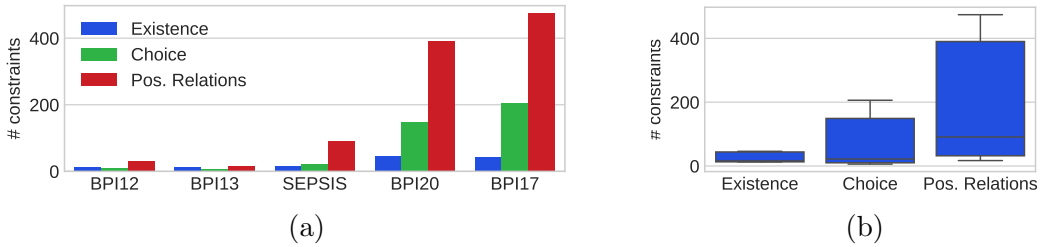


Figure 4.3: Analysis of event log constraints from two perspectives: (a) count per log across templates, (b) distribution per template.

teristics: the *BPI13*, the *BPI12*, and the *BPI17*, respectively illustrating a small, medium, and large event log. We can notice a slower convergence for CoSMo and this is due to the fact we are increasing the complexity of the model by including the constraints to be learned. However, around 20 epochs both the Vanilla and CoSMo can converge and achieve a similar performance. We are not diving into this discussion but in the future, we plan to derive the timestamps from the predicted remaining time and employ relevant metrics to better assess the quality of these simulations. Moreover, it is not novel that neural networks perform very well for this specific task, but we include this brief discussion to ensure that the remaining time prediction also performs well under the constrained scenario proposed in this work.

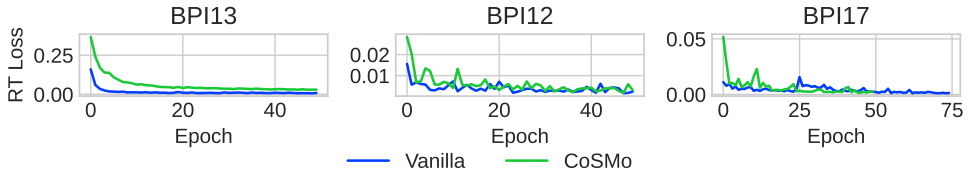


Figure 4.4: Data-flow of the proposed CoSMo framework.

4.5 Final Remarks

In this Chapter, we introduced an innovative methodology for process simulation using deep learning by moving beyond the conventional stochastic generative models, focusing instead on an architecture that integrates extensive constraints and a-priori knowledge for a nuanced understanding and simulation of data patterns, diverging from mere sequence prediction. We successfully answered our RQ2: "How can modern deep learning techniques enhance traditional heuristic

based process simulation to improve the accuracy and reliability of what-if analysis?" Unlike conventional deep learning solutions [28], which are predominantly stochastic generative models (as elaborated in Section 4.1), our approach takes a step towards overcoming this limitation. Existing methods focus primarily on sequence prediction, which is often similar to, but not synonymous with, simulation. In contrast, our proposed architecture is tailored to incorporate a wide range of constraints or a-priori knowledge, facilitating a more sophisticated and complex understanding of data patterns and dependencies. We adopt conformance checking as an evaluation method to ensure the generated sequences adhere to specific LTL_f semantics, offering a meaningful assessment of compliance and success in scenarios demanding strict policy adherence. In contrast, a key innovation in CoSMo's inference stage is its ability to generalize and adaptively manipulate process changes through the constraint vector, represented in this study by LTL_f rules. This feature represents a significant step toward addressing the stochastic nature inherent in deep learning models and incorporating critical information into the learned model to allow more sophisticated evaluations (e.g., what-if analysis). Such capabilities are not only essential for PM practitioners and stakeholders but are also ideal for complex situations where a deep understanding and adherence to an extensive set of rules or a-priori knowledge is essential.

In a nutshell, we train recurrent networks to learn how to predict the remaining suffix along with their original set of rules. Subsequently, at simulation time, we manipulate such rules to check if the model is capable of generalizing to the modified set of rules. In recent literature, two similar approaches have been introduced as competitors of our approach. First, Barón-Espitia et al. [9] leverages DECLARE constraints to enable targeted scenario generation while minimizing deadlocks through LSTM-based trace generation. Second, Graziosi et al. [64] adopt Conditional Variational Autoencoders to generate diverse traces conditioned on binary constraints, ensuring compliance with process rules. While both methods bring unique strengths, CoSMo theoretically provides the same capabilities plus the flexibility to adapt to any user-defined constraint. In the future, we intend to perform a systematic comparison of these proposals to assess their practical boundaries.

It is also notable that another recent contribution, GEDI [99], aims to systematically generate benchmark datasets with specific characteristics for evaluation purposes. The primary distinction between this approach and ours lies in their objectives: while CoSMo ensures realistic, constrained event logs for process simulation, GEDI provides a broader and more controlled space for testing process mining algorithms.

While our study has demonstrated promising outcomes, it is important to acknowledge certain limitations to be addressed in future research. First, the ap-

plication of the DECLARE language modeling in our case study highlighted some bottlenecks, notably the high correlation among some rules (e.g., existence and absence rules), which might introduce redundant information into the learning process. To enhance model efficiency and accuracy, future work will explore preprocessing steps to eliminate these correlated features [39], thereby streamlining the feature vector and potentially improving performance on complex templates and datasets. An alternative evaluation will also be included by simulating a smaller set of rules and increasing them accordingly. Secondly, our focus has been predominantly on methodological validation, with less attention to the architecture's applicability in real-world settings. This was accomplished by performing a conformance checking between the simulated log and the imposed constraints. However, more evaluation metrics from the literature must be taken into consideration as well [37]. Moreover, the process of rules and activity selections (see Table 4.3), conducted in a data-driven fashion in this study, would benefit from the involvement of domain experts for each event log in practical scenarios to ensure the relevance and impact of chosen rules on the simulations are accurately assessed. For instance, although we demonstrated strong capabilities to simulate data-aware processes by estimating the remaining time at each time step, more features and outcomes should be considered, for instance, resource usage. Lastly, there is a need to broaden the scope of exploration to include diverse types of constraints and a-priori knowledge, moving beyond declarative rules to incorporate, for instance, user-based constraints (such as time-related constraints to change, for instance, the distribution of waiting time) and intra-case feature availability (e.g., resources).

In the next Chapter, we present a discussion of the findings from this work, highlighting common flaws and limitations in the literature of PPM. We also propose a new open-source tool to help researchers and practitioners conduct more robust studies in the future.

Chapter 5

Sustainable Process Mining: Enhancing Environmental Impact whilst Preserving Predictive Performances

Computational scalability paired with environmental challenges has emerged as a critical research concern in recent years. Industries worldwide are under pressure to adopt sustainable practices, not only to mitigate their environmental impact but also to ensure long-term viability in an increasingly resource-constrained world. In this context, process mining has begun to offer powerful means to identify inefficiencies, optimize resource utilization, and uncover opportunities to reduce emissions and energy consumption [63]. While process mining has been extensively applied to enhance operational performance, its potential to drive sustainability objectives remains vastly underexplored and represents a significant opportunity for both environmental protection and economic efficiency. Notably, recent efforts towards addressing this topic have been shaped, like dedicated workshops[1] and peer-reviewed publications [88, 19, 63].

The urgency of addressing sustainability in computational applications, especially in artificial intelligence, is underscored by the simple reality of climate change. Kaack et al. [77] describe two key actions to be taken in this regard: first, reducing energy consumption through improved efficiency in machine learning models and infrastructure. Second, applying machine learning in ways that directly reduce greenhouse gas emissions, while avoiding or mitigating applications that may increase emissions. The information and communications technology ecosystem has a crucial role to play in this global effort. According to Jones

[1] See the workshop details at `https://pm4s-ws.github.io/workshop/`.

[75], by 2030 it is estimated that the ICT sector could account for up to 20% of global electricity demand, with data centers alone consuming 2% of global electricity. This projection highlights the crucial need for sustainable approaches in computational research and applications.

Meanwhile, the last decade has seen remarkable advancements in artificial intelligence, particularly in deep learning. The breakthrough performance of models like AlexNet [85] on large-scale image recognition tasks, which was facilitated by the efficient training of deep neural networks using GPUs, has aroused the development of innovative architectures such as convolutional neural networks, residual networks, recurrent neural networks, and transformers. These architectures form the foundation of many modern AI applications, revolutionizing several fields.

However, the computational cost of these advancements is increasingly concerning. Training large-scale deep learning models demands substantial computational resources and energy, raising serious sustainability issues. A single training run for an LLM can emit as much CO2 (carbon) as five cars over their entire lifetimes [150]. The training process is not only highly energy-consuming but also involves several experimental iterations that may not lead to successful models, further leading to negative environmental impacts. Addressing these issues is crucial to ensure that the advancements in AI become more sustainable and align with sustainable goals.

Encouragingly, the AI community is reacting to this challenge with innovative approaches. Techniques like transfer learning, synthetic data generation, model compression, pruning, and quantization are examples that can be employed to mitigate high energy consumption and CO2 emissions. These methods not only reduce the environmental footprint of AI but also democratize access to advanced AI capabilities by lowering the computational barriers to entry [159, 138]. Automated machine learning (AutoML) is another promising direction to reduce unnecessary computational expenditure by mitigating trial-and-error in model development, potentially saving significant energy and resources [140].

Regarding process mining, the intersection of this field with deep learning has recently gained attention due to the outstanding predictive capabilities these models bring to the field. Deep learning models can capture complex patterns in large-scale event data, significantly augmenting and enhancing traditional process mining applications [10]. As introduced in Chapter 2 (see 2.4), predictive process monitoring based on deep learning enables the forecasting of future events and outcomes in business processes, facilitating proactive decision-making and prescriptive actions.

Regarding the aforementioned scientific advancements along with the environmental concerns, this Chapter provides a critical discussion of how to maintain sustainability in AI applications within process mining. We ground our discussion upon the following statement: ***reducing computational resources di-***

rectly contributes to energy consumption reduction. Computational performance metrics, e.g., execution time and memory usage, are often disregarded in the literature of process mining. We believe this is due to the fact process mining can be considered an embryonic research field, having its first appearance in the late 90s. Nevertheless, our principle serves as a guide for sustainable AI development in process mining, emphasizing that predictive efficiency and environmental responsibility are intrinsically linked. We perform a methodology to properly develop benchmarks and perform several analyses of how to speed up AI pipelines in process mining for three different tasks: anomaly detection, remaining time prediction, and next activity prediction. We use each of these tasks to discuss how energy consumption can be reduced from different perspectives. Accordingly, the first task has been peer-reviewed in [153], the second one in [111], and the third one is being originally introduced in this thesis.

5.1 Problem Motivation and Methodology

The integration of deep learning into process mining has shown great promise, but its environmental impact has received limited attention. Model sizes, represented by the number of learnable parameters, are directly correlated with the amount of time needed for training (the larger the model, the longer it takes). While small recurrent networks currently dominate successful applications [177, 127], which is beneficial for energy consumption, the field's growth is pushing toward GPU-heavy models. Transformer architectures, though previously overlooked due to limited data in process mining, are now gaining traction as new training strategies and modeling techniques are developed [182], increasing computational demands due to their naturally higher number of parameters. Moreover, several LLM-based applications started to be proposed in the past two years as well [15]. This shift highlights the need to prioritize energy-efficient models and minimize the carbon footprint, reflecting broader concerns in AI research [150].

From another perspective, process mining has also emerged as a valuable tool for promoting sustainability within organizations, although no work has addressed the impact of deep learning in this field. However, process mining tackles challenges such as aligning organizational goals and ensuring proper data management [74]. Although it has significant potential for optimizing business processes with sustainability in mind, more research is needed to fully integrate sustainability metrics [63]. Additionally, combining process mining with formal logic and rule learning can help businesses comply with complex sustainability regulations, offering scalable and efficient solutions [142]. In the context of production networks, process mining has been used to assess sustainability during the early design phase when real-world data is limited. This approach, coupled

with key performance indicators, provides a practical framework for evaluating sustainability, as demonstrated in the automotive industry [86]. Nevertheless, the impact of deep learning combined with process mining for sustainability remains an open area for future exploration.

To introduce the first step toward assessing environmental concerns in process mining using deep learning applications, we propose in this Chapter a methodological approach based on three key ideas, focusing on the direct correlation between computational resources and predictive performances. First, we extend the benchmark presented in Chapter 3 by proposing novel evaluation metrics to assess computational efficiency in process mining tasks. Second, we explore whether simpler process mining techniques can match the predictive performance of advanced deep learning models while being more computationally efficient. Third, we propose a new trace encoding method to improve the efficiency of training deep learning models in process mining without sacrificing accuracy.

More specifically, we evaluate our proposed methodology in these tasks to demonstrate its wide applicability. As in Chapter 3, the benchmark focuses on anomaly detection, where now we discuss our first idea to evaluate the correlation between computational impact and predictive accuracy. For the second idea, we compare the execution time of simpler feature engineering techniques with advanced deep learning methods. By optimizing the preprocessing phase, we aim to reduce computational time and energy consumption while maintaining performance. This involves a systematic comparison of traditional machine learning techniques and deep learning models, using statistical analysis to measure significant differences in predictive performance and efficiency. Lastly, we propose a new trace encoding technique to further enhance the efficiency of deep learning models in process mining tasks.

The goal of this Chapter is to show from different perspectives (tasks) how our methodology can be employed to take into consideration environmental concerns while implementing new AI-based solutions for process mining. Therefore, we dedicate one section for each task and introduce the necessary background fundamentals and experimental design for each of them.

5.2 Finding a Balance between Computational Resources and Predictive Performances

In this Section, we follow a similar experimental setup from Chapter 3 (more specifically, section 3.2). Thus, we only describe the new setup. This analysis aims to provide insights into how a benchmark and tailored metrics are crucial to assess the trade-off between predictive performance and computational performance.

5.2.1 Experimental Setup

We first briefly recall the necessary setup from Chapter 3. Five scenarios were proposed, where for each scenario, we generated event logs by combining three different cardinalities, injecting seven different anomalies (attribute, early, insert, late, rework, skip, and all), at four different rates of injection (5%, 10%, 15%, and 20%). This resulted in 84 event logs for each scenario and 420 event logs in total. We evaluated four big groups of encoding techniques (baseline, graph-, pm-, and text-based) to better understand their behaviors and how performances achieved by these techniques are aligned with the dataset characteristics. Regarding the new setup, we evaluate the scalability of encoding methods in terms of increasing or decreasing the computational cost regarding the elapsed time and the memory usage of encoding methods. The elapsed time is measured from the end-to-end pipeline: preprocessing, training, and inference. Thus, the encoding method should be able to map the event log quickly without compromising the PM pipeline run time. Furthermore, we include the F1-score to evaluate the predictive performance and assess the trade-offs of it with the scalability.

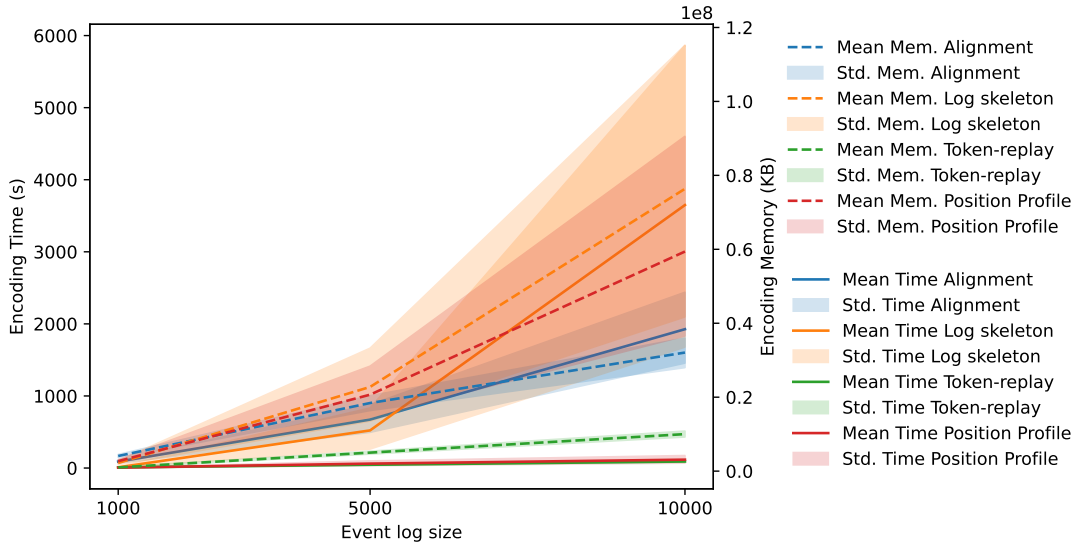
5.2.2 Scalability Analysis

Let us first focus on the time and memory consumption during the encoding process. We analyze three event log sizes (1k, 5k, and 10k) to understand how encoding costs scale when different methods are applied to the same set of problems.

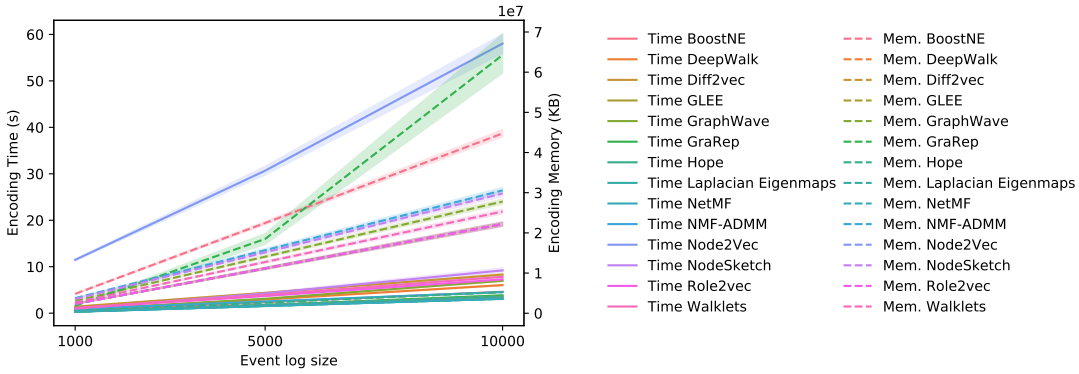
We evaluate computational resources in terms of total time (seconds) and memory (KB) used throughout the encoding process. Keeping track of the costs is important to keep environmental concerns aligned with resource-heavy methods. The choice of a suitable encoding method in terms of sustainability is influenced directly by these factors. In Figure 5.1, we compare two major encoding families: PM- and Graph-based methods. We chose these two groups due to their overall favorable trade-offs. For full details on all encoding methods, please refer to the published benchmark [153]. In the figure, we display the scalability of both time and memory, with the y-axes limited to the highest observed values for each encoding method (time on the left, memory on the right).

The PM encoding family (Figure 5.1a) showed high memory usage with moderate time costs, but it scaled well as the dataset size increased. Among the PM methods, the *Alignment* method had the lowest memory usage, while *Token-replay* offered a good balance between time and memory. However, *Log skeleton* was the most resource-intensive, with high costs in both time and memory. The Graph encoding family (Figure 5.1b) exhibited more diverse memory usage patterns, with average time costs overall. *GLEE*, *Hope*, and *NetMF* were the fastest

methods in this group, all showing moderate scalability. The best scalability in this family was observed in the *NodeSketch* method. On the other hand, *GraRep* had the highest memory consumption, which was comparable to the *Token-replay* method from the PM family, but its time scalability was average. The slowest method in the Graph family was *node2vec*, which took four times longer than the average for smaller event logs and up to seven times longer for the largest event logs. Nevertheless, it presents a low memory usage.



(a) Process Mining encoding family.



(b) Graph encoding family.

Figure 5.1: Time and memory costs across different event log sizes (1k, 5k, and 10k) for both encoding families.

We organized the results of scalability and consumption of both time and

memory analysis as a heat map (Figure 5.2). In the figure, it is possible to observe that general low memory and little time consumption do not reflect the scalability, i.e., increasing event log sizes, some methods compromise their costs with quadratic complexity costs of time and memory (e.g., *count2vec*). Alternatively, *Log skeleton*, *token-replay*, and *NodeSketch* are very scalable but have a high memory and time cost. When considering raw time consumption, it is very evident how the PM family contains the slowest methods, being positioned in the last three points.

This analysis highlights an important insight: execution time and memory usage are not always correlated. The key takeaway from this is that selecting the best encoding method depends on understanding the specific requirements of the environment. For example, in applications where the encoding process needs to be repeated frequently, the method with the fastest execution time should be prioritized. On the other hand, if memory consumption is a greater concern, choosing a method with lower memory usage would be more suitable.

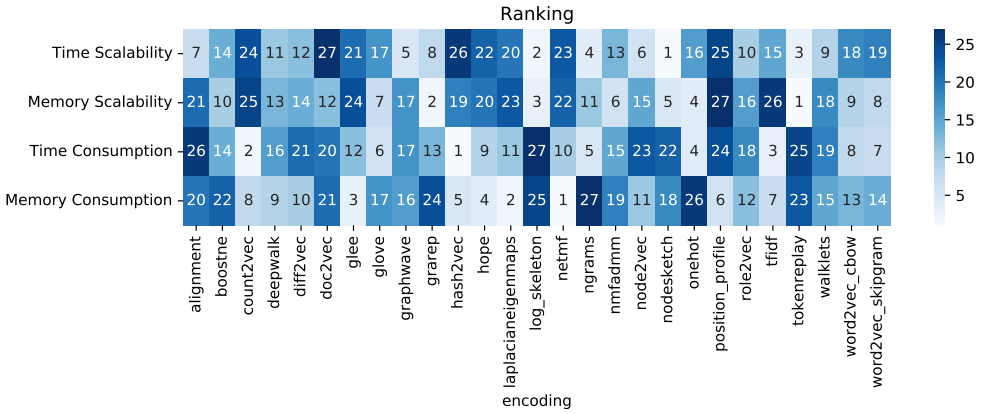


Figure 5.2: Ranking of Time Scalability, Memory Scalability, Time Consumption, and Memory Consumption. The better approaches are positioned in the first ranking position, colored by white. The most costly and least scalable are the last potions with dark colors.

Subsequently, we perform a high-level evaluation by grouping all the encoding methods by their respective families. In this approach, we have a drastic decrease in predictive performances in terms of the F1-score for the PM-based methods. This is due to the fact that part of this group performs very well, whereas the other part presents the poorest performances, as shown in Chapter 3. Moreover, the results illustrated in Figure 5.3 highlight a clear distinction between the predictive performances and scalability of different approaches for anomaly detection. It is evident that PM-based methods struggle with most types of

anomalies, performing well only in simpler cases. This limitation is due to their reliance on process models, which are not always suited for the complexity of the data inherent in anomaly detection tasks. In contrast, graph-based methods stand out as the most effective, with their ability to capture the structural nature of event logs providing a significant advantage. Their performance is followed by the baseline methods, which also perform well due to their better-suited data encoding for classification tasks. These results suggest that the representation of event logs as graph structures, rather than the textual representations that PM-based methods use, is more aligned with the underlying data patterns relevant for anomaly detection.

Regarding scalability, the results further underscore the limitations of PM-based methods. They are not only time-consuming but also do not offer a significant advantage in memory usage, rendering them less practical for large-scale applications. Text-based methods, although less effective in terms of predictive performance, shine in scalability, offering fast execution times. This could make them attractive for time-sensitive applications where rapid results are required and slight compromises in predictive quality are acceptable. Graph-based methods, while generally performing well in terms of execution time, exhibit some variability depending on the anomaly type. This suggests that while they offer strong overall performance, careful consideration is needed when deploying them in real-world scenarios with different anomaly patterns. Finally, baseline methods, although strong in predictive performance, are penalized by their memory consumption due to the high-dimensional nature of the data they handle. This trade-off makes them less scalable but highlights their suitability for tasks where memory resources are abundant and predictive performance is a key priority.

In conclusion, graph-based methods are the overall best performers in terms of F1-score, making them the most promising for high-quality anomaly detection, despite not being the best option in terms of sustainability. However, concerning sustainable purposes, text-based methods provide a compelling trade-off, balancing execution time and acceptable predictive performance, although they might be influenced by the datasets' characteristics. PM-based methods, while useful in very specific scenarios, require thorough evaluation to justify their use, given their high execution times and limited predictive quality in most cases.

5.3 Speeding Pipelines Up to Reduce Energy Consumption

Deep learning applications in process mining have gained tremendous attention in the last few years. It is well-known that these models are computationally costly, which reflects in non-environmental friendly. Therefore, in this Section,

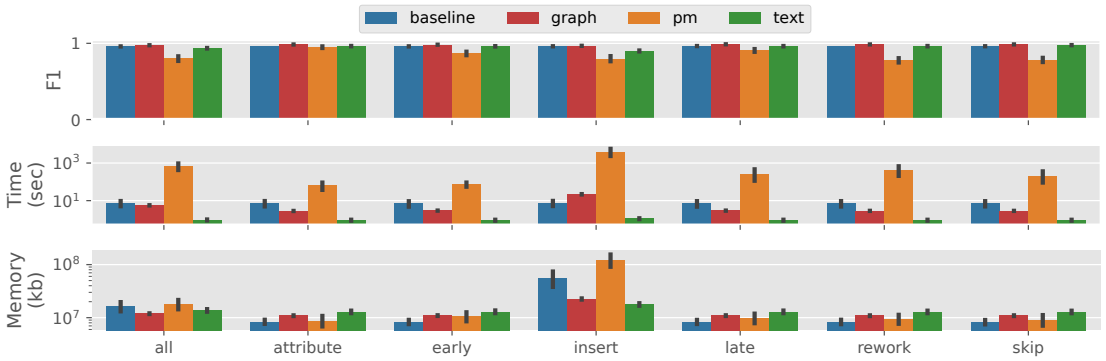


Figure 5.3: Overall performances for each anomaly type and grouped by algorithm families.

we motivate the usage, development, and investigation of cheaper strategies to achieve similar predictive performances. Our results show it is possible to balance predictive performance and computational resources in order to contribute to the field and to the environment. More specifically, in this contribution, we show that putting extra effort into the preprocessing phase improves predictive performance for the tabular algorithm in the remaining time prediction task. Once again, this analysis is an extension of Chapter 3 (more specifically, section 3.3). We only recall that each pipeline is composed of an optional preprocessing step and a shallow/deep learning algorithm (consisting of both training and inference time). No extra description regarding the experimental setup is needed.

5.3.1 Experiments

Roughly, deep neural networks outperform shallow methods for remaining time prediction without requiring extensive preprocessing, as shown in Chapter 3. However, their higher predictive performance comes at a cost in execution time. While LSTMs yield better results without data preprocessing, they demand significantly more computational resources and training time [65]. Figure 5.4 highlights the difference in execution times, with LSTMs taking about one order of magnitude longer than RF and KNN, emphasizing their resource-intensive nature. SVM, due to the high dimensionality from feature extraction, incurs even longer execution times.

In order to systematically assess the trade-offs between execution time and predictive performance, we conducted a Nemenyi post-hoc test following a Friedman rank test. The results, presented in Figures 5.5a and 5.5b, indicate significant differences across the pipelines. Regarding RMSE, pipelines involving comprehensive preprocessing steps (*n2v+ALL* and *OH+ALL*) consistently outperform

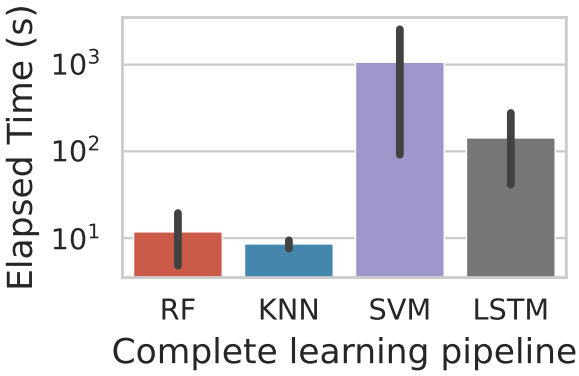
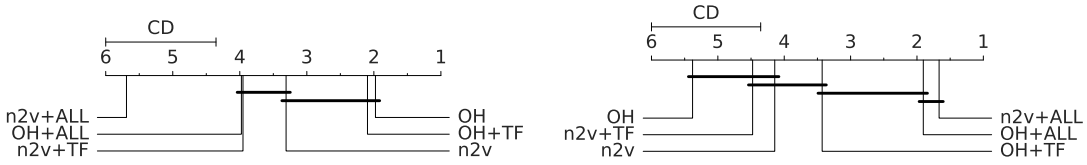


Figure 5.4: Execution time using *n2v+ALL* pipeline against the complete LSTM pipeline.



(a) Elapsed time

(b) RMSE

Figure 5.5: Post-hoc Nemenyi test of both preprocessing and training model. Groups that are not significantly different (p=0.05 and CD=1.62) are connected.

others, confirming that additional preprocessing can boost predictive accuracy. However, these improvements come at a cost, as both pipelines show significantly longer execution times, as depicted in Figure 5.5a. Although these pipelines demand more resources, they still perform faster than deep neural networks. Importantly, the test shows no significant difference between the *n2v+ALL* and *OH+ALL* pipelines in terms of both RMSE and execution time. This statistical evidence highlights the need for more research on less impactful methods in terms of sustainability since this analysis shows that deep learning is not always necessary.

5.4 Analysis of Execution Times for Trace Encoding Methods

So far, this Chapter have extended Chapter 3 by including analysis and experiments that address environmental issues. To conclude, we will now introduce a

new contribution to speed up the training of deep learning models. More specifically, we propose a new trace modeling or trace encoding to replace the traditional strategy of building datasets of prefixes. In our experiments, we show that both modeling approaches learn the same thing whereas our approach can be trained one order of magnitude faster. We also provide a thorough discussion.

5.4.1 Fundamentals

Outside the PM literature, sequence modeling involves creating models to process, predict, or generate sequences of data where the order matters [6]. These models handle sequential data like time series, text, or audio, making predictions based on observed data and considering dependencies between time steps. In process mining, the term 'encoding' is commonly used for sequence modeling at different levels, with various proposals for different problem scenarios in the literature. For example, Senderovich et al. [144] refer to encoding at case level to represent inter- and intra-case relationships, Tavares et al. [153] define it as the control-flow abstraction of ongoing traces, and Maneiro et al. [127] define it as either creating dataset of sequences or handling event attributes. In this Section, we recall definition 9 to propose a new approach to building a dataset of sequences.

As defined in definition 9, the *tra2seq* technique encodes a trace into prefixes that are shorter than the original trace. Thus, we can write $S \subseteq T$, where S is a subsequence within T. Let us define the two best methods according to the empirical evaluation from the benchmark [127]: the dataset of prefixes and the dataset of continuous traces. Consider a trace $t = \langle e_0, e_1, \ldots, e_n \rangle$. A prefix is a subtrace $st = \langle e_k, e_{k+1}, \ldots, e_m \rangle$, for $0 \leq k \leq m \leq n$. The prefix can either start from the first event e_0 or the w most recent events. The most popular approach is generating all the possible prefixes starting from the first event, resulting in n new prefixes given a trace of length n (each event becomes one target). On the other hand, the dataset of continuous traces flattens the event log and disregards the sense of cases. Similarly to the prefix length, in this method, we also set a hyperparameter to define the subsequence length. In both cases, an event log L might be augmented with a special event $e_{<EOT>}$ denoting the end of a trace. Moreover, the generated sequences (prefixes or continuous) that compose the dataset D are padded using the special event $e_{<PAD>}$ so every sample matches in length.

5.4.2 Speeding Up Deep Learning Training by Efficiently Managing Sequences

Building datasets composed of all the possible prefixes starting from the first event is the current state-of-the-art in the literature [127]. However, this is extremely

memory intensive, since for each trace of length n, n new sequences are generated. For instance, consider a trace $t = \langle a, b, c, d, e \rangle$. In order to build a dataset of prefixes, given an arbitrary prefix length of $l = 3$, the dataset of traces containing t will become a dataset of prefixes $D_p = \{\langle a, b, c \rangle, \langle b, c, d \rangle, \langle c, d, e \rangle\}$. When training a neural network, instead of having one batch containing one sequence, we now have a batch of three sequences. This significantly increases the computational cost. Furthermore, the prefix modeling is known as a many-to-one approach since each prefix will map to the next target only; that is, given an input $x = (a, b, c)$, its respective target will be $y = d$. Training under the many-to-many setup, at each time step of the trace, the model will predict the next one and backpropagate the errors at once. Finally, this limitation of the prefix modeling is seen in most PPM applications from the literature, and it is considered the current state-of-the-art [127].

To overcome this issue and decrease the environmental impact by reducing computational resource usage, we propose two key innovations: (i) a new *tra2seq* approach, called *trace modeling* for constructing a dataset of traces and to replace the prefix modeling. (ii) a many-to-many architecture to effectively learn from this dataset. The classic many-to-one predicts one target at a time for each prefix during an epoch, whereas the many-to-many learns to predict the same thing but all at once, resulting in more efficient training. Rather than flattening the entire event log continuously or chunking it into smaller prefixes, we utilize the entire trace. We argue that the current SOTA prefix approach has two drawbacks: it fails to capture complete temporal dependencies and leads to increased memory usage due to the generation of larger datasets. We believe the former issue particularly affects generalization capabilities, while the latter demands greater computational resources. In our experiments, we will validate these allegations.

To overcome the limitations of both continuous and prefix-based datasets, we propose the use of a dataset composed of complete traces to be trained under the many-to-many paradigm. In the trace modeling approach, the concept of case is kept, and they are used independently (a batch of size 4 will have 4 traces). For the event log L introduced earlier, the dataset of traces will consist of traces in their original form: $L_t = \langle \{a, b, c, EOS\}, \{d, e, EOS\} \rangle$. As a result, the samples remain independent and do not overlap. In summary, the trace modeling approach results in a dataset of n traces of length l, whereas the prefix modeling results in a dataset of $n * l$ prefixes of length p. Regarding the many-to-many architecture, during training, shorter sequences will be padded to match the length of the longest sequence. For validation, each sequence from the validation set will be fed starting from the first time step, and the model will autoregressively generate the remaining trace. During inference, the trained model can generate the rest of the trace based on either a single event or multiple events provided as input.

5.4.3 Experiments

We compare the prefix modeling (prefix length = 5) under the many-to-one paradigm versus the trace modeling under the many-to-many paradigm. We employ 5 real event logs of different sizes. From the smallest to the biggest one: BPI 2013 Problems, Sepsis, BPI 2012, BPI 2020 Permit to Travel, and BPI 2017. The goal of employing these event logs is to have a significant diversification in dataset sizes to validate our proposal. We employ the validation loss and accuracy as metrics to evaluate the predictive performances and the runtime (in seconds) and memory usage (in megabytes) to evaluate the resource usage needed to train a vanilla recurrent network for the next activity prediction problem. Notice that the same network under the same configuration is trained for both modeling approaches (prefix and trace). We consider the activity labels and the execution time extracted from time stamps as event attributes. The same hyperparameters are used for both *tra2seq* approaches to ensure a fair evaluation. We do not focus on tuning since the goal is to evaluate the capabilities of each approach to learn from the data by using less or more computational resources.

Figure 5.6 illustrates all the findings. We can see in Figure(a) and Figure(b) that in terms of predictive performances, both *tra2seq* approaches perform alike. This invalidates our theory that prefix modeling harms the generalization capabilities. By leveraging only the local information, i.e., the most recent events, or the whole trace information is indifferent, it is possible to effectively predict the next events using both approaches.

Since both *tra2seq* approaches perform similarly, Figure(c) and Figure(d) are remarkable since we can see for some datasets a speedup higher than one order of magnitude. The main cause for this outstanding performance is due to the fact the many-to-many paradigm already predicts the next event for every possible prefix. Building a dataset of prefixes is not efficient since each trace will be encoded as multiple samples, instead of treating a trace as the whole sample. In our approach, our dataset is composed of n traces rather than $n * l$ prefixes.

5.5 Final Remarks

In this Chapter, we demonstrated how to address environmental concerns related to computational resource usage when developing AI-based solutions for process mining by answering our RQ3: "*How do the complex temporal, relational, and semantic patterns in process data impact an organization's ability to leverage Process Mining solutions for optimizing processes and addressing environmental challenges, such as energy consumption?*". We emphasized that reducing resource consumption directly lowers environmental impact. To support this, we

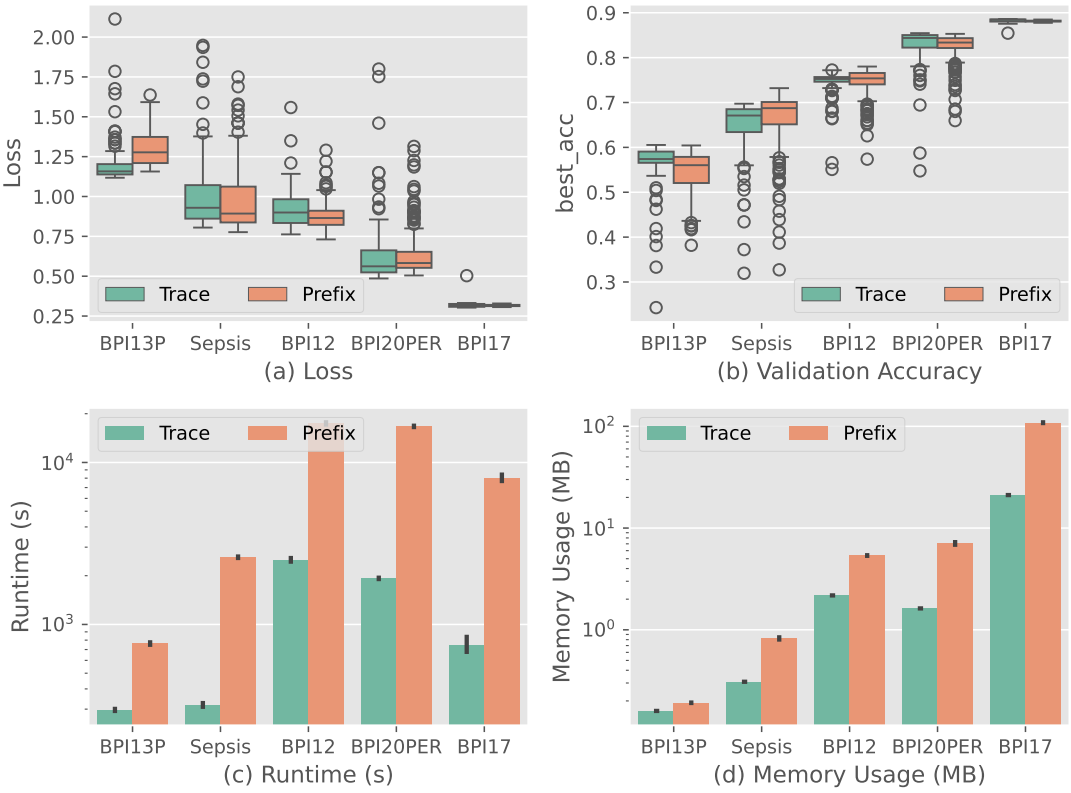


Figure 5.6: Figure showing both modeling approaches (prefix and continuous) converge but the runtime for continuous modeling is much faster.

proposed incorporating resource usage metrics alongside traditional predictive performance measures commonly used in PPM tasks. Additionally, we encouraged the exploration and development of less resource-intensive methods as an alternative to GPU-heavy approaches like deep learning.

Building on the encoding techniques from chapter 3, we extended it to include scalability analysis, which allowed us to assess the trade-offs between predictive performance and resource usage. Subsequently, our findings show that it is possible to create robust, sustainable AI solutions in process mining without relying solely on deep learning. In fact, deep learning may not always be the best choice, and simple techniques in data preprocessing can often achieve a better balance between performance and resource consumption. Moreover, we introduced a new approach to training deep learning models that significantly reduces training time. Traditional prefix modeling in process mining is memory-intensive, leading to more GPU operations and longer training phases. By switching to our proposed trace modeling, we were able to reduce training time by one order of

magnitude, offering a more efficient way to build datasets for PPM applications based on deep learning.

Focusing again on process representation, in the next Chapter, we show how declarative process models can improve training data and enable more flexible process simulations using deep learning. These models encode semantic rules, allowing users to simulate new behaviors based on custom-defined guidelines.

Chapter 6

Predictive Process Monitoring and Future Directions

PPM plays an important role in PM research by helping analysts forecast the future states of ongoing processes. Throughout this thesis, we have advanced the state-of-the-art through peer-reviewed publications. In this Chapter, we aim to discuss flaws in the field and directions for future research based on our findings. We claim that current work in this area often suffers from problems related to reproducibility and fairness. For instance, it is common that open-source repositories become outdated, as researchers who share their code might not continuously update it in line with evolving tools and programming languages. This makes it difficult for new practitioners to reproduce the same results with the exact same setup, or to know how changes in data handling affect results.

Another issue is fairness. Different studies may handle data in different ways. For example, one group might remove rare events from the dataset, while another keeps them. These differences can affect how final results are interpreted. If models are not all trained and tested on exactly the same data, we cannot be sure if a claimed improvement is due to a better model or simply due to different preprocessing steps. Using a fixed and standardized test set across all comparisons, as seen in popular datasets like MNIST [40], can help ensure that differences in performance come from the models themselves, not from data choices.

To address these problems, this Chapter deeply discusses such issues and presents an extension to the well-known Scikit-learn library tailored to PPM purposes. Scikit-learn is widely used in machine learning for its clear interface and ability to create complex pipelines. By adapting it to the needs of PPM, we provide a uniform framework for data preparation and model evaluation. This should allow researchers to build and compare models in a consistent, repeatable, and fair way. We aim to improve standards, encourage best practices, and help align PPM studies with the broader machine learning community's focus on clear

benchmarks and reproducible results.

6.1 Challenges in the Literature

It is difficult to evaluate and compare machine learning pipelines when multiple components change at the same time or when data splitting is inconsistent. Consider a dataset $\mathcal{D}$, a set of preprocessing functions $\mathcal{P}$, a set of machine learning models $\mathcal{M}$, and an evaluation metric $\mathcal{E}$. A pipeline involves selecting a preprocessing function $p \in \mathcal{P}$, applying it to the dataset $\mathcal{D}$ to create a transformed dataset $p(\mathcal{D}) = d$, and then training a model $m \in \mathcal{M}$ on d. The pipeline's performance is evaluated as $\mathcal{E}(pipe) = e$, where $pipe = m(p(\mathcal{D}))$.

When comparing pipelines such as $pipe_1 = m_1(p_1(\mathcal{D}))$ and $pipe_2 = m_2(p_2(\mathcal{D}))$, differences in performance might result from changes in preprocessing, changes in the model, or both, making it difficult to track where the improvement actually lies. For instance, claiming that $\mathcal{E}(pipe_1)$ is better than $\mathcal{E}(pipe_2)$ does not allow us to fully understand where the actual improvment comes from, since training m_2 on d_1 could result in a third, completely different performance. Furthermore, splitting data after or before preprocessing can lead to different test sets, which can also affect the reliability of results. Thus, the sequence of operations regarding the split matters. Splitting the data before preprocessing ensures all pipelines use the same test set, which improves reliability. For example, take into consideration the benchmarked approaches detailed in the supplementary material from [127], where deep learning applications were evaluated. From a machine learning point of view, the differences in the preprocessing steps are not fully fair since the splits are arbitrary. However, the benchmark uses cross-validation for those methods that ensure a more balanced comparison despite the lack of a fixed test set.

On the other hand, comparing deep learning versus machine learning models often involves inherent nuances, particularly when dealing with sequential data. In deep learning applications for PPM, RNNs are commonly employed due to their natural ability to process sequential data. These models handle batches of (sub)sequences, often padded to ensure uniformity. In contrast, machine learning models require the creation of datasets comprising encoded (sub)traces, as previously discussed. Unlike RNNs, which inherently respect the sequential nature of data, machine learning models operate under the assumption that data is independently and identically distributed. Consequently, the sequential dependencies are captured indirectly through the encoding techniques that transform sequences into feature vectors. These differences imply that datasets used for training deep learning and machine learning models are inherently non-identical. Nevertheless, fairness in comparisons mandates the use of consistent feature engineering and data split techniques before training both types of models.

Take as an example the contribution in [57], which compares LSTMs with CatBoost models. In the study, LSTMs are trained on numerical and boolean features in their raw form, with no additional transformations, while categorical features are one-hot encoded. In contrast, the CatBoost models utilize an expanded set of engineered features derived from the raw numerical features, with categorical features automatically handled by the model. The primary contribution of this paper lies in its focus on explainability achieved through the newly engineered features. However, predictive performance validation remains crucial, as the value of the explanations depends on the model's ability to achieve adequate predictive accuracy. While the authors' contribution is significant, it is well-known that the LSTM's performance can benefit from similar feature engineering [155, 71], potentially enabling practitioners to capitalize on both improved predictive accuracy and enhanced explainability. Similar disparities in preprocessing and feature engineering can be observed in related studies [127], highlighting a recurring challenge in achieving fair comparisons.

Data splitting also raises several important concerns from a machine learning perspective. First, the lack of standardized test sets makes it hard to compare results across different experiments. Recent work [178] has tried to address this by recommending standardized data splits in the context of PPM. The authors also attempted to reduce bias at the preprocessing stage. Ideally, however, machine learning practice would suggest setting aside a raw, untouched test set and applying any bias-reduction steps only to the training data. Nevertheless, this approach is difficult because the processes producing the events often overlap in time, characterizing a particular intrinsic difficulty in handling this nature of data. As a result, time-based splits still leave some traces of temporal overlap between training and testing subsets, as observed in their work. Second, using random cross-validation for temporal event data, as done in recent benchmarks [127], may be inappropriate because it ignores the natural time order of events, does not take into consideration the case perspective since prefixes become independent, and also lacks a standardized test set. Although time-based splits have been explored in prior studies, there is still a need for better-defined and more careful benchmarks that respect the temporal structure of the data.

All the technical details discussed so far represent design decisions, guided by established research evidence and insights derived from the data. We want to shed light on the machine learning perspective, where it is equally important to consider how these choices can influence fairness and comparability in evaluations. Recent work [80] offers a framework to better understand how different design options affect the performance of PPM tasks. Their approach considers both model-level and PPM-specific hyperparameters, including choices in sequence and event encoding as well as the handling of time-related features. Their findings show that these decisions can lead to noticeably different results across various

tasks, reinforcing the issues discussed in this Chapter. Finally, this evidence supports the idea that design decisions should be selected with careful attention to the specific task and characteristics of the dataset, ensuring more reliable and fair evaluations in practice.

6.2 Centralized Open-source Tool for Robust Research

In this Section, our goal is to address the aforementioned issues by proposing an extension to the widely recognized Scikit-learn library [115] tailored specifically to the demands of process mining. Scikit-learn, renowned for its versatility and user-friendly Application Programming Interface (API), has emerged as a foundation in the ML community, empowering researchers to construct intricate pipelines, ensuring reproducibility, and facilitating comprehensive benchmarking. We aim to bridge the gap between the demands of process mining and the capabilities of a mature and well-established ML library. This way, we expect that the accessibility and efficiency of ML experiments within the PM domain will be enhanced by mitigating the flaws we have delineated so far. We will dive into the specifics of our proposed extension, describe its design and functionalities, and present a few use cases for predicting the remaining time and the next activity of processes.

6.2.1 Motivation

Within the domain of PM, where the unique characteristics of the data demand specialized attention, there exists a need for a dedicated ML library to facilitate the essential aspects of reproducibility, fair evaluations, and benchmarking. Due to the particular characteristics of process data (cases, traces, events, attributes, attribute sharing, etc.), the research field requires specific preprocessing steps for ML purposes which usually consist of extracting temporal feature representations, mapping resource usage, and measuring process costs [46].

Existing tools, including generic PM software (e.g., ProM[1][166], Apromore[2][134], and pm4py[3][13]), process analytics tools (e.g., DyLo[4][181]), and comprehensive surveys offering open-source access to their experimental setups [158, 127], while valuable resources, do not fulfill the role of dedicated tools for practical ML application and extensibility. However, Nirdizati[5] [133] is a tool that provides a

[1] `https://promtools.org/`
[2] `https://apromore.com/`
[3] `https://pm4py.fit.fraunhofer.de/static/assets/api/2.7.5.1/index.html`
[4] `https://github.com/BrechtWts/DyLoPro`
[5] `http://nirdizati.org/`

user interface and automates the learning pipelines. It is the most similar tool to our purpose, although it does not work at scale since each experiment has to be executed manually, whereas our approach aims to automate benchmarks and speed up extensive experimental evaluations.

The benchmarks commonly employed are static in nature and present challenges when it comes to extensibility and adaptation. The well-known *pm4py* may be considered the most popular tool for managing event logs. Although the library has a module dedicated to ML, it is limited to a few simple feature engineering/extraction solutions. Thus, in the end, the design of ML pipelines consisting of tuning for both preprocessing and training steps is not supported, i.e., is left to the user. While existing benchmarks [158, 127] utilize the core methods of *Scikit-learn* and *pm4py*, the incorporation of essential elements for implementing PPM pipelines often require significant overhead. Such overhead is illustrated by the reliance on external libraries beyond Scikit-learn, introducing complexities in terms of maintenance, reproducibility, and adaptability as these external dependencies may quickly become outdated.

6.2.2 Design and Implementation

This Section presents technical details. First, we present an overview of how to extend Scikit-learn. Second, we discuss how PM-related features/steps might be included under the Scikit-learn API.

Scikit-learn Overview. In summary, the Scikit-learn API involves the following fundamental steps: firstly, the *instantiation* of an estimator or transformer; secondly, the *fitting* of the object instance to acquire knowledge from the input data set; and finally, the *prediction* process if the object serves as a predictor, such as an RF, or the *transformation* process if the object is intended to modify the data in a certain manner. From the official documentation, we summarize the main characteristics of the library below.

- **Instantiation**. The `__init__()` should contain arguments that determine the estimator's behavior. More specifically, only tunable hyperparameters should be included here. This is important in more complex applications, for instance when performing grid search, to differentiate the hyperparameters from attributes. No logic nor validation should be implemented here.

- **Fitting**. The `fit()` method takes the training data as an argument, which can be just one array if it is an unsupervised problem or a data transformation; or two arrays if it is a supervised learning scenario. Additionally, it accepts optional data-dependent parameters.

- **Predicting or Transforming**. If `fit()` succeeds, it will estimate parameters from the data (e.g., the mean and standard deviation for z-score normalization). Finally, the methods `predict()` or `transform()` can be called, depending on the nature of the estimator.

Extending Scikit-learn for Process Mining. Following the specified criteria, we formulate a Scikit-learn extension for PM by utilizing its core methods for standardizing ML experiments in the PM community. In this initial work, our focus centers on the event attributes, including the case identifier, timestamp, activity, and categorical resources.

For activity representation, we employ one-hot encoding, aligning with the prevalent practice in the PM community, as indicated by prior research [158], despite recent findings suggesting superior performance with alternative encoding methods [153]. Concerning timestamp-related feature extraction, our library accounts for both the case identifier and the timestamp to ensure proper attribution of events to their respective process instances, especially when calculating time-based metrics such as the time difference between consecutive events (e_{i-1} and e_i). Additionally, while categorical resource attributes (e.g. actors playing events) can be one-hot encoded, our library also offers an alternative organizational mining algorithm for role discovery [149]. This algorithm identifies common roles among actors, facilitating resource grouping and reducing category dimensionality.

When instantiating our estimators, users have the flexibility to specify parameters such as the time-related features to extract from timestamps and the sensitivity threshold for the role discovery algorithm. During the fit method invocation, users must provide the column names corresponding to each required event attribute for a given estimator. By meticulously designing each estimator for event feature extraction, we seamlessly integrate them into the well-established Scikit-learn pipelines and model selection algorithms. In the subsequent Section, we illustrate a practical use case by constructing a concise pipeline using our proposed library.

6.3 SkPM Usage

Our work is under continuous development and we are already able to perform a complete pipeline involving preprocessing and prediction. In this Section, we present an overview and introduce code snippets of a pipeline for remaining time prediction to illustrate the initial issue of evaluating ML models under different preprocessing setups.

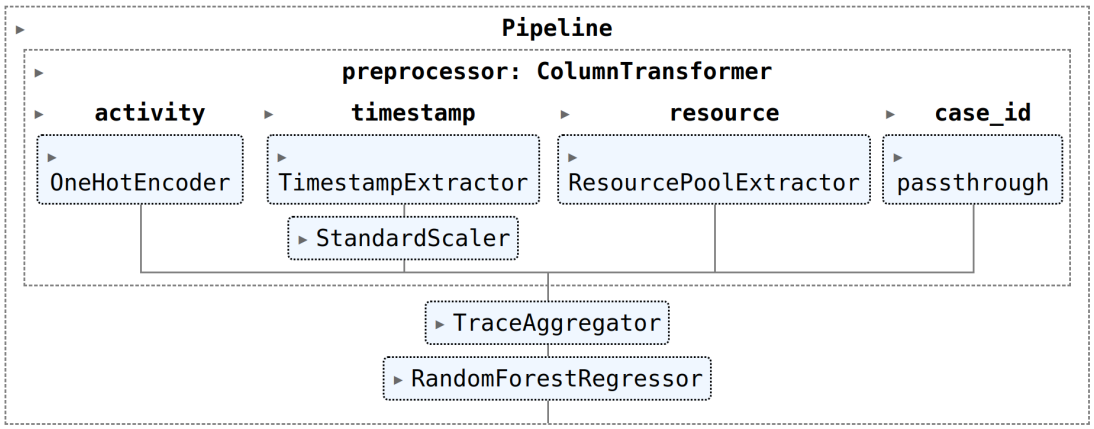


Figure 6.1: An example of our SkPM pipeline implemented extending the traditional Scikit-learn library.

6.3.1 A High-level Overview

To showcase the practicality of SkPM, we have implemented two feature extractors, one for timestamps and another for categorical resources, and tested them on public event logs (see our repository). The class `TimestampExtractor` has only one argument `features` that describes all the features that should be extracted from the timestamp. These time-related features were collected from the literature and they currently include `execution_time`, `accumulated_time`, `within_day`, and `remaining_time`. The latter is commonly employed as a target to be predicted. Regarding the categorical resources, we have implemented the resource role discovery as mentioned in the previous Section, which in a nutshell aggregates common resources into groups (roles in the organization), hence reducing the set of available categorical values. This algorithm has a hyperparameter that is used as a relationship threshold, which can be seen as a threshold to decide if resources belong or not to the same group.

That said, we can instantiate a Scikit-learn pipeline as illustrated in Figure6.1. For activity encoding, we simply use the traditional one-hot encoding. The timestamp can have all the implemented features extracted in order to increase the knowledge space, and normalized subsequently. The resource has its set of unique and individual labels reduced to roles (a.k.a. pool). Lastly, the case identifier needs to be passed through, i.e., it is fed to the ColumnTransformer and should not be dropped (default operation in Scikit-learn) in order to be used in the next step. The TraceAggregator is a trace encoding technique described in [158] which consists of aggregating the previous events of an ongoing case in order to output a single array. This aggregation considers averaging or summing all the knowledge up. Finally, an RF or any other learning algorithm can be

employed. In our example, we illustrate the *RandomForestRegressor* since we are predicting the remaining time of ongoing cases. The reader is able to run the pipeline by following the instructions in our repository.

The proposed extension builds upon the original Scikit-learn interfaces, allowing for the seamless development of PPM applications. By retaining compatibility with Scikit-learn's established structure, researchers can implement PPM methods with greater ease and efficiency. Additionally, the ability to visualize the constructed pipelines enhances interpretability, facilitates meaningful discussions, and streamlines the experimental workflow.

6.3.2 Remaining Time Prediction Example

In this Section, we provide code snippets to demonstrate the simplicity and usability of SkPM. To keep the presentation focused and concise, we include only the essential code, omitting details such as imports, function arguments, and variable definitions.

We begin by showing how to define two preprocessing pipelines in Listing 6.1. The first example is a basic pipeline that processes activity labels using one-hot encoding. The second example introduces a more advanced pipeline that incorporates additional features, such as time-related attributes and resource pool extraction. These functionalities are accessed through intuitive APIs that illustrate well-known methods in the literature. The differences between the pipelines are easily managed and clearly readable using the 'ColumnTransformer' method, which highlights the flexibility and clarity of SkPM.

```
1
2  # Simple preprocessing pipeline
3  data_prep_simple = Pipeline([
4      ("preprocessing", ColumnTransformer(
5          transformers=[
6              ("activity_encode", OneHotEncoder()),
7      ])),
8      ("encode_agg", Aggregation()),
9      ("scaling", StandardScaler()),
10 ])
11
12 # Advanced preprocessing pipeline
13 data_prep_advanced = Pipeline([
14     ("preprocessing", ColumnTransformer(
15         transformers=[
16             ("timestamp_features", TimestampExtractor()),
17             ("activity_encode", OneHotEncoder()),
```

```
            ("resource_pool", ResourcePoolExtractor()),
        ])),
    ("encode_agg", Aggregation()),
    ("scaling", StandardScaler()),
])
```

Listing 6.1: Code snippet illustrating how to define complex and simple pipelines using SkPM.

Next, we demonstrate how two distinct machine learning models, Gradient Boosting and RF, can be trained using both preprocessing pipelines. The implementation is clear and intuitive, showcasing the ease of integrating these models into the SkPM framework. By providing these functionalities, we aim to speed up advancements in the research field by enabling faster, more systematic, and reproducible experimentation. This not only starts to address the challenges discussed earlier in this Chapter, such as inefficiency and lack of standardization but also enhances the robustness and reliability of process mining research.

```
# Gradient Boosting pipelines
gb_pipe_advanced = Pipeline([
    ("preprocessing", data_prep_advanced),
    ("regressor", GradientBoostingRegressor())
])

gb_pipe_simple = Pipeline([
    ("preprocessing", data_prep_simple),
    ("regressor", GradientBoostingRegressor())
])

# Random Forest pipelines
rf_pipe_advanced = Pipeline([
    ("preprocessing", data_prep_advanced),
    ("regressor", RandomForestRegressor(n_estimators=10, ))
])

rf_pipe_simple = Pipeline([
    ("preprocessing", data_prep_simple),
    ("regressor", RandomForestRegressor(n_estimators=10, ))
])
```

Listing 6.2: Code snippet illustrating how to define training pipelines using different preprocessing pipelines and different machine learning models.

6.4 Conclusion and Discussion

In this Chapter, we discussed the current state of PPM in the literature and identified key flaws found during this research. These flaws are often from technical details that are overlooked. This is reasonable since machine learning is a mere support tool in process mining research. To address these challenges, we introduced a new tool designed to support and guide researchers: an extension of the widely used Scikit-learn library. By leveraging Scikit-learn's existing interfaces, we implemented well-established PPM features in an accessible, intuitive manner. This centralized and open implementation allows researchers and practitioners to efficiently build, test, and refine their solutions, fostering greater reproducibility and reducing technical barriers.

In the future, we plan to use this library as the foundation for a continually updated public benchmark, ensuring long-term relevance and accessibility for the process mining community. Finally, we encourage researchers and practitioners to actively contribute to the development of SkPM by visiting our public repository at `https://github.com/raseidi/skpm`. The collaboration will ensure the tool evolves to meet the community's needs and drives further progress in process mining research.

Chapter 7

Use Case at Avio Aero

In this chapter, we present a novel solution for Avio Aero [1] that builds upon the key findings and developments from our previous research. The intersection between process mining and deep learning offers promising opportunities in modern industries and business applications, particularly for companies where operational excellence and precision are crucial and costly. Avio Aero, a partial sponsor of this PhD program, seeks to innovate in this domain. The company distinguishes itself through advanced manufacturing in the aerospace domain, including technologies that improve operational efficiency while reducing environmental impact.

Our use cases, though, address a different sector within Avio Aero. We leverage the knowledge gathered during the development of this thesis to propose a new solution that enhances the internal processes of the organizational issue-tracking system. As their information systems generate increasingly complex and large amounts of data, traditional process optimization methods might not be enough for decision-making. The scale and complexity of process data extracted from Avio Aero's issue-tracking system often exceed the capabilities of conventional process mining approaches. Thus, we employ deep learning technologies to complement these limitations by offering sophisticated pattern recognition and predictive capabilities based on textual data recorded in such event logs.

For instance, the integration of process mining with LLMs creates new possibilities for process improvement. LLMs can interpret process descriptions and user interactions to classify processes and identify subtle anomalies that may indicate quality issues or inefficiencies, enabling interventions, automation, and changes. This integration is particularly relevant for Avio Aero. The company's business model generates rich process data suitable for both process mining and deep learning applications. Their extensive log of users' incidents provides an ideal environment for implementing and testing these advanced technologies.

[1] `https://www.avioaero.com/en/`

Thus, our research explores both theoretical foundations and practical applications of combining process mining with deep learning in their settings. We demonstrate how this integration can enhance operational efficiency, reduce costs, and provide meaningful insights for stakeholders through intelligent, data-driven solutions.

The chapter is structured as follows: First, we describe Avio Aero's operational framework and daily business tools. Next, we explain how these frameworks enhance productivity in their Information Technology (IT) environment. Finally, we present three case studies: the first employs traditional process mining techniques, the second utilizes LLMs, and the third demonstrates the promising combination of both technologies.

7.1 Business Frameworks

Avio Aero employs the Information Technology Infrastructure Library (ITIL) [125, 100] framework. ITIL has emerged as a cornerstone in the area of IT Service Management (ITSM) and in this section we introduce its origins, evolution, and significance, providing a comprehensive background for understanding its role in aligning IT services with business objectives.

7.1.1 Information Technology Infrastructure Library

The origin of ITIL can be traced back to the 1980s when the British government's Central Computer and Telecommunications Agency recognized the need for a standardized approach to ITSM [20]. In response, the CCTA meticulously documented a plethora of best practices, culminating in the creation of the ITIL. Originally, ITIL was conceived as a series of books, each addressing specific aspects of ITSM. However, as the digital landscape evolved, so did the nomenclature; today, ITIL stands as a standalone term, no longer an acronym but a brand in its own right.

Evolution and Contemporary Relevance of the ITIL Framework in ITSM. The ITIL has undergone a transformative journey since its inception in the late 20th century [73]. Initially developed as a UK government initiative during the 1980s, ITIL aimed to standardize and introduce best practices within the rapidly expanding domain of IT management. The framework's early manifestation comprised an extensive series of over 30 volumes, reflecting the comprehensive nature of ITSM at the time. As the turn of the millennium approached, the ITIL framework underwent significant revisions. The second iteration, introduced around 2000, strategically streamlined these volumes into cohesive sets, each delineating specific aspects of IT management, services, and applications. Notably,

this period also marked the adoption of ITIL by global technology conglomerate Microsoft, integrating it as a foundational element of its Microsoft Operations Framework.

The subsequent versions of ITIL, particularly version 3 released in 2007, emphasized process improvement and introduced a lifecycle-centric approach. This version aimed to bridge the ever-narrowing gap between business and IT objectives, ensuring that IT services were not only efficient but also aligned with broader organizational goals. The direction of ITIL transitioned to AXELOS in 2013, marking a new era of administration and periodic enhancements to the framework. Under AXELOS's guidance, ITIL has been adapted towards its fourth iteration, addressing the challenges and opportunities presented by contemporary technological paradigms such as digital transformation, artificial intelligence, and cloud computing. Furthermore, ITIL 4 has embraced modern methodologies, integrating principles from Agile and DevOps, reflecting the evolving nature of IT service delivery.

In summation, the evolution of ITIL, from its foundational principles in the 1980s to its current holistic approach, encapsulates the dynamic trajectory of ITSM. The framework's continuous refinement, coupled with its adaptability to integrate modern methodologies and technological advancements, underscores its enduring relevance and pivotal role in shaping ITSM best practices. Finally, we provide a summary of the evolution of ITIL versions in Table 7.1

Key pillars of the ITIL framework. The Configuration Management Database is a pivotal component that serves as the authoritative repository for all assets essential for IT service delivery. The CMDB meticulously tracks and records the location, modifications, attributes, and interrelationships of these assets, ensuring a holistic view of the IT infrastructure.

ITIL's architecture is further assisted by its five key stages, each encompassing specific processes:

- **Service Strategy**: This foundational stage underscores the importance of aligning IT services with overarching business objectives. It encompasses processes such as strategy management for IT services, service portfolio management, and financial management for IT services, among others.

- **Service Design**: This stage is dedicated to the meticulous design of services and processes, ensuring they are robust, scalable, and aligned with business needs.

- **Service Transition**: A critical phase, it focuses on the seamless transition of new or modified services into the operational environment.

Version	Release year	Summary
v1	1980	It consisted of 30 volumes of best IT practices. It became a global standard in the early 1990s, transforming IT management in the UK, Europe, and other regions.
v2	2000	It reduced the previous 30-volume catalog to nine categories. It became the most accessible ITSM tool globally. This version was also adopted by Microsoft, leading to the creation of the Microsoft Operations Framework Foundation.
v3	2007	It emphasized business integration around the service lifecycle structure. It introduced new processes for improvement, change management, and service delivery. AXELOS released a revised version in 2011 to address inconsistencies and errors.
v4	2019	It focuses on collaborative IT environments. It aligns with modern methodologies like Agile, DevOps, and Lean, making it more suitable for contemporary digital environments.

Table 7.1: Versions of ITIL.

- **Service Operation**: This operational stage ensures that the delivered services function optimally and reliably, meeting the stipulated service level agreements.
- **Continual Service Improvement**: As the name suggests, this stage is dedicated to the iterative improvement of IT services, ensuring they evolve in tandem with changing business requirements.

A distinction that ITIL meticulously draws is between incident and problem management. While incident management is reactive, addressing individual disruptions like hardware failures, problem management is proactive, delving into the root causes and formulating strategies to preclude recurrence. In this thesis, we cover only incident management and will further detail it in the following sections.

ITIL is not just a library but a holistic framework employed in ITSM [146]. Modern ITSM tools, underpinned by ITIL principles, automate the service management process, offering invaluable insights into service levels and facilitating resource optimization.

Certification and Professional Development. AXELOS, recognizing the need for standardized training, offers a gamut of ITIL certification programs [2]. These range from foundational courses to advanced masterclasses, catering to professionals at various stages of their careers. Such certifications not only enhance organizational capabilities but also bolster individual career trajectories. The certification programs are divided into levels, as follows:

1. **Foundation**: introduces basic concepts, elements, and terminology of the ITIL framework.
2. **Practitioner**: focuses on the Continual Service Improvement approach and covers organizational change management, communication, and measurement and metrics.
3. **Intermediate**: it is defined in two sub-levels: Service Lifecycle: concentrates on core itil phases such as strategy, design, transition, operation, and continual service improvement; and Service Capability: focuses on specific aspects of ITSM, including operational support, analysis, planning, protection, optimization, release, control, validation, and service offerings.
4. **Expert**: demonstrates a comprehensive understanding of the entire ITIL scheme and requires completion of the ITIL managing across the lifecycle capstone course.

[2] `https://www.axelos.com/certifications/itil-service-management/itil-4-foundation/`

5. **Master**: requires a minimum of five years of leadership in ITSM. Demonstrates the ability to apply ITIL principles, methods, and techniques in real-world scenarios.

The Road Ahead for ITIL. ITIL has undergone significant transformations across its versions, each iteration reflecting the changing dynamics and requirements of the IT industry. From its comprehensive 30-volume structure in ITIL v1 to the more streamlined and business-integrated approach in the subsequent versions, ITIL's journey encapsulates the continuous efforts to align IT services with technological advancements and organizational objectives.

In the ever-evolving digital landscape, ITIL's adaptability will be its greatest asset. As businesses undergo rapid transformations, ITIL processes, such as the change approval board [3], will need to be agile, possibly transitioning to more dynamic, policy-driven approval mechanisms. This adaptability is evident in the framework's recent versions, which emphasize collaboration and co-creation of value, integrating modern methodologies like Agile and DevOps. Such evolutions highlight ITIL's commitment to staying relevant and addressing contemporary challenges in the IT domain.

In conclusion, ITIL, with its rich legacy and forward-looking approach, remains an indispensable framework for ITSM. Its enduring relevance is a testament to its foundational principles and its ability to evolve in response to the industry's needs. Ensuring that IT services are not just efficient but also in harmony with business imperatives, ITIL continues to provide organizations with a roadmap to navigate the complexities of IT service delivery in a rapidly changing digital environment.

7.1.2 Application Management Services

Application Management Services (AMS) is another framework employed by Avio Aero and it offers various types of support designed to assist business applications in specific ways. These include service requests, incident management, routine maintenance, and enhancement hours. For instance, service requests provide users with guidance or configuration assistance, whereas incidents address unexpected application issues.

AMS can benefit businesses of all sizes by giving them access to specialized expertise in a cost-effective way, offering continuous, 24/7 support that helps maintain optimal system performance. As businesses in companies like Avio Aero increasingly rely on technology, AMS becomes crucial in helping them sustain efficient, reliable operations and remain competitive in a fast-paced market.

[3] `https://en.wikipedia.org/wiki/Change-advisory_board`

This framework's services follow a structured, well-defined methodology, using standardized processes, policies, procedures, and templates. Each AMS process is tailored to the unique needs of each engagement, with specific tools like checklists, forms, and guidelines that support quality-focused activities across various stages, including planning, governance, validation, auditing, and progress reviews.

AMS support is organized into distinct levels, or tiers, to efficiently address issues based on complexity and expertise requirements:

1. **Level 1** (Operational Support): This is the initial support tier, managing straightforward issues and routine inquiries. Level 1 addresses basic troubleshooting, issue identification, and initial data collection. If issues are more complex, they are escalated to Level 2.

2. **Level 2** (Advanced Support): This level deals with more complex problems that Level 1 could not resolve. Support here includes advanced troubleshooting, application configuration, and minor customization. Issues beyond this scope are escalated to Level 3.

3. **Level 3** (Expert Support): At this top level, AMS handles the most complex issues requiring specialized knowledge. Tasks here may include advanced configurations, custom code adjustments, and in-depth problem-solving.

These tiers and stages establish a systematic approach to application management, ensuring issues are addressed with the appropriate expertise level, ultimately leading to effective, efficient resolution.

7.1.3 ServiceNow

ServiceNow [4] is a comprehensive platform designed to streamline digital workflows and enhance the efficiency of business operations across an organization. Built with flexibility and scalability in mind, ServiceNow integrates seamlessly with various enterprise systems, supporting a broad range of applications from ITSM to Customer Service, HR, and Security Operations. This platform provides a centralized hub for managing tasks, tracking incidents, and facilitating cross-departmental collaboration, creating a unified digital workspace that enhances productivity and accountability. At Avio Aero, ServiceNow is used as the official organizational issue-tracking system.

[4] `https://www.servicenow.com/`

The platform's extensibility is one of its defining features, allowing businesses to customize workflows, integrate third-party applications, and develop specialized tools tailored to their needs. As organizations scale, ServiceNow's cloud-based architecture enables them to adapt and expand their digital infrastructure efficiently. This adaptability, coupled with a focus on user-friendly design, makes ServiceNow an invaluable asset to drive digital transformation, optimize operations, and foster continuous improvement in a dynamic and competitive business environment.

7.1.4 Business Frameworks in Big Industries

Besides Avio Aero, other key reference large organizations, including Microsoft, IBM, Facebook, Amazon, Google, and Apple, utilize diverse IT governance frameworks and practices to effectively manage their IT services. A prominent framework commonly adopted by these companies is the ITIL, which provides industry-recognized best practices for ITSM, as discussed in the previous chapter. Here, we focus specifically on how leading organizations integrate ITIL into their governance structures, a central theme of this thesis.

Microsoft exemplifies ITIL integration through its managed desktop [5], a comprehensive ITSM model that merges best practices with cutting-edge technology. The MMD framework aligns closely with ITIL principles, particularly across service design, transition, and operations phases. In the design phase, MMD commits to transparent response times, well-defined service descriptions, and a robust security framework. During the service transition phase, it ensures meticulous management of updates and precise delineation of responsibilities, reflecting ITIL's approach to change management. The service operation phase is marked by MMD's extensive event and incident management protocols, supported by a structured request fulfillment process. Thus, Microsoft's MMD is more than a technological solution; it is a comprehensive system that unites advanced technology with recognized IT management practices to support organizational needs with precision.

Further highlighting this alignment, Microsoft's Managed Desktop also integrates Microsoft 365 Enterprise components to offer a cloud-based, end-to-end service plan. This solution encompasses device provisioning, ITSM operations, and round-the-clock security monitoring, demonstrating a proactive approach to updates and change management. Reflecting ITIL's focus on continual improvement, Microsoft's support for transitions, such as the upcoming end-of-life phase for MMD, reinforces its commitment to providing technologically advanced, ITIL-aligned solutions that meet global best practices.

[5] `https://learn.microsoft.com/en-us/managed-desktop/overview/mmd-and-itsm`

IBM's approach to ITSM [6] underscores the value of structured methodologies that enhance IT services, aligning them with end-user needs and organizational goals. IBM's ITSM model focuses on improving user experience and maximizing IT infrastructure productivity through a range of processes that span incident and problem management to change and service level management. Notably, IBM incorporates artificial intelligence to build a proactive ITSM environment, with machine learning algorithms continuously optimizing IT systems under its AI for IT Operations framework. Acknowledging ITIL's influence on ITSM, IBM also integrates other frameworks like DevOps, COBIT, and MOF, highlighting ITSM's multifaceted nature. IBM's approach combines technology, best practices, and user-centered strategies through solutions like AIOps Insights and IBM Control Desk, establishing a comprehensive model of ITSM.

In its AMS offerings, IBM utilizes a next-generation AMS toolkit [7] and its extensive expertise to assist clients in effectively managing enterprise applications. With services anchored by IBM Consulting and its capabilities in automation, process mining, FinOps, and innovative engineering practices, IBM delivers high availability, cost-efficiency, resilience, and agility in AMS. IBM Global Services in Australia, for example, manages over 1,000 annual application development and support projects, consistently meeting schedule, budget, and productivity targets. This success underscores IBM's extensive experience and industry recognition, with IDC naming IBM as a global leader in AMS, affirming the company's significant impact in this field.

7.2 Process Mining with Business Management Frameworks: Literature Review

After extensive and manual research on the ITIL and AMS topics, we were able to define a proper systematic review flow. This systematic literature review investigates the intersection of PM and established business frameworks, specifically ITIL/ITSM and AMS. The study aims to elucidate the extent to which recent scholarly works have integrated PM with these frameworks to optimize, analyze, or redesign business processes within the IT domain.

The review leveraged three authoritative digital libraries: ACM Digital Library, Scopus, and IEEE Explore. The search strategy focused on titles, abstracts, and keywords, employing the queries described below:

- ("Process Mining" AND "Business Process Management") AND ("Information Technology Infrastructure Library" OR "Information Technology

[6] https://www.ibm.com/topics/it-service-management

[7] https://www.ibm.com/consulting/applications

Service Management")

- ("Process Mining" AND "Business Process Management") AND ("Application Management Services")

This approach ensured specificity to our domain of interest while affording a comprehensive breadth of literature. The ITIL and ITSM were included together with an *OR* operator since the latter one is considered a subfield of the former and both are often used interchangeably. The results were then subjected to a set of criteria designed to curate a relevant and high-quality dataset for analysis. Specifically, entries without titles were eliminated, and the scope was narrowed to journal articles and conference papers published from 2019 onwards. This temporal filter ensures the results reflect the most current version of the ITIL framework, as described in Table 7.1.

7.2.1 Quantitative Overview

The search culminated in a corpus of 61 academic papers, indicative of a modest body of research at the intersection of PM and the specified business frameworks. Notably, the absence of papers concerning AMS suggests a research gap or perhaps a nascent interest in exploring PM within the AMS context.

From the remaining 11 papers after filtering, the synthesis revealed a concentrated exploration into how PM can be employed along with the ITIL/ITSM processes and they are illustrated in Figure 7.1. However, after reading these papers we found out one of them was an outdated version from another paper and another one was not in English, thus we excluded both of them and finished the filtering phase with 9 papers only.

Several studies have underlined the potential of PM to provide actionable insights into ITSM activities, thereby facilitating a data-driven approach to continual service improvement—a core principle of ITIL/ITSM. A recurrent theme across the surveyed literature is the alignment of process mining techniques with ITIL's service lifecycle, particularly in the areas of service design and service transition. By leveraging PM, organizations have been shown to achieve a higher level of process visibility, allowing for the identification of bottlenecks, deviations, and opportunities for process optimization.

The discussion also touches on the role of PM in service operation, with empirical studies demonstrating how process mining can uncover inefficiencies in incident management, problem management, request fulfillment processes, and ticket analytics. The empirical evidence suggests a significant reduction in the mean time to repair and an increase in process conformance after the application of PM techniques.

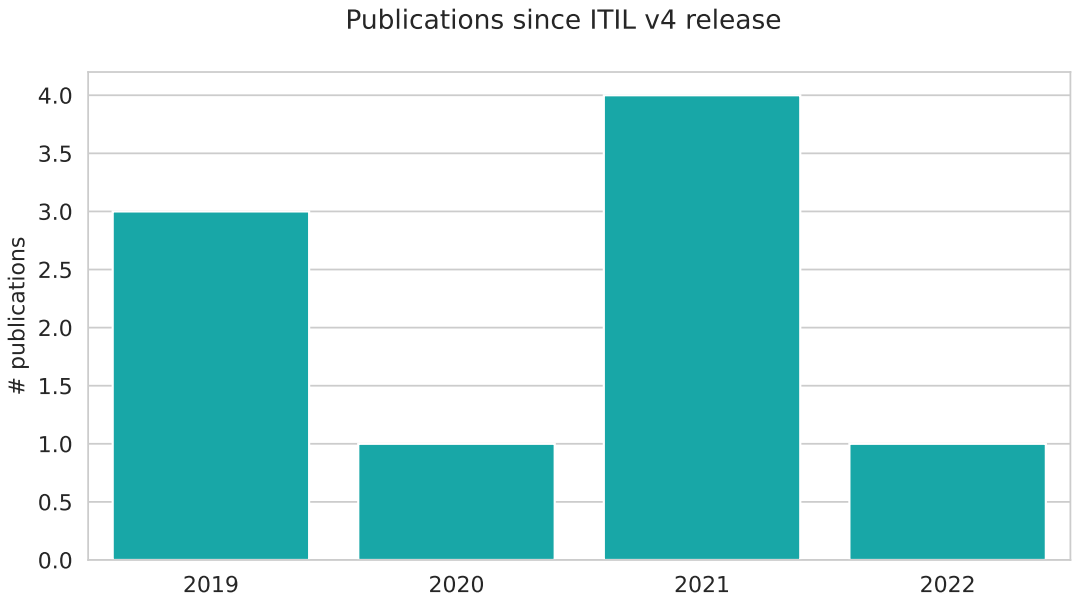


Figure 7.1: Number of publications retrieved over the past 10 years.

7.2.2 Qualitative Overview

Through a review of selected papers, we observe that applying PM within the ITIL framework can improve ITSM by better aligning operations with business needs. However, our systematic literature review reveals that only a few researchers have directly addressed the intersection of PM and ITIL, offering varied approaches and insights.

In [117] and [118], Pereira et al. examine how BPM principles can enhance ITIL's incident management. These studies show BPM's potential to improve time performance in ITSM by automating activities and optimizing technology. Case studies in both papers demonstrate BPM's ability to address inconsistencies in ITSM processes, driving efficiency and effectiveness. Additionally, Revina [130] explores the role of AI within BPM and its application to the ITIL framework. The author highlights the lack of formal AI integration frameworks and proposes research into methods that leverage Big Data, AI, and Machine Learning. This approach has significant potential for extending ITIL with AI to support decision-making and automate ITSM processes informed by PM findings. The study introduces a unique method to measure business process complexity, using the Theory of situation awareness and common NLP techniques to evaluate the cognitive and attentional demands on BP workers.

In a predictive context, [41] applies PM techniques to estimate completion times for ITIL-based incident management processes. This work emphasizes the

importance of selecting log attributes to improve prediction accuracy, which is essential for meeting Service Level Agreements (SLA) in ITIL processes. The finding that wrapper methods can enhance prediction accuracy points to a more proactive approach to managing IT service incidents. Focusing on anomaly detection, [87] investigates various approaches within PM to detect anomalies in ITIL incident management. This study compares techniques for improving process model accuracy, contributing to better process discovery. A neural-based anomaly filtering approach discussed here also opens pathways for integrating machine learning with PM, enhancing ITIL-based process refinement.

In another aspect of PM-ITIL integration, [131] uses the Strategic Alignment Model to analyze task content within ITIL processes. This paper underscores the importance of task alignment in ITSM and how PM can help ensure ITIL tasks support business strategies. Building on this, Rizun et al. [132] analyze task complexity in ITIL change management using NLP techniques to assess the cognitive demands of complex tasks. These insights are vital for understanding how task complexity impacts ITSM quality and efficiency. [109] introduces Met4ITIL, a method that applies BPM lifecycle and simulation modeling to improve ITIL processes. The flexibility and expert reception of Met4ITIL underscores PM's role in successful ITIL adoption and ongoing process optimization. Experiment results indicate that simulation models provide detailed insights into process performance, highlighting parameters like SLA compliance, incident resolution rates, and support group performance.

Additionally, recent studies, such as Eggert and Dorr [50], show how ITIL-aligned PM can advance ITSM in SMEs, emphasizing tailored PM methods. Gupta et al. [67] contribute with a practical tool for ticket analytics using text mining and clustering without depending on formal frameworks. Garcia et al. [45] expand PM applications across sectors, providing a context for studies like Swain and Goel [152], who use predictive analytics to improve SLA adherence in IT incident management. Bras et al. [18] address the research gap in combining Robotic Process Automation (RPA) and Intelligent Process Automation (IPA) with business continuity, proposing automation as an advancement within ITIL-guided ITSM.

7.2.3 An Intersection with Process Mining

The literature points to a promising synergy between PM and ITIL that could lead to more efficient, reliable, and customer-centric IT service delivery. However, it also underscores the need for further research, particularly in the context of AMS where the integration of PM could potentially drive similar improvements. Given the lack of research in the AMS domain, we could establish an exploratory study to establish a foundational understanding of how PM can be applied to

AMS processes. Additionally, future work should address the methodological and practical challenges of implementing PM in complex service environments, and how it interacts with IT governance frameworks.

This review contributes to the theoretical discourse by mapping the current state of research on PM within ITIL, identifying both the contributions and limitations of the existing body of work. Practically, the findings highlight the value of PM to IT organizations seeking to embrace a culture of continuous improvement in line with ITIL/ITSM principles. It also spotlights the research void in the context of AMS, providing an impetus for future academic inquiry. In conclusion, the research indicates that while the integration of PM and ITIL/ITSM is gaining academic attention, there is a conspicuous research gap in the context of AMS. The absence of literature on PM within AMS processes signals an opportunity for scholars and practitioners to pioneer research in this area. Meanwhile, the current findings surrounding ITIL/ITSM offer valuable insights into the transformative potential of PM but also suggest a need for deeper, more diverse investigations to fully comprehend its impact on ITSM.

7.3 Fundamentals

After presenting a broad review of business frameworks employed by Avio Aero and a systematic review of the intersection of PM and such frameworks, in this section, we will introduce the technical aspects for the understanding of our methodology and contributions. More specifically, we will dive into ServiceNow, a platform used to log the company's activities; and LLMs, used to enhance process workflows and derive meaningful insights. Other aspects, such as basic PM techniques, are omitted here since we have already introduced all the essentials in chapter 2.

7.3.1 ServiceNow

The ServiceNow platform is built on the ITIL framework. ITIL serves as the foundation for ServiceNow's **Incident Management** and **Incident Tasks Management**, setting a structured path for handling organizational issues and streamlining IT workflows. Within this framework, Incident Management in ServiceNow focuses on quickly restoring normal operations to minimize interruptions, prioritizing and escalating issues according to their impact on business continuity. Accordingly, Incident Tasks Management, breaks down complex incidents into manageable tasks, encouraging a systematic, collaborative approach to issue resolution. Analogously to PM, an incident represents a case, and an incident task represents an event. An incident consists of a sequence of tasks, which different

support groups can also handle in parallel. Table Table 7.2 shows this similar terminology, along with the ID template and descriptions for each component.

Table 7.2: The ServiceNow terminology aligned with PM terminology. Incidents can be seen as cases and Incident Tasks as events.

ServiceNow ID example	**Terminology**		**Description**
	ServiceNow	***PM***	
GEINC1234567	Incident	Case	Incident aims to quickly restore service, minimize business impact, and maintain quality by assigning incidents to an owner group responsible for resolution and creating tasks if other groups are needed.
GEINCTASK1234567	Incident Task	Event	Incident tasks allow the owner to coordinate with support groups, and track specific work for outage repair, and should not be reassigned; incorrect assignments must be closed and reissued correctly.

Support groups (SGs) within ServiceNow are specialized teams of IT professionals or experts dedicated to resolving incidents in their specific areas, such as networking, software, or hardware. Organized by expertise, these support groups enable incidents to be handled by the most capable team, which accelerates resolution and maintains the service working. This structure ensures that incidents are directed to those best prepared to address them.

Nevertheless, incidents are often **reassigned** between support groups when the initial group lacks the necessary expertise or resources to fully resolve the incident. Guided by ITIL principles of accountability and precision, this reassignment usually follows an initial assessment or diagnostic stage, where it becomes clear that a specific skill set or more specialized intervention is required. Reassignments also occur as incidents escalate, allowing higher-level support teams with broader authority to take over when needed. ServiceNow's flexibility in reassigning incidents across support groups promotes an adaptable approach to problem-solving, ensuring each issue is matched to the support group best suited for resolution. A flowchart example is illustrated in Figure 7.2, and we can see that users work on the task level, not on the incident. The incident will automatically close when there are no tasks anymore. Moreover, tasks cannot be reassigned: if a reassignment is necessary, the user first shall close the task and open a new one.

The ServiceNow platform provides several attributes for both incidents and incident tasks. However, for the purposes of this thesis, the only incident/case attributes taken into consideration are the identifier, opened at, and resolved at. On the other hand, the task/event attributes are deeply explored for our purposes

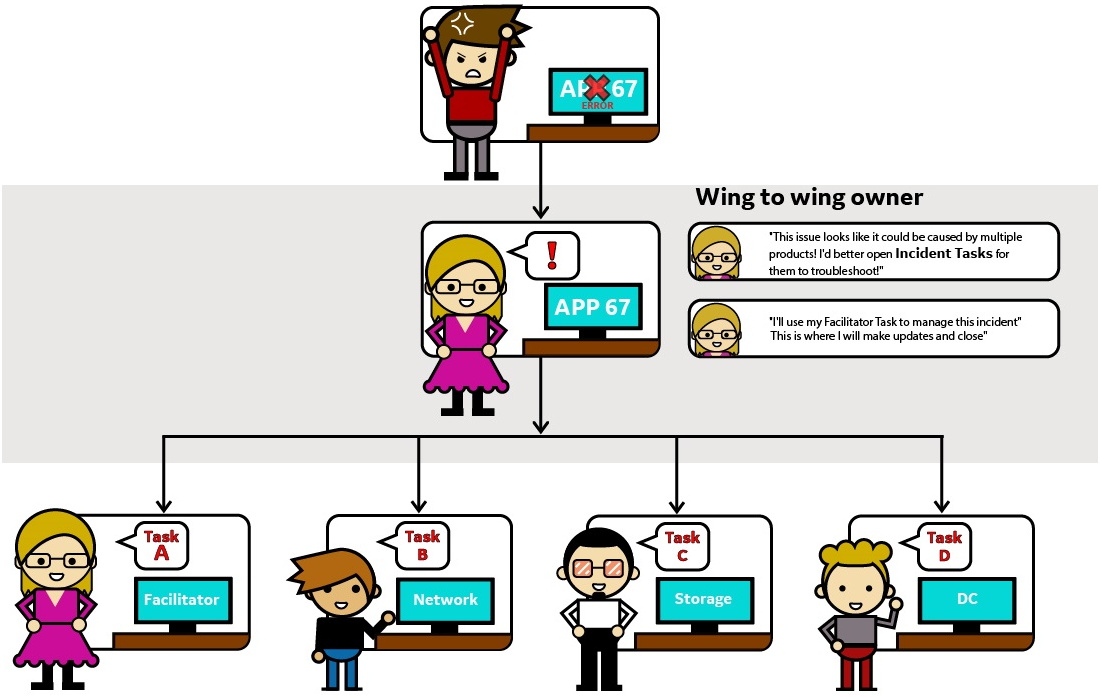


Figure 7.2: Flowchart example of a ServiceNow incident. An incident is composed of one or more tasks, and the solution might be reached by combining solutions from several support groups. Illustration adapted from the ServiceNow documentation.

and they are described in Table 7.3. The *state* is used at the preprocessing step since we filter out all the incidents where the last task is *in progress* or *on hold*. That means only finished incidents are considered.

7.3.2 Large Language Models

Generative models [81] are used in a range of fields, including image and video synthesis, text generation, drug discovery, and data augmentation. They are a type of machine learning model that creates new data instances similar to the training data they have seen. Instead of just predicting a label or category for input data (like traditional models), generative models learn the underlying patterns in the training data to produce new, synthetic examples that resemble the original data.

LLMs are advanced and modern machine learning models designed for this purpose, aiming to understand and generate human-like language [21]. They are trained using deep neural networks with billions of parameters, which allows

Table 7.3: Incident task attributes. Translating into PM terminology, this refers to event attributes.

Task Attributes	Description
State	The state the ticket is in. Includes: In Progress, On Hold, Resolved, Canceled, Closed
Opened at	Date of opening
Resolved at	Date of closing
Short Description	A brief description of the outage
Work Notes	The work notes recorded throughout the incident lifecycle (from each incident task)
Closed Notes	How quickly IT needs to respond
Assignment group	The support group identified to manage the outage

them to process complex language patterns and make predictions about words, phrases, or entire sentences. An LLM is trained on massive datasets containing text from books, websites, articles, and other sources to learn the structure and meaning of language. The effectiveness of handling such datasets is thanks to the transformer architecture [170], a key component of such deep networks. It has been the most effective architecture in the past few years to analyze context and understand relationships between words even when separated by long distances in a sentence. As a result, LLMs are very powerful for tasks like answering questions, summarizing text, translating languages, and even generating text that closely resembles human writing.

However, training an LLM requires significant computing power and resources, as the model needs to analyze large amounts of data. For this reason, many researchers and companies choose to use pretrained, publicly available models rather than training their own from scratch. These models, such as GPT or Llama, have been trained on large datasets and made available for reuse in various applications. By using pretrained models, developers can save time, reduce costs, and still achieve high-quality results, as the model already has a strong foundation in language understanding.

LLMs are available in two ways: as an online service (e.g., ChatGPT[8]) or as pretrained weights that can be downloaded and used locally (e.g., Llama[9]). Online services require users to send data to remote servers in order to get responses. For companies like Avio Aero, this scenario is often impractical since it involves sending private data outside the company, which can be risky. Sensitive information, such as user-generated content from emails or discussions from incidents, may be exposed in these interactions, raising privacy concerns. On the other hand, using models locally overcomes these privacy issues, as data stays within the company's own systems. However, another limitation arises in this scenario because running these models locally also requires significant computing

[8] `https://chatgpt.com/`

[9] `https://github.com/meta-llama/llama`

resources, which can be a barrier for some organizations.

In the context of LLMs, prompt engineering [175] is used to create input prompts to guide LLMs in generating accurate and relevant outputs. It plays an important role in optimizing LLM performance, especially when using online services or local models, as it can mitigate issues such as resource constraints or privacy concerns by enabling more precise interactions without extensive fine-tuning. Popular techniques include zero-shot, one-shot, few-shot, and chain-of-thought prompting. Zero-shot prompting relies on the model's pretraining to perform tasks without examples, while one-shot prompting includes a single example to demonstrate the desired behavior. Few-shot prompting provides a small set of examples, offering more context to improve task understanding. Chain-of-thought prompting enhances reasoning tasks by encouraging the model to generate intermediate steps, breaking down complex problems into smaller, logical parts. These techniques optimize LLM outputs, balancing performance with computational and privacy constraints.

In PM, LLMs have recently started to be leveraged for tasks such as automated process description generation and extracting process-related insights from unstructured textual data [15]. Their ability to understand and generate human language enables more intuitive interactions, bridging the gap between technical models and non-technical stakeholders, and enhancing the accessibility and interpretability of process mining insights.

7.4 Integrating Process Mining and Large Language Models at Avio Aero

This section explains our method for developing a solution at Avio Aero using PM and LLMs. We center our approach around four main research problems, validating our answers with three distinct use cases. Our goal is to apply these technologies to improve Avio's procedural data from the ServiceNow platform. We use PM to analyze and interpret such data, while we apply LLMs to leverage textual data from incident reports. By combining both, we aim to provide valuable insights that support decision-making, stakeholder understanding, and process improvement.

The research problems are:

- **RP1**: Can PM methods identify and reduce unnecessary incident reassignments among AMS groups?
- **RP2**: Can LLMs help us use textual data to generate useful insights for the organization?

- **RP3**: Can PM and LLMs be combined to automate ticket handling and establish standard workflows for similar incidents?
- **RP4**: Is it possible to define Key Performance Indicators (KPI) for evaluating AMS performance?

Regarding **RP1**: incidents are often reassigned when new tasks are created. Reassignments may happen because one support group can partially resolve the issue but needs to escalate it to another group, or tasks might be assigned to the wrong group due to initial misclassification. For example, a network issue might mistakenly go to the financial support group. Such misassignments are costly, leading to delays and resource waste. Our aim is to use process mining to confirm if this issue might or might not actually impact the organization. Moreover, by detecting loops in the discovered process models, we also aim to identify which support groups are most affected, using KPIs such as the time taken to close an incident and the number of reassignments involved.

To address RP1, we use process mining techniques to create a model of the process from incidents logged on ServiceNow. Here, task descriptions, manually provided by users, form the procedural data but lack explicit activity labels for task types. Therefore, we label activities by the support group name and analyze reassignments to highlight affected groups. Our results show that certain groups experience frequent reassignments, and this detection helps improve the process. The discovered model often resembles a "spaghetti" structure, with complex paths, so reducing reassignments simplifies and enhances the process.

Regarding **RP2**: we use LLMs for two major tasks and one minor task. The minor task regards the translation task, which simply translates incident dialogues between users from Italian to English. For **intent extraction**, the goal is to summarize the purpose of each incident. Often, incidents with the same intent have different descriptions or assignments. By grouping these cases through the labels derived from the LLMs, we can analyze patterns, such as common misassignments. Similarly, **conclusion extraction** classifies the reason for task closure, for instance: "reassigned: wrong support group". This information allows us to see which issues are frequently misassigned, supporting stakeholders in addressing recurring problems.

Regarding **RP3** and **RP4**: we solve these problems by combining PM and LLM insights. Insights from RP2 help us identify, within the process model from RP1, which support groups that face frequent incorrect assignments, as well as the intents most prone to misassignment. This approach allows us to define KPIs for tracking misassignments, understanding that not all reassignments are unnecessary: some are actually natural for solving the issue being handled. We define two KPIs: one for the rate of incorrect reassignments and the total number

of tasks per incident, with lower values being better. While this first metric characterizes a more global assessment that measures the performance of the whole process execution, we propose the second KPI that compares the daily elapsed time that a support group spends on tasks versus the elapsed time if wrong assignments had been avoided. This provides a more local assessment that captures narrowed and specific information for a given support group.

Therefore, we organize our experiments in three **use cases** that can be described as follows:

1. PM-analysis: discovering a process model, identifying loops, and calculating KPIs.

2. LLM-analysis: extracting the underlying intent of incidents and outcomes of each task.

3. Combining both: using insights from LLMs to locate bottlenecks in the process model.

7.5 Experiments

This section describes our experimental setup, the data preparation, and the proposed use cases to solve our research problems.

7.5.1 Experimental Setup

We build the event log by retrieving data from the ServiceNow platform. Each case is identified by an incident ID, and each event is marked by its task ID, with the support group name serving as the activity label. Before diving into our use cases, we perform a preprocessing step to refine the obtained event log. Since we employ a private LLM, privacy considerations are essential in this setup. To ensure data privacy, we anonymize identifiable information, such as support group names and task descriptions, early in the pipeline. This approach mitigates privacy concerns while maintaining the analytical value of the data.

Regarding the first use case, we perform the process discovery step using Apromore[10], creating a BPMN (Business Process Model and Notation) model [48] to visualize the workflow. This model shows task transitions, bottlenecks, and patterns in incident handling. For the following use cases, we leverage OpenAI's 'GPT-4o-mini' model to analyze qualitative patterns in the data, such as reasons for task reassignments. We opted for OpenAI's model due to resource constraints,

[10] `https://documentation.apromore.org/index.html`

as open-source alternatives like Llama require extensive local computational resources that exceed the available capacity.

The reliability of our evaluation methods for translation, intent extraction, and conclusion extraction is grounded in both established research and expert analysis. For translation tasks, language models have demonstrated remarkable proficiency, as validated by extensive public benchmarks and studies [129, 69]. This widespread validation allows us to trust their translation capabilities without conducting additional systematic evaluations in this thesis. When it comes to intent extraction, the absence of ground truth labels poses a challenge; however, expert reviews serve as a robust alternative. Experts can provide nuanced assessments of the language models' performance, ensuring that the evaluation captures subtle aspects of intent that automated metrics might miss. Similarly, for conclusion extraction, the language model's classification of conclusions into concluded or reassigned categories plus its inference of the reasoning behind them, is evaluated through a combination of visual inspection and expert reviews. The visual evaluation allows us to observe the clear separation of labels, indicating consistency and accuracy in classification. Expert reviews further reinforce this by providing an in-depth analysis of the model's inferential capabilities.

7.5.2 Event Log Extraction

This task has two main steps: querying data from ServiceNow and preprocessing the data. In the first step, we use the ServiceNow GUI to retrieve all tasks for incidents opened between April 1, 2022, and September 30, 2024. We also apply a filter to include only tasks assigned to support groups with "Avio Aero" in their names. Subsequently, we collect all incident IDs from these tasks and fetch those ones. This is essential because incidents may involve tasks assigned to other support groups besides Avio Aero, and we want the complete incident data. By doing our proposed filtering, we are initially excluding those alternative groups.

In the second step, we preprocess the data by filtering out incomplete cases. Specifically, we exclude incidents where the last task remains open. Then, we keep incidents with at least three tasks and at least three reassignments. We do this because incidents with fewer tasks or reassignments tend to resolve quickly, and our goal is to identify patterns of repeated reassignment and understand their causes. Figure 7.3 shows the number of incident reassignments before and after preprocessing. Most incidents have only one reassignment, indicating they were assigned correctly from the start. We focus on the 837 incidents with three or more reassignments for further analysis.

The final preprocessing step is the anonymization of work notes. We mask user IDs, IP addresses, and email addresses, and we remove dates and locations. For example, Avio Aero users are typically listed as 'Surname, Name (ID num-

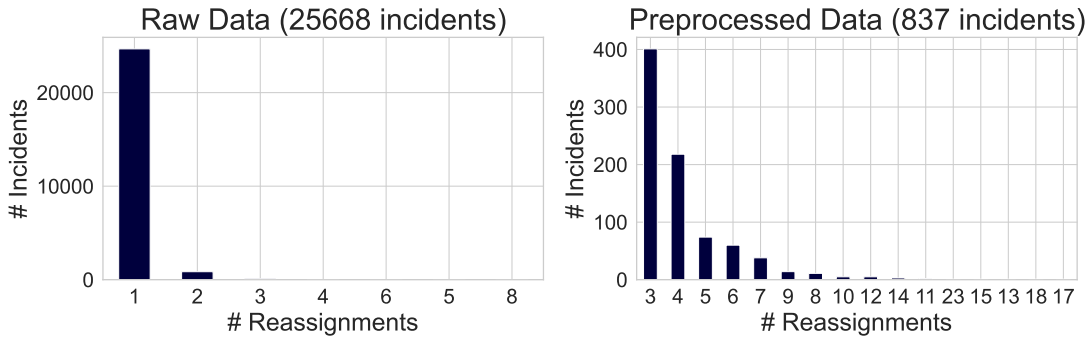


Figure 7.3: The raw data fetched from ServiceNow (left) and the preprocessed data (right).

ber)', which we can detect using a regular expression. We also filter out any lines in the work notes that contain emails, dates, or addresses. To ensure that only valid information remains, we implemented a parser to process this text data. Table 7.4 shows an example of two real incidents after preprocessing. Note that some work notes provide more or less information. For instance, the event '*TASK001*' provides a clear description of what is happening and the reason behind it: the incident is being reassigned to another group due to the current one (*Accouting*) not being the right one to solve the issue. On the other hand, '*TASK005*' describes only the fact it is being reassigned, but not the reason for that.

At this point, a new limitation emerged due to the extensive length of some work notes. Although this is not the case in the illustrated examples, some work notes are very long. To fit within our budget when running experiments using the employed LLM, we had to limit each work note to 1000 characters. Therefore, the final event log includes 837 incidents and 3552 tasks, with about 500 tasks impacted by this chunking step.

Table 7.4: Two real incidents after all the preprocessing steps.

Case ID	Event ID	Opened at	Closed at	Sup. Group	Work notes
INC123	TASK001	07/09/22 11:01:46	07/09/22 11:10:25	Accounting	Task Closed Skipped with close notes: Incident changed assignment group. not in our scope. Assign it to application team (refer to <HIDDEN_NAME>) | Description: Lack of possibility for digital WO to interrupt HT with not compliant result
INC123	TASK002	07/09/22 11:10:26	07/09/22 11:24:18	GE	Task Closed Skipped with close notes: Incident changed assignment group. | Description: Lack of possibility for digital WO to interrupt HT with not compliant result
INC123	TASK003	07/09/22 11:24:18	07/09/24 12:39:40	ERP Support Group	Task Closed Skipped with close notes: Incident has been marked resolved or canceled. Task state is moving to close skipped. | Description: Lack of possibility for digital WO to interrupt HT with not compliant result
INC456	TASK004	16/09/2022 12:26:35	16/09/2022 14:11:12	CAE CAT	Task Closed Skipped with close notes: Wrong assignment to the support group, the user doesn't belong to Avio Aero company. MyTech <HIDDEN_NAME> (<HIDDEN_ID>). File count: 2. File extensions: .png(2) | Description: Internet settings issue
INC456	TASK005	16/09/2022 14:11:20	16/09/2022 14:21:12	GE	Task Closed Skipped with close notes: Incident changed assignment group. | Description: Internet settings issue

7.5.3 PM Analytics

In this section, we present our use case using process mining to evaluate and analyze the event log built from Avio Aero's information system. In Table 7.3 we map the ServiceNow terminology to the PM terminology. Nevertheless, throughout this section, we use both terminologies, according to each specific context. Thus, as a quick recap: incidents are cases, tasks are events and support groups are activities.

In this application, we achieved significant milestones. First, we conducted process discovery using the Apromore tool, producing a BPMN model of the underlying process. This model provided an initial visual and structural representation of the workflow. However, the discovered model was characterized by a spaghetti model, as partially illustrated in Figure 7.4, which limits interpretation and analysis. Therefore, we enhanced the process model by filtering activities based on their statistical characteristic: the relative frequency.

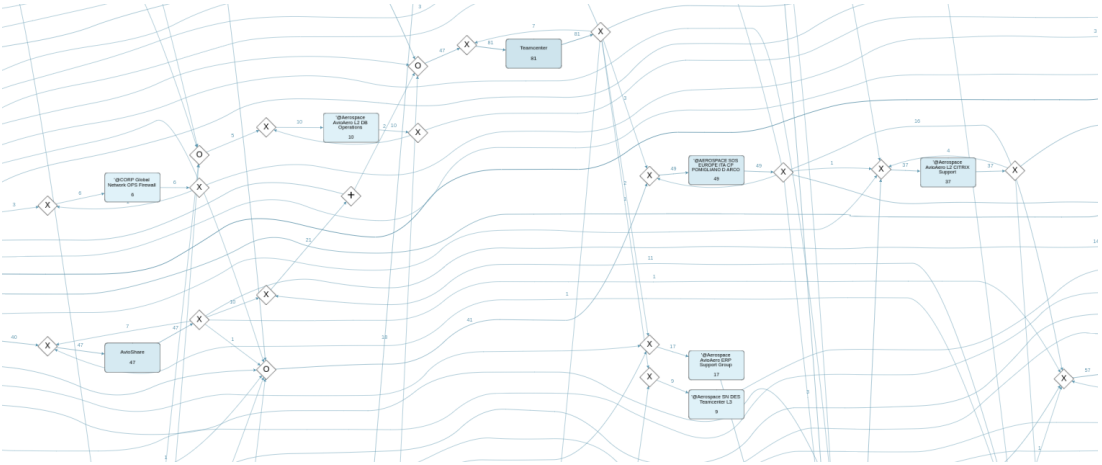


Figure 7.4: A tiny piece (for better visualization) from the discovered spaghetti process model.

The activities filtered out were first evaluated to ensure they were not going to impact the process model. We realized that these activities were wrongly instantiated and most of them were the first activity. What actually happens is that although traces are created in the ServiceNow platform, there is a third-party service that also automatically starts a case. Thus, users from this third-party service assigned tasks to support groups that were not suited to solve the given issue, which consequently caused several reassignments. From the original process model, we calculated the total execution time to complete all the traces. From the new process model dropping activities with frequency less or equal to two, we achieved a 13% reduction of total execution time. Nevertheless, the spaghetti

aspect persists. However, the behavior characterized as a bottleneck was detected and reported so stakeholders could handle and improve the platform for better performance.

Diving into this execution time KPI analysis, we evaluated the execution time of process instances. The distribution of these times, visualized in various time intervals in Figure 7.5, showed that most instances require over 72 hours to complete. This insight led us to investigate further. Instances that take this long are often characterized by cycles: a sequence of activities that starts and ends at the same activity. We analyzed all the activities that fall within such cycles to check how often a given activity is characterized by this problem and what is its average execution time.

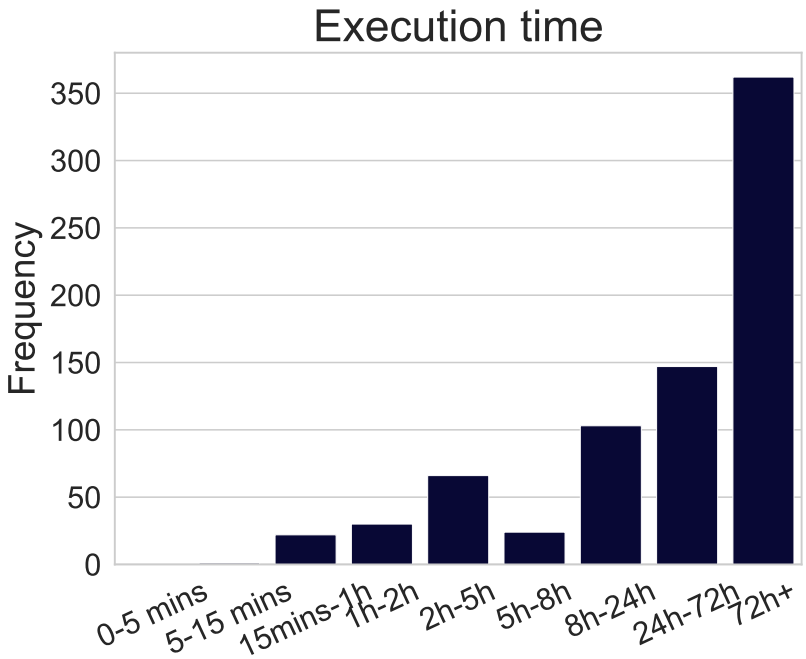


Figure 7.5: Execution time of intents.

This is illustrated in Figure 7.6. This analysis helped us pinpoint the support groups contributing the most to prolonged process durations and detect the major bottleneck we were looking for. An interesting insight is that the most frequent activity falling within cycles does not reflect the one that takes longer to execute. The 'GE' activity is the most frequent one to be part of cycles but the 'GE SAP SUpport' is the one that takes longer to execute. This is due to the fact the 'GE' activity receives most of the wrong transitions: every time an event is created and assigned to the wrong support group, the trace is reassigned to the 'GE' group and this group tries to find the correct group.

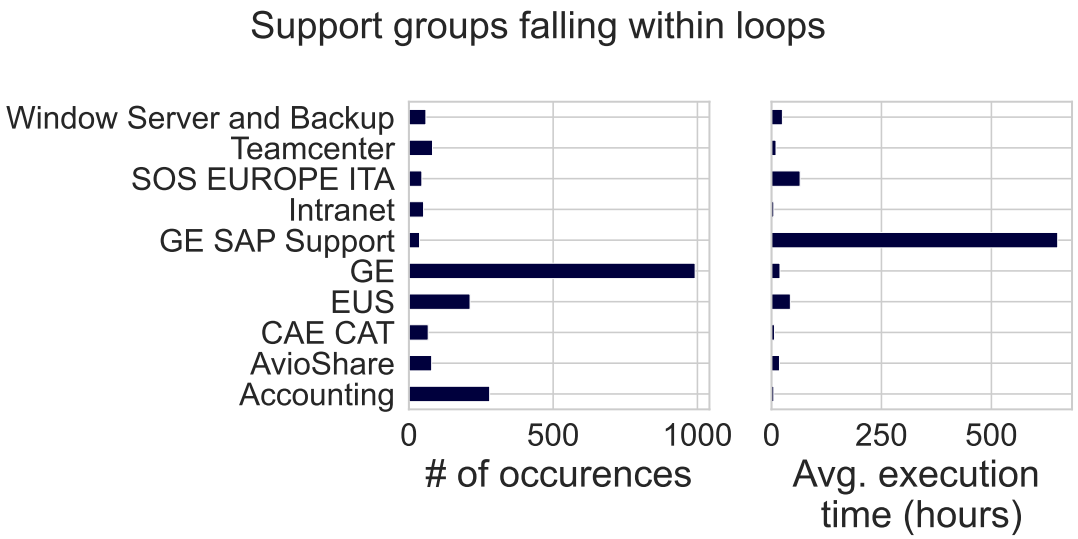


Figure 7.6: Group falling within loops

Our cycle analysis identified several repeating cycles in the process. To explore the impact of these loops, we removed these cycles by excluding the events from the cycle except for the first one. This adjustment reduced overall execution times significantly. However, this new model without loops still requires validation from process experts to confirm its effectiveness and accuracy. The new execution time KPI is illustrated in Figure 7.7, where the left bar shows the execution time from the original process model and the right bar from the enhanced model.

Subsequently, we performed a variant analysis to identify common patterns in process instances. This analysis, illustrated in Figure 7.8, highlighted a classic Pareto distribution [107], suggesting that a small number of process variants account for the majority of cases. This behavior is expected in most event logs and it is also known as the 80-20 rule: 80% of the cases are executed as expected and the remaining 20% account for the natural operational variability. Furthermore, infrequent variants are usually the focus of further, deeper analysis that often needs to go through repair and rework procedures.

Another popular benefit brought by PM is the possibility of automating parts of a process model. Aalst [163] proposes a quick strategy to detect which parts of a process can be automated and how. Basically, analyzing the process variants, we can group them into three parts: (i) regular high-frequent subprocesses automated in the traditional way, (ii) frequent standardized subprocesses taken over by robots, and (iii) infrequent and/or exceptional behaviors still handled by people. Clearly, each problem domain has its own peculiarities and must be investigated accordingly. Therefore, the decision to automate a process variant

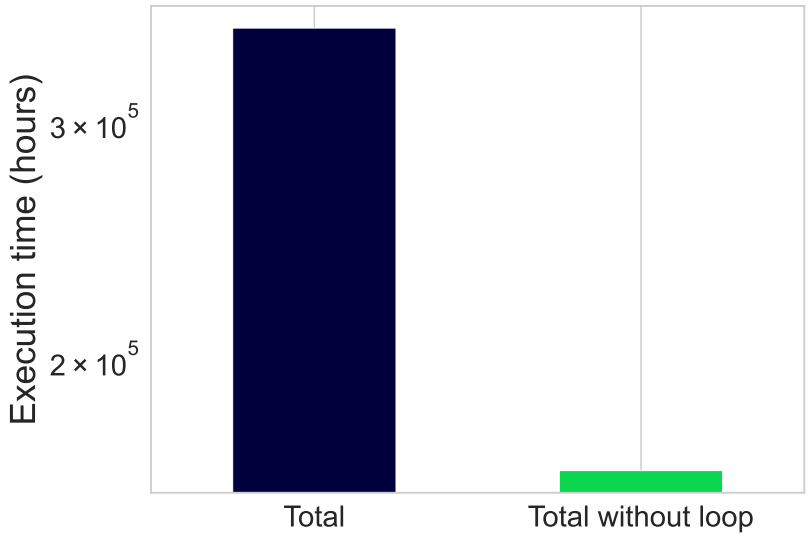


Figure 7.7: Elapsed time with and without loops.

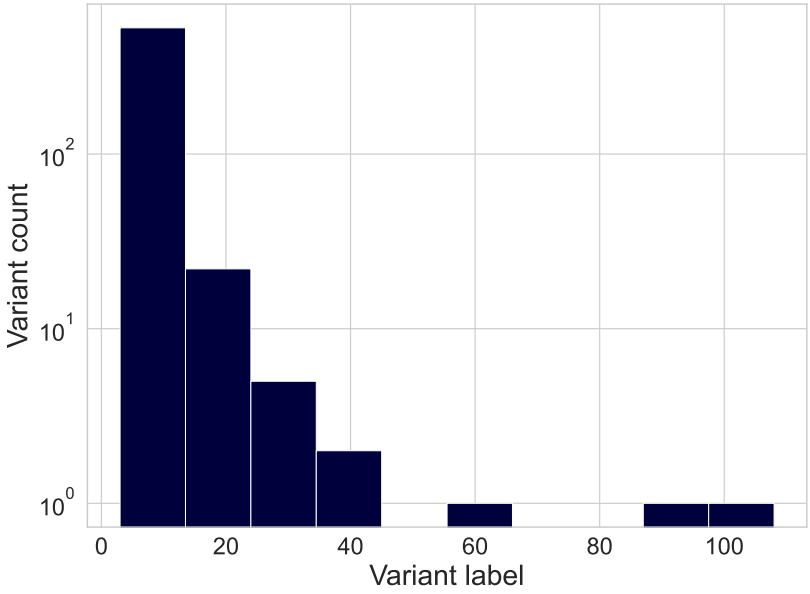


Figure 7.8: Variant distribution.

depends on its frequency, complexity, and impact on the overall process. This analysis will support stakeholders from Avio Aero to take a final decision in the future.

Following this analysis of grouping variants by their frequencies, we can also evaluate another KPI to better understand our process. As previously men-

tioned, the infrequent variants usually have to go through repair procedures due to their problematic nature. This is reflected by the total wait time summed up through each incident. In Figure 7.9, we group the variants according to their frequency. Roughly, we count the frequencies and segment them into three bins: low, medium, and high frequency. We can see that high-frequency variants are well-behaved due to their very low wait time values. The variance increases accordingly as the frequency decreases, showcasing the difficulty of analyzing and handling these variants.

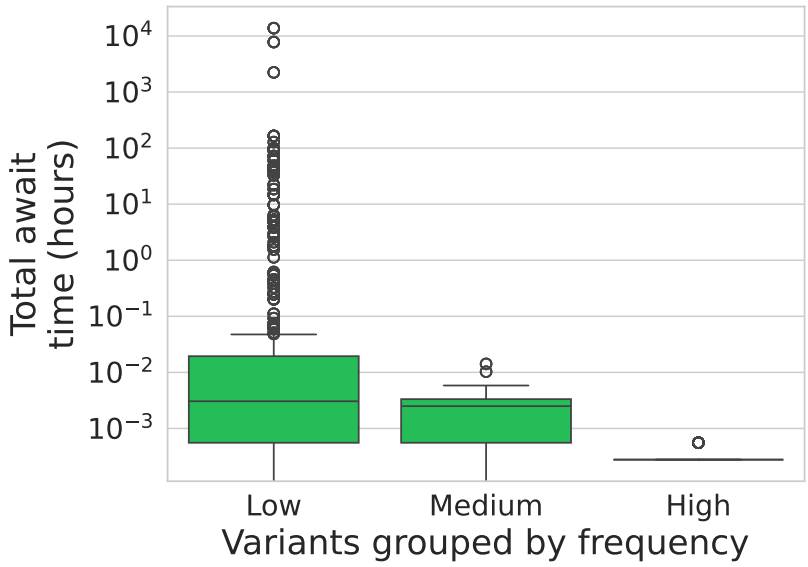


Figure 7.9: Await time according to each variant frequency group.

After applying the frequentist analysis to this use case, we produced a significantly enhanced process model. The model shows an improvement in clarity, as demonstrated in Figure 7.10, where the previous spaghetti structure was greatly simplified. Furthermore, the total execution time required to resolve all incidents was reduced by an impressive 72%. This reduction was achieved by identifying and removing infrequent variants, many of which were clear outliers. These outliers represented incidents that often remained open for extended periods (in some cases exceeding a year) and were eventually closed without any meaningful explanation in the work notes. Moreover, we also reduced in 45% the number of incidents with cycles, by successfully answering and handling RP1. This is remarkable, especially due to the fact it was achieved by simple techniques without taking into consideration contextual information. This emphasizes the effectiveness and capabilities of using PM.

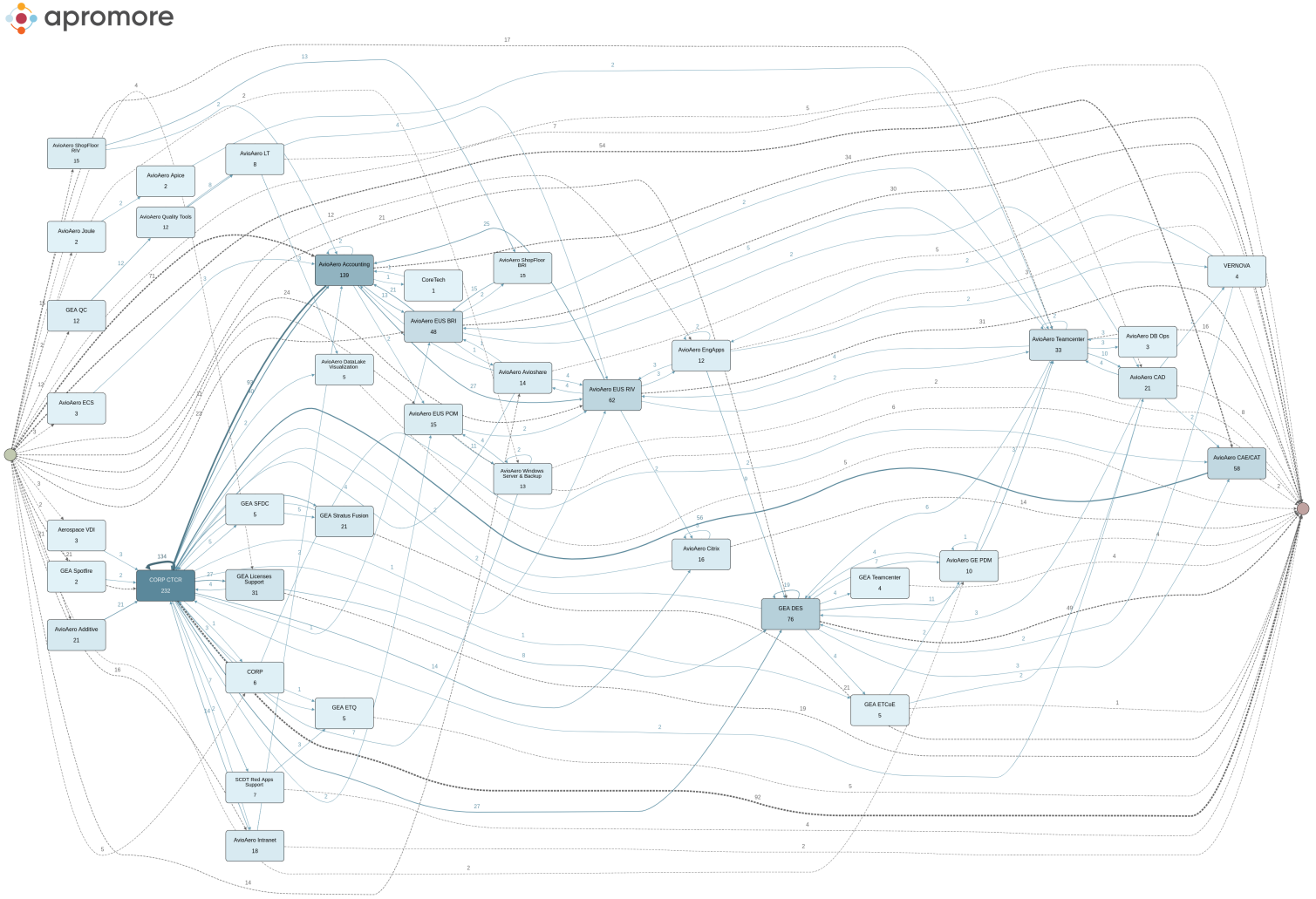


Figure 7.10: Enhanced process model.

In the next section, we will leverage an LLM to extract valuable insights from these work notes, providing further context and understanding.

7.5.4 LLMs

In our second use case, we explore the application of an LLM to improve our event log analysis and address our second research problem (RP2): *Can LLMs help us utilize textual data to generate meaningful insights for the organization?* Specifically, we employ the LLM to extract two types of labels from our data. The first label operates at the case-level, where the goal is to identify the primary **intent** behind the entire process execution or the lifecycle of a ticket. This label provides a broader understanding of the purpose of the case. The second label is at the event-level and aims to infer the **conclusion**, summarizing the reasons and context for the completion of individual events within the process. However, before tackling these labeling tasks, we must solve a minor challenge: the textual data stored as work notes is multilingual, with some parts written in English and others in Italian. To address this, the initial task assigned to the LLM is the **translation** of all textual data into a single, consistent language (English). Only after completing this preliminary step can we move on to the case-level and event-level labeling tasks. In the following section, we will provide detailed explanations of the prompts designed to execute each of these tasks effectively.

Prompts

Listing 7.1 illustrates the proposed prompt for translation. Note that we use Python syntax, which means the braces '' will be filled by the input work note.

Listing 7.1: One-shot prompt proposed to translate into English any piece of text that is found in Italian.

```
Read the following piece of text delimited by triple
   backticks. Your first task is to detect if there is
   any piece of text in Italian. If so, translate it to
    English and keep the original text flow. For
   example, a text can be written as follows:

- TEXT: ```This is an automatic message. Here is an
   arbitrary message: Buongiorno, come stai?```

In this case, your output must be:
```

```
- TRANSLATED TEXT: ‘‘‘This is an automatic message.
   Here is an arbitrary message: Good morning, how are
   you?

Let’s get started. Here is your text to be translated
   to English when necessary:

- TEXT: ‘‘‘{}‘‘‘
```

We now introduce the prompt to extract both the conclusion and intent from incident tasks[11]. We employed the few-shot prompt engineering technique, which guides the model through a sequence of reasoning steps through examples to reach the final answer. In our case, these examples are the tasks themselves, manually labeled by experts; we aim to observe how the tasks evolve over time and capture each label of interest at the appropriate moment (either conclusion or intent). Essentially, each subsequent task is influenced by the previous ones because they are interconnected. Listing 7.2 illustrates the proposed prompt.

Listing 7.2: Few-shot prompting proposed to extract the intent of incidents.

```
    An incident is a process execution that is composed
        of a sequence of tasks. As the manager of your
       company’s incident system, you are responsible
       for overseeing tasks that include work notes
       serving various purposes:

    - Describing the issue
    - Facilitating discussions between users
    - Determining correct assignment to support groups

    Our goal is to resolve the first task in a
       lifecycle of an incident as soon as possible.
       However, if the work notes of a task contain
       terms like ‘Reassigned’ or ‘Forwarding’, it
       indicates the issue is being transferred to
       another support group, which increases company
       costs. Additionally, some work notes may include
        hidden anonymized information marked by tags
       like ‘<HIDDEN_TOKEN>’.
```

[11]We chose to use a single prompt for both tasks due to resource constraints, as OpenAI’s service is not free. While the results were good, we believe that using separate prompts could further enhance the final outcome.

```
Your task is to identify two key pieces of
   information:

1. **INTENT**: This describes the problem the
   entire incident lifecycle aims to address. The
   intent **must be** consistent and unique across
   all tasks within the same lifecycle. Determine
   this after reading all the provided work notes.

2. **CONCLUSION**: This explains how each
   individual task was concluded or why it was
   reassigned to another team.

When a new lifecycle starts, you will receive a
   work note beginning with `- TASK:` followed by
   its content. For each task, provide a response
   in the following format:

- **CONCLUSION:** ```Outcome or reason for
   reassignment'''.

Finally, after reading all tasks, you must provide
   the final intent of the whole incident, using
   this format:

- **INTENT:** ```Type of problem'''.

Below, find a few examples of how to extract and
   format the desired information:

Example 1: ```{}'''

Example 2: ```{}'''

Example 3: ```{}'''

Now, here is a real incident lifecycle for you:

```{}```
```
```

An example of the final outcome is illustrated in Listing 7.3. This template format is ensured by formatting each example in our prompt accordingly. The input case was opened by a user requesting database access. Initially, the user assigned the event to the wrong support group, which resulted in a cascade of reassignments until the right support group was reached. The 'escalated' conclusion means a support group partially solved the issue and escalated it to the next support group to fully resolve it. This specific reassignment illustrates a necessary reassignment since some issues can only be fully solved by the collaboration of a few support groups. Detecting such reassignments is crucial for the automation, interpretation, and enhancement of the process model.

Listing 7.3: Outcome example when using our few-show prompting.

```
CONCLUSION: ‘‘‘REASSIGNED: wrong support group’’’
==========
CONCLUSION: ‘‘‘REASSIGNED: wrong support group’’’
==========
CONCLUSION: ‘‘‘REASSIGNED: wrong support group’’’
==========
CONCLUSION: ‘‘‘REASSIGNED: wrong support group’’’
==========
CONCLUSION: ‘‘‘REASSIGNED: escalated’’’
==========
CONCLUSION: ‘‘‘CONCLUDED: permissions granted’’’
==========
INTENT: ‘‘‘Database access permissions’’’
==========
```

Assessing the Effectiveness of our LLM

Following a review process with Avio Aero experts, we confirmed the accuracy of the new labels extracted for each incident and task. Since this is the first dataset built for this purpose, we lack a suitable evaluation metric, although such a limitation will be addressed in the future. On the other hand, a key limitation emerged: out of 837 incidents, around 600 unique intents were identified, making interpretation difficult. This issue extended to the conclusions as well. To address this, we used our LLM to generate embeddings for these labels in order to cluster them, enabling us to reduce redundancy.

To consolidate similar intents and conclusions, we applied the DBSCAN clustering algorithm. This approach significantly reduced the number of unique labels, making the dataset more manageable and insightful. For example, prior to clustering, the LLM assigned specific intents such as 'network connectivity issues'

and ‘network access issues‘, even when varying the LLM temperature hyperparameter. By embedding and clustering these intents, we simplified them to a broader label: ‘network issues‘. This step enhanced our ability to analyze and interpret the results effectively.

Figure 7.11 shows the most frequent original generated intents, highlighting redundancies for the aforementioned examples. On the other hand, Figure 7.12 presents the most frequent clustered labels, demonstrating the successful reduction of unique intents and improved clarity. Specifically, from an initial 837 incidents, we originally obtained 662 intent labels. After the clustering process, this was reduced to 67 labels, reaching a 71% reduction of unique labels.

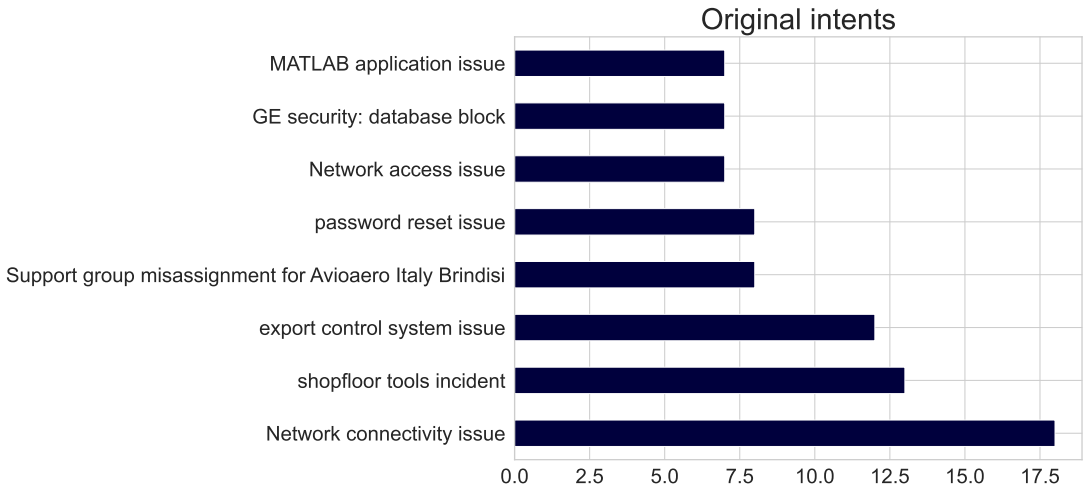


Figure 7.11: Intent labels originally inferred by the LLM. Similar intents were obtained, such as ‘Network connectivity issue’ and ‘Network access issue’.

Figure 7.13 shows the top 10 conclusions extracted by the LLM to illustrate the original format. After the expert feedback, we further assessed the conclusions’ embeddings through visual analysis. To achieve this, we applied the t-SNE algorithm to map and visualize the conclusions, offering a clear view of the clustering patterns. In order to facilitate the visualization and validate this methodology, we split each conclusion string, which originally followed the format ‘(label): (reason)’, where ‘(label)’ represented either ‘CONCLUDED‘ or ‘REASSIGNED‘ and ‘(reason)’ label described the justification.

The t-SNE plot, shown in Figure 7.14, illustrates clear clustering between the categories. It is worth noting that while the labels were split for visualization purposes, the full strings were used to generate the embeddings. Although there are minor overlaps (such as one REASSIGNED label near the CONCLUDED cluster and two CONCLUDED labels near the REASSIGNED cluster) the overall

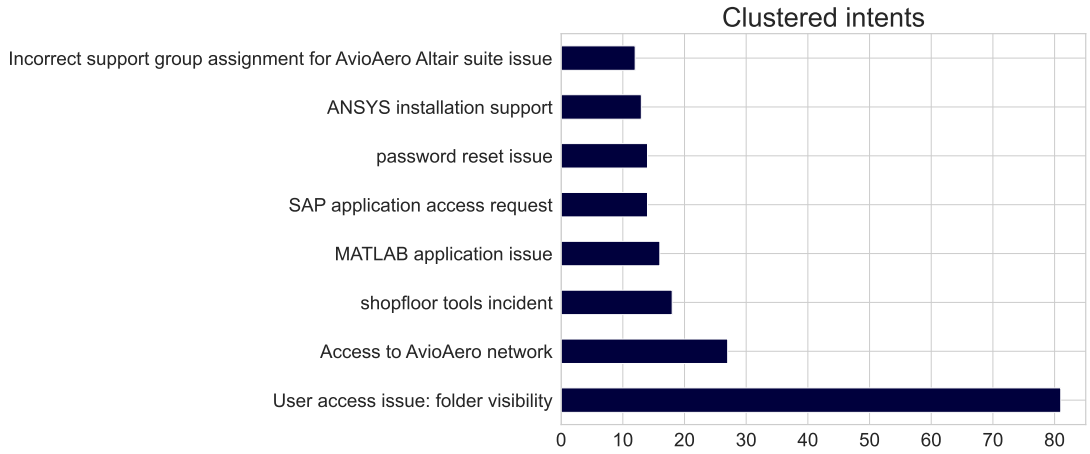


Figure 7.12: Intent labels after clustering their embeddings.

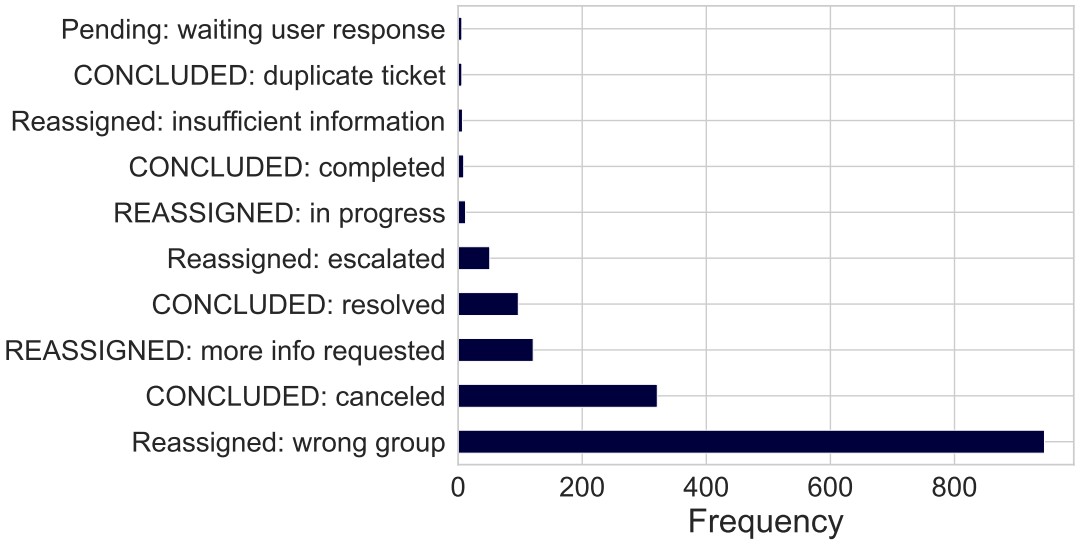


Figure 7.13: Conclusions derived by the LLM.

separation between clusters remains distinct. These findings demonstrate that the embeddings derived using the LLM effectively differentiate between categories, reinforcing the reliability of our approach and boosting confidence in the LLM's output. This was not an intended task initially, but we successfully proposed a suitable solution to overcome the issue raised by LLM regarding several different intent labels with similar meanings.

The methodology proposed in this section effectively addressed our RP2, demonstrating that the textual data could indeed provide valuable contextual insights. Through validation, we confirmed that it is possible to uncover key

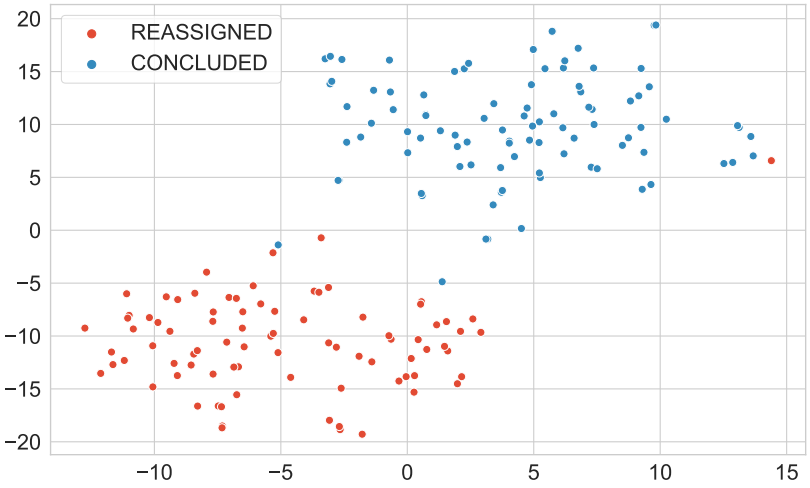


Figure 7.14: t-SNE plot of conclusion embeddings.

factors, such as the reasons behind frequent incident reassignments. The newly derived labels not only enriched our understanding but also allowed us to extend our analysis by integrating the insights generated through the LLM with PM techniques. In the next section, we present our final use case, where we discuss the additional achievements and further explore the impact of these integrated methods.

7.5.5 Combining PM and LLMs

This section presents our last use case, which combines PM techniques with an LLM to improve the organization's issue handling from Avio Aero and propose standard workflows for incident management. In our previous section, we validated the intent and conclusion extraction step using an LLM, and we now use the gathered knowledge to solve the two remaining research problems (RP3 and RP4). Regarding RP3 (*Can PM and LLMs be combined to automate ticket handling and establish standard workflows for similar incidents?*), we solve this problem throughout the section. For RP4 (*Is it possible to define Key Performance Indicators (KPIs) for evaluating AMS performance?*), we introduce now the proposed KPIs and discuss them through the section as well.

We introduce two KPIs:

- Execution Time by Intent (ETI): This KPI calculates the time taken for each intent and identifies bottlenecks. Analyzing this KPI revealed areas where delays occur, guiding future improvements to the process model.

- Wrong Reassignment Ratio (WRR): This KPI measures the proportion of frequent activities (support groups) that are incorrectly reassigned. By filtering frequent activities and their intents, we calculated the ratio of wrong reassignments to activity frequency. This metric highlights inefficiencies in task assignments.

Obtaining ETI is quite straightforward: we group the event log by the intents and sum the execution time. On the other hand, the WRR is calculated by first grouping the data based on support groups. For each support group, we count its frequency. Then, we identify and count the number of wrong reassignments within that support group. Finally, we divide the number of wrong reassignments by the total count of activities for the support group. This ratio represents the proportion of incorrect reassignments relative to the assigned support group. Summarizing, this metric is formally defined as follows.

$$WRR = \frac{\text{Number of wrong reassignments for a support group}}{\text{Total number of activities assigned to the support group}}$$

In Figure 7.15, we focus on the intents with the longest ETI values, sorting the intents on the y-axis by their ETI values as derived from the event log. The 'SAP application access request' stands out with the highest average time to be resolved, measured in hours. Upon closer examination, we realized that many of these cases were never actually resolved by eventually marked as resolved after a long time. This pattern makes these cases outliers, requiring further investigation by stakeholders. In contrast, the 'password reset issue' is among the most frequent intents and is resolved much faster than others, such as the 'MATLAB application issue' and 'ANSYS installation support.' These results highlight significant variability in resolution times across intents. Furthermore, identifying faster and slower intents is remarkable for the company.

Analyzing ETI provides a useful initial insight, but combining this with intent frequency yields a deeper understanding. Referring to Figure 7.12 in the previous section, we observe that the 'SAP application access request' is not a frequently occurring intent with respect to the remaining intents. This, along with its outlier behavior identified through the PM plus the LLM analysis, confirms its status as a bottleneck. Stakeholders could address this issue in future process improvements. Additionally, for the two most frequent intents ('Access to AvioAero network' and 'User access issue: folder visibility'), the average ETI is significantly lower (close to an order of magnitude less) than most other intents. However, we show in the remaining section that our analysis reveals that these intents are often incorrectly (re)assigned, resulting in longer resolution times despite being relatively straightforward issues. This mistake represents another inefficiency that could be addressed to improve overall process performance.

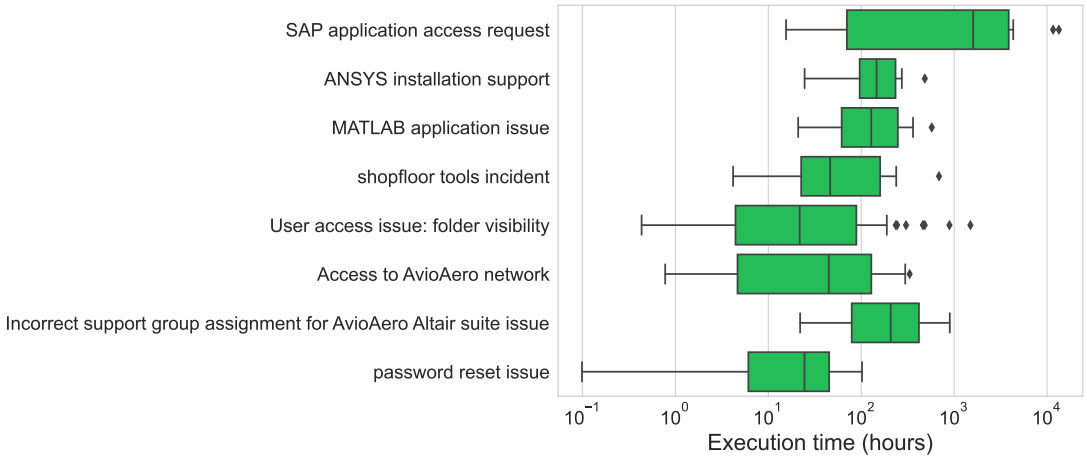


Figure 7.15: Top 8 largest ETI values. The intents on y-axis are sorted by their average execution time.

In order to contextualize the second KPI, we illustrate support groups with the highest rate of wrong reassignments. This analysis illustrates the potential to reduce inefficiencies through better process design. We extended the insights from Figure 7.9 by analyzing process variants in relation to intents. This analysis confirmed that similar intents correspond to similar variants. It also revealed that cases with the same intents but divergent variants often result from additional reassignments, reinforcing the need for standardization and automation. Let us now showcase this analysis in detail.

First, we begin by filtering the two most frequent intents identified in the previous analysis to examine which support groups they are assigned to. The first intent is consistently assigned to three support groups: 'L2 Accounting,' 'CTCR SD Global,' and 'CTCR SD EMEA,' as shown in Figure 7.16. For the second most frequent intent, we observe overlap in support group assignments, specifically 'L2 Accounting' and 'CTCR SD Global.' This overlap suggests a similarity between the two intents, even after the clustering procedure. Their shared support groups further support the idea that these intents may belong to similar process variants.

Next, we analyze the conclusions extracted for all the tasks from these intents, as shown in Table 7.5. The most frequent conclusion identified is the wrong reassignment, highlighting inefficiencies in handling these intents. Given their frequency, resolving these intents faster should be a priority, but this requires minimizing the number of incorrect assignments. Additionally, a recurring problem emerges: similar to the 'SAP application access request' intent discussed earlier, many incidents are incorrectly marked as resolved after being reassigned

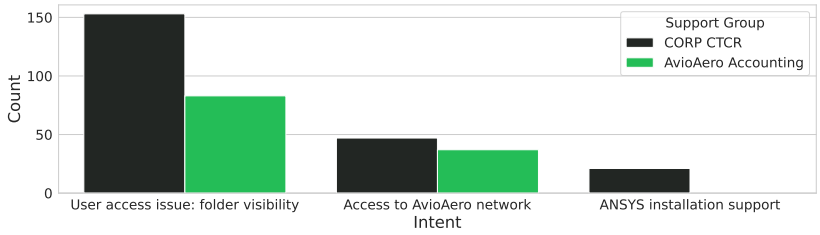


Figure 7.16: Top support groups and their most frequent intents. The frequency of assignments for each support group (y-axis) according to the intent (x-axis).

to the wrong support group ('CONCLUDED: canceled'). In these cases, after communicating with experts from the company, it appears the user who opened the incident manually marked it resolved without adding relevant remarks. This behavior suggests a need for company-wide training or clearer guidance on incident management best practices to align with the AMS framework. Finally, it is important to notice that some reassignments are necessary. The reassignment requesting more information about the incident is an example of such a scenario and illustrates a normal, essential step.

Table 7.5: Conclusion count regarding the most frequent support groups and intents.

Conclusion	**Count**
REASSIGNED: wrong support group	171
CONCLUDED: canceled	46
REASSIGNED: more info requested	33
REASSIGNED: escalated to correct support group	21
CONCLUDED: resolved	16

We then calculate the WRR values for the identified support groups, as presented in Table 7.6. Interestingly, 'GEA DES' and 'AvioAero EUS RIV' are the third and fourth, respectively, most frequently involved support groups but have the lowest WRR, suggesting that previous incorrect reassignments should be avoided by assigning incidents directly to this group. In contrast, the most problematic support group is "AvioAero CAE/CAT,' which has an alarming WRR of 96.15%. To illustrate this issue, we examine a real-life example involving this support group.

In Table 7.7, we select an arbitrary incident where 'AvioAero CAE/CAT' appears at least once. The table includes details such as the incident's conclusion, execution time (in hours), support group involvement, and work notes. For

Table 7.6: The most frequent support groups and their ratio of wrong assignments. SG stands for support group and WR for wrong reassignments.

SG name	SG count	WR count	WR Ratio
CORP CTCR	512	321	62.70
AvioAero Accounting	204	122	59.80
GEA DES	105	43	40.95
AvioAero EUS RIV	79	29	36.71
AvioAero CAE/CAT	78	75	96.15

readability, we abbreviate the work notes with '...' and highlight in bold key sentences to clarify the incident's progression.

In the first event, the work notes reveal that the user reporting the issue does not belong to Avio Aero, which remains a problem in subsequent events. In the third event, another incorrect reassignment perpetuates the delay. The fourth event again routes the issue to the wrong support group. Finally, the last event records a solution, with the work notes documenting positive interactions between the user and the support assistant. If the incident had been correctly assigned from the beginning, the resolution time could have been reduced by 12.38 hours, showcasing the significant impact of proper initial assignments.

Table 7.7: An example of an incident where the support group "AvioAero CAE/CAT" is involved. This support group has the highest ratio of wrong (re)assignments. The incident's intent is the 'ANSYS Tool Suite installation issue'.

Conclusion		Exec. time (hours)	Support group	Work notes
REASSIGNED: support group	wrong	0.34	AvioAero CAE/CAT	Task Closed Skipped with close notes: Wrong assignment to the support group, the **user doesn't belong to Avio Aero company**. MyTech HIDDEN_NAME (HIDDEN_ID). Attached files:
REASSIGNED: support group	wrong	0.049	CORP CTCR	Task Closed Skipped with close notes: Incident changed assignment group.
REASSIGNED: support group	wrong	11.402	AvioAero CAE/CAT	Task Closed Skipped with close notes: Wrong assignment to the support group, **the user doesn't belong to Avio Aero company. Please do not turn the ticket over again** (this is the second time). Thanks
REASSIGNED: support group	wrong	0.596	CORP CTCR	Task Closed Skipped with close notes: Incident changed assignment group. [...] Hi Dispatch team, **support group CAE/CAT Support has received the ticket twice and informed user is not part of Avio Aero business and therefore does not fall under their support**
CONCLUDED: resolved		46.8	Gas&Power	Task Closed Complete with close notes: **Issue resolved after user tried this solution 1**. Solution 1: From the Start menu, type %appdata% in the search field and press enter. That opens a window [...] [Yesterday 10:17 PM] HIDDEN_NAME (GE Aerospace). It works. [Yesterday 10:17 PM] HIDDEN_NAME (GE Gas Power, consultant). Okay thanks for the update. [Yesterday 10:17 PM] HIDDEN_NAME (GE Gas Power, consultant). May I close the Incident which is raised by you?. [Yesterday 10:17 PM] HIDDEN_NAME (GE Aerospace). Yes, thanks HIDDEN_NAME (HIDDEN_ID) [...]

Table 7.8: The top three variants, each accompanied by a list of their associated intents and the corresponding frequency of occurrence.

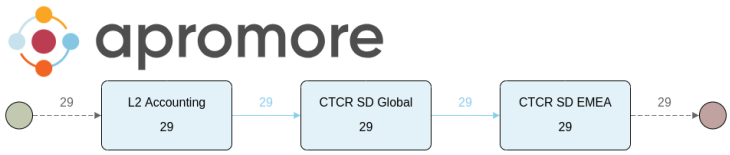
Variant	Intent	Count
1	User access issue: folder visibility	66
	password reset issue	15
	Access issue to shared file and print	3
	Access to AvioAero network	3
2	User access issue: folder visibility	20
	Access to AvioAero network	12
	Storage issue on network device	8
	file restoration issue	8
3	shopfloor tools incident	36

Finally, to address RP3, we extend the variant analysis by taking into consideration the intents extracted by the LLM. These intents enable the alignment of process variants with underlying patterns. By analyzing process variants through their intents, we observed that similar intents typically follow the same process flow. This finding underscores the capability of LLMs to extract actionable knowledge from unstructured data, streamlining the analysis of ticket lifecycles.

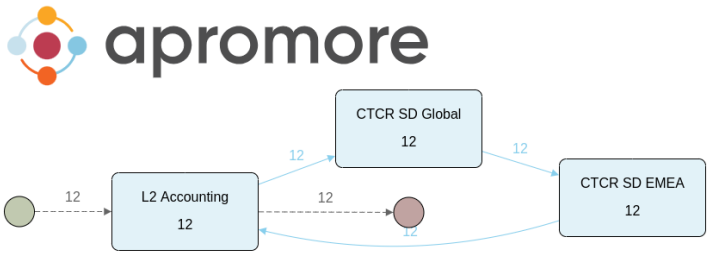
In a deeper analysis of process variants, we select the three most frequent ones and count their intents, as illustrated in Table 7.8. The table shows similar intents for variants 1 and 2: all of them are related to network or file access. This behavior persisted even after the clustering procedure, which points out a better tuning strategy to better group them, although we have already significantly reduced the number of unique intents (from around 800 to around 67, as discussed in previous sections). We found that cases with identical intents but different variants frequently involve an extra reassignment (e.g., 'User access issue: folder visibility'). This behavior points to potential opportunities for automation, reducing manual handling and improving efficiency, since the same outcome is achieved by different variants. Additionally, we analyzed the frequency of intent-support group pairs. The results provided insights into recurring patterns, allowing us to detect process workflows that can be standardized and tailored to common scenarios. Finally, variant 3 was successfully summarized by the LLM since only one single intent was assigned to it.

To support the claim of automation, we compare two closely related process variants presented in Figure 7.17. Variant 1, illustrated in Figure 7.17a, completes the process in three straightforward steps, progressing from start to finish despite some wrong reassignments being found for some cases. On the other hand, variant 2, shown in Figure 7.17b, requires four steps to reach completion.

The additional fourth step involves a reassignment of the task back to the 'L2 Accounting' support group, which serves no functional purpose and introduces an avoidable cycle into the process model. By eliminating this redundant behavior and adopting variant 1 only for all these related intents, the process could then achieve greater efficiency and simplicity.



(a) Variant 1.



(b) Varian 2.

Figure 7.17: The two most frequent variants derived from our event log.

7.6 Final Remarks

The introduced use cases developed a comprehensive methodology to detect and analyze unnecessary reassignments in support groups using PM and LLM techniques. We divided the our research question (RQ4) "*What are the advantages, disadvantages, and practical challenges of integrating process mining and deep learning in an industrial context at Avio Aero, and how can these technologies be effectively combined to address real-world process optimisation problems?*"

into four research problems. We focused on incidents and tasks logged in Avio Aero's organizational issue tracking system, where reassignments between support groups often represent a significant cost. The use cases, therefore, were designed to reduce the number of such reassignments.

In order to accomplish our goals, we first built an event log by extracting process data from the ServiceNow system. Using a traditional PM pipeline, we performed process discovery and analyzed key performance indicators (execution and await time) to understand the flow of reassignments and which support groups presented the worst KPI values. In typical PM settings, activity labels represent clear process steps, like 'order placement' or 'payment,' which facilitates process modeling and interpretation. However, in our case, support group names served as activity labels, complicating and limiting visualization and understanding to some extent. Nevertheless, this challenge is also common in the manufacturing sector, and in the future, we intend to apply our methodology in this sector at Avio Aero as well.

Through the PM analysis, we identified support groups that were heavily impacted by reassignments. To overcome the limitations regarding the activity labels of the initial PM analysis, we employed LLMs to enrich the event log by extracting the problem intent and the outcome of each event. Carefully designed prompts allowed the model to interpret trace data effectively, especially using techniques like one-shot and few-shot prompting. Validation of this enriched log was performed manually by company experts, although in the future a more robust validation approach is needed. This enriched event log could serve as a benchmark dataset for future studies aiming to automate event log validation in similar contexts.

Our analysis yielded actionable insights for stakeholders. We identified support groups that experienced high volumes of unnecessary reassignments, intents that were most frequently reassigned in error, and patterns of recurring intents associated with specific support groups. Additionally, we found trends in process variants indicating which intents took longer to resolve, providing the company with targeted areas for process improvement. This methodology not only advances process mining applications in complex, real-world settings but also demonstrates the value of integrating LLMs into PM for analyzing text-heavy data.

The methodology proposed in this work is an initial work. In the future, we intend to extend the methodology to the manufacturing sector at Avio Aero. This will be aligned with the resilient steps introduced in chapter 5 in order to guarantee sustainable research in a sector that will have very large amounts of data. Moreover, aligning the future directions with the reproducibility protocols proposed in chapter 6 will also guarantee consistent research to ensure the development of reliable products for Avio Aero.

Chapter 8

Conclusion

This thesis has explored critical aspects of PM, encoding methodologies, and the integration of advanced technologies such as deep learning and LLMs. Through a series of focused studies, we have shown how these techniques can address real-world challenges, emphasizing both methodological rigor and practical applicability. We defined 4 research questions and successfully answered them through systematic experiments.

In the area of process data representation, we discussed several potential directions for innovation. First, we reviewed several encoding techniques from different literature and answered **RQ1** by showing that there is not one best choice for all PM scenarios. We accomplished this answer by surveying several encoding techniques from other research domains and adapting them to the anomaly detection problem in PM. Our results showed that each encoding technique presents different performances on the downstream tasks depending on the types of anomalies, making the choice of the right technique challenging. In addition to process data encoding experiments, further findings showed that even simple feature engineering strategies can significantly improve predictive performance in the remaining time prediction task, providing an accessible alternative to the state-of-the-art deep learning models, which are often resource-intensive and difficult to interpret. We hope these insights will motivate and guide researchers to develop tailored, efficient encoding methods that balance computational cost with accuracy for process data.

In the domain of process simulation, we introduced a novel contribution that integrates constraints and prior knowledge, moving beyond traditional generative models proposed in PM to simulate processes or generate synthetic event logs. This innovation filled the gap of DL-based simulation not having the flexibility to perform what-if analysis or user-constrained simulation. Such contribution successfully answered our **RQ2**, showing that deep learning can improve traditional, data-driven process simulation approaches. Such data-driven approaches have

several advantages and unique strengths, for instance, more flexibility regarding resource allocation. Nevertheless, we advanced the field by proposing this innovative idea of simulating new process instances based on user-defined constraints, which, to the best of our knowledge, has been only achieved through our learned-based solution. By incorporating conformance checking and leveraging temporal logical rules, we enhanced the practical usage of simulations, enabling practitioners to perform more nuanced and alternative analyses. We expect this contribution to promote circular process reengineering while enhancing the resilience and safety of process management. By enabling stakeholders to simulate workflow changes based on custom constraints without taking risks or costs, it offers a practical and secure path to process optimization. However, this research also highlighted areas for improvement, particularly in preprocessing and feature selection, which remain open challenges for future work.

We also addressed the unexplored issue of sustainability in process mining applications based on AI. By introducing resource consumption as a critical metric alongside traditional performance metrics, this work highlighted the importance of developing environmentally conscious AI solutions. As a result, we successfully answered **RQ3** and showed that it is possible to develop sustainable and resilient AI-based solutions in PM. Several experiments in different tasks were carried out to compare the current state-of-the-art with lightweight pipelines proposed in this work. In a nutshell, in terms of predictive accuracy, our proposals outperformed or performed similarly to methods from the literature, whereas the resource consumption metrics were drastically improved. The main contribution presented in this regard consists of insights showing how to correctly select or find a suitable encoding technique for a task and improving traditional (and computationally costly) learning pipelines. Furthermore, we introduced a new trace modeling approach that not only optimizes the training of deep learning models but also demonstrates that sustainable methodologies can be achieved through the careful use of appropriate techniques existing in the literature.

Looking ahead, the research community should focus on clear studies and avoid the overreliance on empirical experiments. Work in PPM must use careful experiments with consistent evaluation methods to show the strengths and limits of each approach. For instance, research should compare different techniques with the same criteria regarding the end-to-end pipeline of data science so that practitioners can understand which methods work best in various situations. It is important to study how changes in process data affect outcomes while keeping an eye on resource use as well. Overall, future studies should be simple to apply and verify, ensuring that new methods can work well in real-life settings.

Lastly, our collaboration with Avio Aero effectively addressed the last research question **RQ4**, by showcasing the practical integration of PM and LLM technologies. We elaborated 3 different use cases using PM only, LLM only, and

combining both. The use cases successfully solved the four research problems defined at Avio Aero along with the experts from the company. By analyzing inefficient support group reassignments, we demonstrated how LLMs could enrich event logs with contextual insights, improving both process understanding and stakeholder decision-making. This solution reduced the number of transactions between events, an action that is often costly for the company. This application represents a step forward in bridging traditional process mining with modern text analysis techniques, opening new possibilities for industries with text-heavy data.

As this work concludes, the overarching theme emerges: the need for balanced, adaptable solutions that address both the technical and practical challenges of process mining. Future research should build upon these foundations, expanding methodological rigor while maintaining a focus on sustainable, impactful applications. By doing so, we can ensure the continued advancement of process mining as a field that not only drives innovation but also creates value for industries and society at large.

Bibliography

[1] Jan Niklas Adams, Gyunam Park, and Wil M. P. van der Aalst. Preserving complex object-centric graph structures to improve machine learning tasks in process mining. *Eng. Appl. Artif. Intell.*, 125:106764, 2023. doi: 10.1016/J.ENGAPPAI.2023.106764. URL `https://doi.org/10.1016/j.engappai.2023.106764`.

[2] Nesreen K. Ahmed, Ryan A. Rossi, John Boaz Lee, Theodore L. Willke, Rong Zhou, Xiangnan Kong, and Hoda Eldardiry. Role-based graph embeddings. *IEEE Transactions on Knowledge and Data Engineering*, 34(5): 2401–2415, 2022. doi: 10.1109/TKDE.2020.3006475.

[3] Edesio Alcobaca, Felipe Siqueira, Adriano Rivolli, Luis Paulo F. Garcia, Jefferson Tales Oliva, and Andre C. P. L. F. de Carvalho. MFE: towards reproducible meta-feature extraction. *J. Mach. Learn. Res.*, 2020.

[4] Anti Alman, Ivan Donadello, Fabrizio Maria Maggi, and Marco Montali. Declarative process mining for software processes: The rum toolkit and the declare4py python library. In *Product-Focused Software Process Improvement*, pages 13–19, 2024.

[5] Ethem Alpaydin. *Machine Learning: The New AI*. MIT Press, 2016. ISBN 9780262337588. URL `https://mitpress.mit.edu/books/machine-learning-1`.

[6] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271, 2018. URL `http://arxiv.org/abs/1803.01271`.

[7] Sylvio Barbon Junior, Paolo Ceravolo, Ernesto Damiani, and Gabriel Marques Tavares. Evaluating trace encoding methods in process mining. In *International Symposium: From Data to Models and Back*, pages 174–189. Springer, 2020.

[8] H. B. Barlow. Unsupervised learning. *Neural Comput.*, 1(3):295–311, 1989. doi: 10.1162/NECO.1989.1.3.295. URL https://doi.org/10.1162/neco.1989.1.3.295.

[9] Daniel Barón-Espitia, Marlon Dumas, and Oscar González Rojas. Automated generation of process simulation scenarios from declarative control-flow changes. *PeerJ Comput. Sci.*, 10:e2094, 2024. doi: 10.7717/PEERJ-CS.2094. URL https://doi.org/10.7717/peerj-cs.2094.

[10] Iris Beerepoot, Claudio Di Ciccio, Hajo A. Reijers, Stefanie Rinderle-Ma, Wasana Bandara, Andrea Burattin, Diego Calvanese, Tianwa Chen, Izack Cohen, Benoît Depaire, Gemma Di Federico, Marlon Dumas, Christopher G. J. van Dun, Tobias Fehrer, Dominik Andreas Fischer, Avigdor Gal, Marta Indulska, Vatche Isahagian, Christopher Klinkmüller, Wolfgang Kratsch, Henrik Leopold, Amy Van Looy, Hugo A. López, Sanja Lukumbuzya, Jan Mendling, Lara Meyers, Linda Moder, Marco Montali, Vinod Muthusamy, Manfred Reichert, Yara Rizk, Michael Rosemann, Maximilian Röglinger, Shazia Sadiq, Ronny Seiger, Tijs Slaats, Mantas Simkus, Ida Asadi Someh, Barbara Weber, Ingo Weber, Mathias Weske, and Francesca Zerbato. The biggest business process management problems to solve before we die. *Comput. Ind.*, 146:103837, 2023. doi: 10.1016/J.COMPIND.2022.103837. URL https://doi.org/10.1016/j.compind.2022.103837.

[11] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *International Conference on Neural Information Processing Systems: Natural and Synthetic (NIPS'01)*, NIPS'01, page 585–591, Cambridge, MA, USA, 2001. MIT Press.

[12] Alessandro Berti and Wil MP van der Aalst. Reviving token-based replay: Increasing speed while improving diagnostics. In *ATAED@ Petri Nets/ACSD*, pages 87–103, 2019.

[13] Alessandro Berti, Sebastiaan van Zelst, and Daniel Schuster. Pm4py: A process mining library for python. *Software Impacts*, 17:100556, 2023. ISSN 2665-9638.

[14] Alessandro Berti, Johannes Herforth, Mahnaz Sadat Qafari, and Wil M. P. van der Aalst. Graph-based feature extraction on object-centric event logs. *Int. J. Data Sci. Anal.*, 18(2):139–155, 2024. doi: 10.1007/S41060-023-00428-2. URL https://doi.org/10.1007/s41060-023-00428-2.

[15] Alessandro Berti, Humam Kourani, Hannes Häfke, Chiao-Yun Li, and Daniel Schuster. Evaluating large language models in process mining: Capabilities, benchmarks, and evaluation strategies. In Han van der Aa, Dominik Bork, Rainer Schmidt, and Arnon Sturm, editors, *Enterprise, Business-Process and Information Systems Modeling*, pages 13–21, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-61007-3.

[16] Fábio Bezerra and Jacques Wainer. Algorithms for anomaly detection of traces in logs of process aware information systems. *Information Systems*, 38(1):33 – 44, 2013. ISSN 0306-4379.

[17] Christopher M. Bishop. *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer, 2007.

[18] José Brás, Rúben Pereira, and Sérgio Moro. Intelligent process automation and business continuity: Areas for future research. *Inf.*, 14(2): 122, 2023. doi: 10.3390/info14020122. URL `https://doi.org/10.3390/info14020122`.

[19] Lars Brehm, Jessica Slamka, and Andreas Nickmann. *Process Mining for Carbon Accounting: An Analysis of Requirements and Potentials*, pages 209–244. Springer International Publishing, Cham, 2022. ISBN 978-3-031-06543-9. doi: 10.1007/978-3-031-06543-9_9. URL `https://doi.org/10.1007/978-3-031-06543-9_9`.

[20] M. Brenner. Classifying itil processes; a taxonomy under tool support aspects. In *2006 IEEE/IFIP Business Driven IT Management*, pages 19–28, 2006. doi: 10.1109/BDIM.2006.1649207.

[21] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

[22] Andrea Burattin. PLG2: multiperspective processes randomization and simulation for online and offline settings. *CoRR*, abs/1506.08415, 2015.

[23] Andrea Burattin. Plg2: Multiperspective processes randomization and simulation for online and offline settings, 2015.

[24] Andrea Burattin, Barbara Re, Lorenzo Rossi, and Francesco Tiezzi. A purpose-guided log generation framework. In *BPM*, volume 13420, pages 181–198. Springer, 2022.

[25] Manuel Camargo, Marlon Dumas, and Oscar González Rojas. Simod: A tool for automated discovery of business process simulation models. In *BPM Workshop*, volume 2420, pages 139–143, 2019.

[26] Manuel Camargo, Marlon Dumas, and Oscar González Rojas. Learning accurate LSTM models of business processes. In *BPM*, volume 11675, pages 286–302. Springer, 2019.

[27] Manuel Camargo, Marlon Dumas, and Oscar González. Automated discovery of business process simulation models from event logs. *Dec. Sup. Sys*, 134:113284, 2020.

[28] Manuel Camargo, Marlon Dumas, and Oscar González Rojas. Discovering generative models from event logs: data-driven simulation vs deep learning. *PeerJ CS*, 7:e577, 2021.

[29] Manuel Camargo, Marlon Dumas, and Oscar González-Rojas. Learning accurate business process simulation models from event logs via automated process discovery and deep learning. In *International Conference on Advanced Information Systems Engineering*, pages 55–71. Springer, 2022.

[30] Manuel Camargo, Marlon Dumas, and Oscar González Rojas. Learning accurate business process simulation models from event logs via automated process discovery and deep learning. In *CAiSE*, pages 55–71. Springer, 2022.

[31] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *International on Conference on Information and Knowledge Management (CIKM)*, CIKM '15, page 891–900, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450337946. doi: 10.1145/2806416.2806512.

[32] Josep Carmona, Boudewijn F. van Dongen, Andreas Solti, and Matthias Weidlich. *Conformance Checking - Relating Processes and Models*. Springer, 2018. ISBN 978-3-319-99413-0. doi: 10.1007/978-3-319-99414-7.

[33] Paolo Ceravolo, Ernesto Damiani, Mohammadsadegh Torabi, and Sylvio Barbon Junior. Toward a new generation of log pre-processing methods for process mining. In Josep Carmona, Gregor Engels, and Akhil Kumar, editors, *Business Process Management Forum (BPM)*, volume 297 of

Lecture Notes in Business Information Processing, pages 55–70. Springer, 2017. doi: 10.1007/978-3-319-65015-9_4.

[34] Paolo Ceravolo, Gabriel Marques Tavares, Sylvio Barbon Junior, and Ernesto Damiani. Evaluation goals for online process mining: A concept drift perspective. *IEEE Trans. Serv. Comput.*, 2022.

[35] Paolo Ceravolo, Sylvio Barbon, Ernesto Damiani, and Wil Van der Aalst. Tuning machine learning to address process mining requirements. *IEEE Access*, 2024.

[36] Paolo Ceravolo, Marco Comuzzi, Jochen De Weerdt, Chiara Di Francescomarino, and Fabrizio Maria Maggi. Predictive process monitoring: concepts, challenges, and future research directions. *Process Science*, 1(1): 1–22, 2024.

[37] David Chapela-Campa, Ismail Benchekroun, Opher Baron, Marlon Dumas, Dmitry Krass, and Arik Senderovich. A framework for measuring the quality of business process simulation models. *Inf. Syst.*, 127:102447, 2025. doi: 10.1016/J.IS.2024.102447. URL `https://doi.org/10.1016/j.is.2024.102447`.

[38] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL, 2014. doi: 10.3115/V1/D14-1179. URL `https://doi.org/10.3115/v1/d14-1179`.

[39] Carl Corea, Paolo Felli, Marco Montali, and Fabio Patrizi. On the flexibility of declarative process specifications. In Giancarlo Guizzardi, Flávia Maria Santoro, Haralambos Mouratidis, and Pnina Soffer, editors, *Advanced Information Systems Engineering - 36th International Conference, CAiSE 2024, Limassol, Cyprus, June 3-7, 2024, Proceedings*, volume 14663 of *Lecture Notes in Computer Science*, pages 161–177. Springer, 2024. doi: 10.1007/978-3-031-61057-8_10. URL `https://doi.org/10.1007/978-3-031-61057-8_10`.

[40] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[41] Claudio Aparecido Lira do Amaral, Marcelo Fantinato, Hajo A. Reijers, and Sarajane Marques Peres. Enhancing completion time prediction through attribute selection. In Ewa Ziemba, editor, *Information Technology for Management: Emerging Research and Applications - 15th Conference, AITM 2018, and 13th Conference, ISM 2018, Held as Part of FedCSIS, Poznan, Poland, September 9-12, 2018, Revised and Extended Selected Papers*, volume 346 of *Lecture Notes in Business Information Processing*, pages 3–23. Springer, 2018. doi: 10.1007/978-3-030-15154-6_1. URL `https://doi.org/10.1007/978-3-030-15154-6_1`.

[42] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2070–2079. IEEE Computer Society, 2017. doi: 10.1109/ICCV.2017.226. URL `https://doi.org/10.1109/ICCV.2017.226`.

[43] Ivan Donadello, Chiara Di Francescomarino, Fabrizio Maria Maggi, Francesco Ricci, and Aladdin Shikhizada. Outcome-oriented prescriptive process monitoring based on temporal logic patterns. *EAAI journal*, 126, 2023. ISSN 0952-1976.

[44] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. Learning structural node embeddings via diffusion wavelets. In *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, KDD '18, page 1320–1329, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520. doi: 10.1145/3219819.3220025.

[45] Cleiton dos Santos Garcia, Alex Meincheim, Elio Ribeiro Faria Junior, Marcelo Rosano Dallagassa, Denise Maria Vecino Sato, Deborah Ribeiro Carvalho, Eduardo Alves Portela Santos, and Edson Emilio Scalabrin. Process mining techniques and applications – a systematic mapping study. *Expert Systems with Applications*, 133:260–295, 2019. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2019.05.003.

[46] Cleiton dos Santos Garcia, Alex Meincheim, Elio Ribeiro Faria Junior, Marcelo Rosano Dallagassa, Denise Maria Vecino Sato, Deborah Ribeiro Carvalho, Eduardo Alves Portela Santos, and Edson Emílio Scalabrin. Process mining techniques and applications - A systematic mapping study. *Expert Syst. Appl.*, 133:260–295, 2019.

[47] Marlon Dumas. Constructing digital twins for accurate and reliable what-if business process analysis. In *BPM Workshop*, volume 2938, pages 23–27, 2021.

[48] Marlon Dumas, L Marcello Rosa, Jan Mendling, and A Hajo Reijers. *Fundamentals of business process management*. Springer, 2018.

[49] John B. Edwards and Guy H. Orcutt. Should aggregation prior to estimation be the rule? *The Review of Economics and Statistics*, 51(4):409–420, 1969. doi: 10.2307/1926432. URL https://doi.org/10.2307/1926432. Accessed: 25 Feb. 2025.

[50] Mathias Eggert and Julian Dyong. Applying process mining in small and medium sized IT enterprises - challenges and guidelines. In Claudio Di Ciccio, Remco M. Dijkman, Adela del-Río-Ortega, and Stefanie Rinderle-Ma, editors, *Business Process Management - 20th International Conference, BPM 2022, Münster, Germany, September 11-16, 2022, Proceedings*, volume 13420 of *Lecture Notes in Computer Science*, pages 125–142. Springer, 2022. doi: 10.1007/978-3-031-16103-2_11. URL https://doi.org/10.1007/978-3-031-16103-2_11.

[51] Bedilia Estrada-Torres, Manuel Camargo, Marlon Dumas, Luciano García-Bañuelos, Ibrahim Mahdy, and Maksym Yerokhin. Discovering business process simulation models in the presence of multitasking and availability constraints. *Data Knowl. Eng.*, 134:101897, 2021.

[52] Joerg Evermann, Jana-Rebecca Rehse, and Peter Fettke. Predicting process behaviour using deep learning. *Decis. Support Syst.*, 100:129–140, 2017.

[53] Dirk Fahland. Process mining over multiple behavioral dimensions with event knowledge graphs. In Wil M. P. van der Aalst and Josep Carmona, editors, *Process Mining Handbook*, volume 448 of *Lecture Notes in Business Information Processing*, pages 274–319. Springer, 2022. doi: 10.1007/978-3-031-08848-3_9. URL https://doi.org/10.1007/978-3-031-08848-3_9.

[54] Chiara Di Francescomarino, Chiara Ghidini, Fabrizio Maria Maggi, Giulio Petrucci, and Anton Yeshchenko. An eye into the future: Leveraging a-priori knowledge in predictive business process monitoring. In *BPM*, volume 10445, pages 252–268. Springer, 2017.

[55] Chiara Di Francescomarino, Marlon Dumas, Fabrizio Maria Maggi, and Irene Teinemaa. Clustering-based predictive process monitoring. *IEEE Trans. Serv. Comput.*, 12(6):896–909, 2019. doi: 10.1109/TSC.2016.2645153. URL https://doi.org/10.1109/TSC.2016.2645153.

[56] Chiara Di Francescomarino, Ivan Donadello, Chiara Ghidini, Fabrizio Maria Maggi, and Williams Rizzi. Making sense of temporal data: the

DECLARE encoding. In Giuseppe De Giacomo, Antonella Guzzo, Marco Montali, Lior Limonad, Fabiana Fournier, and Tagatha Chakraborti, editors, *Proceedings of the Workshop on Process Management in the AI Era (PMAI 2022) co-located with 31st International Joint Conference on Artificial Intelligence and the 25th European Conference on Artificial Intelligence (IJCAI-ECAI 2022), Wien, Austria, July 23, 2022*, volume 3310 of *CEUR Workshop Proceedings*, pages 77–80. CEUR-WS.org, 2022. URL `https://ceur-ws.org/Vol-3310/paper9.pdf`.

[57] Riccardo Galanti, Massimiliano de Leoni, Merylin Monaro, Nicolò Navarin, Alan Marazzi, Brigida Di Stasi, and Stéphanie Maldera. An explainable decision support system for predictive process analytics. *Eng. Appl. Artif. Intell.*, 120:105904, 2023. doi: 10.1016/J.ENGAPPAI.2023.105904. URL `https://doi.org/10.1016/j.engappai.2023.105904`.

[58] Andrea Gasparetto, Matteo Marcuzzo, Alessandro Zangari, and Andrea Albarelli. A survey on text classification algorithms: From text to predictions. *Information*, 13(2):83, 2022. doi: 10.3390/info13020083.

[59] Anahita Farhang Ghahfarokhi, Gyunam Park, Alessandro Berti, and Wil M. P. van der Aalst. OCEL: A standard for object-centric event logs. In Ladjel Bellatreche, Marlon Dumas, Panagiotis Karras, Raimundas Matulevicius, Ahmed Awad, Matthias Weidlich, Mirjana Ivanovic, and Olaf Hartig, editors, *New Trends in Database and Information Systems - ADBIS 2021 Short Papers, Doctoral Consortium and Workshops: DOING, SIMPDA, MADEISD, MegaData, CAoNS, Tartu, Estonia, August 24-26, 2021, Proceedings*, volume 1450 of *Communications in Computer and Information Science*, pages 169–175. Springer, 2021. doi: 10.1007/978-3-030-85082-1_16. URL `https://doi.org/10.1007/978-3-030-85082-1_16`.

[60] Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, pages 854–860. IJCAI/AAAI, 2013.

[61] Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016. doi: 10.1613/jair.4992.

[62] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018. doi: 10.1016/j.knosys.2018.03.022.

[63] Nina Graves, István Koren, and Wil M. P. van der Aalst. Rethink your processes! A review of process mining for sustainability. In *International Conference on ICT for Sustainability, ICT4S 2023, Rennes, France, June 5-9, 2023*, pages 164–175. IEEE, 2023. doi: 10.1109/ICT4S58814.2023.00025. URL https://doi.org/10.1109/ICT4S58814.2023.00025.

[64] Riccardo Graziosi, Massimiliano Ronzani, Andrei Buliga, Chiara Di Francescomarino, Francesco Folino, Chiara Ghidini, Francesca Meneghello, and Luigi Pontieri. Generating the traces you need: A conditional generative model for process mining data. In *6th International Conference on Process Mining, ICPM 2024, Kgs. Lyngby, Denmark, October 14-18, 2024*, pages 25–32. IEEE, 2024. doi: 10.1109/ICPM63005.2024.10680621. URL https://doi.org/10.1109/ICPM63005.2024.10680621.

[65] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *NIPS*, 2022.

[66] Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, KDD '16, page 855–864, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939754.

[67] Monika Gupta, Allahbaksh M. Asadullah, Srinivas Padmanabhuni, and Alexander Serebrenik. Reducing user input requests to improve IT support ticket resolution process. *Empir. Softw. Eng.*, 23(3):1664–1703, 2018. doi: 10.1007/s10664-017-9532-2.

[68] Marwan Hassani, Pascal Spaus, Mohamed Medhat Gaber, and Thomas Seidl. Density-based projected clustering of data streams. In Eyke Hüllermeier, Sebastian Link, Thomas Fober, and Bernhard Seeger, editors, *Scalable Uncertainty Management - 6th International Conference, SUM 2012, Marburg, Germany, September 17-19, 2012. Proceedings*, volume 7520 of *Lecture Notes in Computer Science*, pages 311–324. Springer, 2012. doi: 10.1007/978-3-642-33362-0_24. URL https://doi.org/10.1007/978-3-642-33362-0_24.

[69] Amr Hendy, M. Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, M. Afify, and Hany Hassan Awadalla. How good are gpt models at machine translation? a comprehensive evaluation. *ArXiv*, abs/2302.09210, 2023. doi: 10.48550/arXiv.2302.09210.

[70] Moritz Herrmann, F. Julian D. Lange, Katharina Eggensperger, Giuseppe Casalicchio, Marcel Wever, Matthias Feurer, David Rügamer, Eyke Hüllermeier, Anne-Laure Boulesteix, and Bernd Bischl. Position: Why we must rethink empirical research in machine learning. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=DprrMz24tk.

[71] Markku Hinkka, Teemu Lehto, and Keijo Heljanko. Exploiting event log event attributes in RNN based prediction. In *SIMPDA*, volume 379, pages 67–85. Springer, 2019.

[72] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

[73] Jon Iden and Tom Roar Eikebrokk. Implementing IT service management: A systematic literature review. *Int. J. Inf. Manag.*, 33(3):512–523, 2013. doi: 10.1016/J.IJINFOMGT.2013.01.004. URL https://doi.org/10.1016/j.ijinfomgt.2013.01.004.

[74] Adrian Joas, Maren Gierlich-Joas, Charlotte Bahr, and Janina Bauer. Towards leveraging process mining for sustainability – an analysis of challenges and potential solutions. In Andrea Marrella, Manuel Resinas, Mieke Jans, and Michael Rosemann, editors, *Business Process Management Forum*, pages 354–371, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-70418-5.

[75] Nicola Jones. How to stop data centres from gobbling up the world's electricity. *Nature*, 561:163–166, 2018. URL https://api.semanticscholar.org/CorpusID:205128890.

[76] Sylvio Barbon Junior, Paolo Ceravolo, Ernesto Damiani, and Gabriel Marques Tavares. Evaluating trace encoding methods in process mining. In Juliana Bowles, Giovanna Broccia, and Mirco Nanni, editors, *From Data to Models and Back - 9th International Symposium, DataMod 2020, Virtual Event, October 20, 2020, Revised Selected Papers*, volume 12611 of *Lecture Notes in Computer Science*, pages 174–189. Springer, 2020. doi: 10.1007/978-3-030-70650-0_11. URL https://doi.org/10.1007/978-3-030-70650-0_11.

[77] Lynn H. Kaack, Priya L. Donti, Emma Strubell, George Kamiya, Felix Creutzig, and David Rolnick. Aligning artificial intelligence with climate change mitigation. *Nature Climate Change*, 12(6):518–527, June 2022.

doi: 10.1038/s41558-022-01377-. URL https://ideas.repec.org/a/nat/natcli/v12y2022i6d10.1038_s41558-022-01377-7.html.

[78] István Ketykó, Felix Mannhardt, Marwan Hassani, and Boudewijn F. van Dongen. What averages do not tell: predicting real life processes with sequential deep learning. In Jiman Hong, Miroslav Bures, Juw Won Park, and Tomás Cerný, editors, *SAC '22: The 37th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, April 25 - 29, 2022*, pages 1128–1131. ACM, 2022. doi: 10.1145/3477314.3507179. URL https://doi.org/10.1145/3477314.3507179.

[79] Jongchan Kim, Marco Comuzzi, Marlon Dumas, Fabrizio Maria Maggi, and Irene Teinemaa. Encoding resource experience for predictive process monitoring. *Decis. Support Syst.*, 153:113669, 2022. doi: 10.1016/J.DSS.2021.113669. URL https://doi.org/10.1016/j.dss.2021.113669.

[80] S. Kim, M. Comuzzi, and C. Di Francescomarino. Explaining the impact of design choices on model quality in predictive process monitoring. *Journal of Intelligent Information Systems*, 2024. doi: 10.1007/s10844-024-00903-7.

[81] Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3581–3589, 2014. URL https://proceedings.neurips.cc/paper/2014/hash/d523773c6b194f37b938d340d5d02232-Abstract.html.

[82] Lukas Kirchdorfer, Robert Blümel, Timotheus Kampik, Han van der Aa, and Heiner Stuckenschmidt. Agentsimulator: An agent-based approach for data-driven business process simulation. In *6th International Conference on Process Mining, ICPM 2024, Kgs. Lyngby, Denmark, October 14-18, 2024*, pages 97–104. IEEE, 2024. doi: 10.1109/ICPM63005.2024.10680660. URL https://doi.org/10.1109/ICPM63005.2024.10680660.

[83] Pieter De Koninck, Seppe vanden Broucke, and Jochen De Weerdt. act2vec, trace2vec, log2vec, and model2vec: Representation learning for business processes. In Mathias Weske, Marco Montali, Ingo Weber, and Jan vom Brocke, editors, *Business Process Management (BPM)*, volume 11080 of *Lecture Notes in Computer Science*, pages 305–321. Springer, 2018. doi: 10.1007/978-3-319-98648-7_18.

[84] Satyapriya Krishna, Tessa Han, Alex Gu, Javin Pombra, Shahin Jabbari, Steven Wu, and Himabindu Lakkaraju. The disagreement problem in explainable machine learning: A practitioner's perspective. *CoRR*, 2022.

[85] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012. URL https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

[86] Sebastian Kroeger, Lasse Streibel, Patrick Jordan, Bjoern Klages, Christoph Soellner, and Michael F. Zaeh. Sustainability assessment of production networks using simulation-data-based process mining. *Procedia Computer Science*, 237:493–501, 2024. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2024.05.132. URL https://www.sciencedirect.com/science/article/pii/S1877050924011487. International Conference on Industry Sciences and Computer Science Innovation.

[87] Esther Maria Rojas Krugger, Ana Rocío Cárdenas Maita, Juliana Cristina Barbosa Alves, Marcelo Fantinato, and Sarajane Marques Peres. Business process analysis based on anomaly detection in event logs: a study on an incident management case. In *54th Hawaii International Conference on System Sciences, HICSS 2021, Kauai, Hawaii, USA, January 5, 2021*, pages 1–10. ScholarSpace, 2021. URL https://hdl.handle.net/10125/70742.

[88] Selim Larsch, Stefanie Betz, Leticia Duboc, Andréa Magalhães Magdaleno, and Camilla Bomfim. Integrating sustainability aspects in business process management. In Marlon Dumas and Marcelo Fantinato, editors, *Business Process Management Workshops - BPM 2016 International Workshops, Rio de Janeiro, Brazil, September 19, 2016, Revised Papers*, volume 281 of *Lecture Notes in Business Information Processing*, pages 403–415, 2016. doi: 10.1007/978-3-319-58457-7_29. URL https://doi.org/10.1007/978-3-319-58457-7_29.

[89] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–1188–II–1196. JMLR.org, 2014.

[90] Anna Leontjeva, Raffaele Conforti, Chiara Di Francescomarino, Marlon Dumas, and Fabrizio Maria Maggi. Complex symbolic sequence encodings for predictive monitoring of business processes. In Hamid Reza Motahari-Nezhad, Jan Recker, and Matthias Weidlich, editors, *Business Process Management - 13th International Conference, BPM 2015, Innsbruck, Austria, August 31 - September 3, 2015, Proceedings*, volume 9253 of *Lecture Notes in Computer Science*, pages 297–313. Springer, 2015. doi: 10.1007/978-3-319-23063-4_21. URL https://doi.org/10.1007/978-3-319-23063-4_21.

[91] Jundong Li, Liang Wu, Ruocheng Guo, Chenghao Liu, and Huan Liu. Multi-level network embedding with boosted low-rank matrix approximation. In Francesca Spezzano, Wei Chen, and Xiaokui Xiao, editors, *International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 49–56. ACM, 2019. doi: 10.1145/3341161.3342864.

[92] Hezheng Lin, Xing Cheng, Xiangyu Wu, and Dong Shen. CAT: cross attention in vision transformer. In *IEEE International Conference on Multimedia and Expo, ICME 2022, Taipei, Taiwan, July 18-22, 2022*, pages 1–6. IEEE, 2022. doi: 10.1109/ICME52920.2022.9859720. URL https://doi.org/10.1109/ICME52920.2022.9859720.

[93] Li Lin, Lijie Wen, and Jianmin Wang. Mm-pred: A deep predictive model for multi-attribute event sequence. In Tanya Y. Berger-Wolf and Nitesh V. Chawla, editors, *Proceedings of the 2019 SIAM International Conference on Data Mining, SDM 2019, Calgary, Alberta, Canada, May 2-4, 2019*, pages 118–126. SIAM, 2019. doi: 10.1137/1.9781611975673.14. URL https://doi.org/10.1137/1.9781611975673.14.

[94] Yang Liu, Sujay Khandagale, Colin White, and Willie Neiswanger. Synthetic benchmarks for scientific research in explainable machine learning. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *NIPS*, 2021.

[95] Orlenys López-Pintado, Serhii Murashko, and Marlon Dumas. Discovery and simulation of data-aware business processes. *CoRR*, abs/2408.13666, 2024. doi: 10.48550/ARXIV.2408.13666. URL https://doi.org/10.48550/arXiv.2408.13666.

[96] Ana C Lorena, Luís PF Garcia, Jens Lehmann, Marcilio CP Souto, and Tin Kam Ho. How complex is your classification problem? a survey on measuring classification complexity. *ACM Computing Surveys (CSUR)*, 52 (5):1–34, 2019.

[97] Stefan Luettgen, Alexander Seeliger, Timo Nolle, and Max Mühlhäuser. Case2vec: Advances in representation learning for business processes. In Sander J. J. Leemans and Henrik Leopold, editors, *Process Mining Workshops - ICPM 2020 International Workshops, Padua, Italy, October 5-8, 2020, Revised Selected Papers*, volume 406 of *Lecture Notes in Business Information Processing*, pages 162–174. Springer, 2020. doi: 10.1007/978-3-030-72693-5_13. URL https://doi.org/10.1007/978-3-030-72693-5_13.

[98] H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165, 1958. doi: 10.1147/rd.22.0159.

[99] Andrea Maldonado, Christian M. M. Frey, Gabriel Marques Tavares, Nikolina Rehwald, and Thomas Seidl. GEDI: generating event data with intentional features for benchmarking process mining. In Andrea Marrella, Manuel Resinas, Mieke Jans, and Michael Rosemann, editors, *Business Process Management - 22nd International Conference, BPM 2024, Krakow, Poland, September 1-6, 2024, Proceedings*, volume 14940 of *Lecture Notes in Computer Science*, pages 221–237. Springer, 2024. doi: 10.1007/978-3-031-70396-6_13. URL https://doi.org/10.1007/978-3-031-70396-6_13.

[100] Mauricio Marrone and Lutz M. Kolbe. Impact of IT service management frameworks on the IT organization - an empirical study on benefits, challenges, and processes. *Bus. Inf. Syst. Eng.*, 3(1):5–18, 2011. doi: 10.1007/s12599-010-0141-5. URL https://doi.org/10.1007/s12599-010-0141-5.

[101] Niels Martin, Benoît Depaire, and An Caris. The use of process mining in business process simulation model construction - structuring the field. *Bus. Inf. Syst. Eng.*, 58(1):73–87, 2016.

[102] Nicola Di Mauro, Annalisa Appice, and Teresa M. A. Basile. Activity prediction of business process instances with inception CNN models. In *AI*IA*, volume 11946, pages 348–361. Springer, 2019.

[103] Francesca Meneghello, Chiara Di Francescomarino, and Chiara Ghidini. Runtime integration of machine learning and simulation for business processes. In *ICPM*, pages 9–16, 2023.

[104] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations (ICLR)*, 2013.

[105] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Neural Information Processing Systems (NIPS)*, pages 3111–3119, 2013.

[106] Nicolò Navarin, Beatrice Vincenzi, Mirko Polato, and Alessandro Sperduti. LSTM networks for data-aware remaining time prediction of business process instances. In *2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017, Honolulu, HI, USA, November 27 - Dec. 1, 2017*, pages 1–7. IEEE, 2017. doi: 10.1109/SSCI.2017.8285184. URL `https://doi.org/10.1109/SSCI.2017.8285184`.

[107] Mark Newman. Power laws, pareto distributions and zipfs law. *Contemporary Physics*, 46(5):323–351, 2005. doi: 10.1080/00107510500052444.

[108] Timo Nolle, Stefan Luettgen, Alexander Seeliger, and Max Mühlhäuser. Binet: Multi-perspective business process anomaly classification. *Information Systems*, page 101458, 2019. ISSN 0306-4379.

[109] Elena Orta and Mercedes Ruiz. Met4itil: A process management and simulation-based method for implementing ITIL. *Comput. Stand. Interfaces*, 61:1–19, 2019. doi: 10.1016/J.CSI.2018.01.006. URL `https://doi.org/10.1016/j.csi.2018.01.006`.

[110] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1105–1114. ACM, 2016. doi: 10.1145/2939672.2939751.

[111] Rafael Seidi Oyamada, Gabriel Marques Tavares, Sylvio Barbon Junior, and Paolo Ceravolo. Enhancing predictive process monitoring with time-related feature engineering. In Giancarlo Guizzardi, Flávia Maria Santoro, Haralambos Mouratidis, and Pnina Soffer, editors, *Advanced Information Systems Engineering - 36th International Conference, CAiSE 2024, Limassol, Cyprus, June 3-7, 2024, Proceedings*, volume 14663 of *Lecture Notes in Computer Science*, pages 71–86. Springer, 2024. doi: 10.1007/978-3-031-61057-8_5. URL `https://doi.org/10.1007/978-3-031-61057-8_5`.

[112] Rafael Seidi Oyamada, Gabriel Marques Tavares, Sylvio Barbon Junior, and Paolo Ceravolo. Cosmo: A framework to instantiate conditioned process simulation models. In Andrea Marrella, Manuel Resinas, Mieke Jans, and Michael Rosemann, editors, *Business Process Management - 22nd International Conference, BPM 2024, Krakow, Poland, September 1-6, 2024, Proceedings*, volume 14940 of *Lecture Notes in Computer Science*, pages 328–344. Springer, 2024. doi: 10.1007/978-3-031-70396-6_19. URL https://doi.org/10.1007/978-3-031-70396-6_19.

[113] Vincenzo Pasquadibisceglie, Annalisa Appice, Giovanna Castellano, and Donato Malerba. Using convolutional neural networks for predictive process analytics. In *ICPM*, pages 129–136. IEEE, 2019.

[114] Vincenzo Pasquadibisceglie, Raffaele Scaringi, Annalisa Appice, Giovanna Castellano, and Donato Malerba. Prophet: Explainable predictive process monitoring with heterogeneous graph neural networks. *IEEE Transactions on Services Computing*, pages 1–14, 2024. doi: 10.1109/TSC.2024.3463487.

[115] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

[116] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[117] Rúben Pereira, Isaias Scalabrin Bianchi, Ana Lúcia Martins, José Braga de Vasconcelos, and Álvaro Rocha. Business process modelling to improve incident management process. In Álvaro Rocha, Hojjat Adeli, Luís Paulo Reis, Sandra Costanzo, Irena Orovic, and Fernando Moreira, editors, *Trends and Innovations in Information Systems and Technologies - Volume 1, WorldCIST 2020, Budva, Montenegro, 7-10 April 2020*, volume 1159 of *Advances in Intelligent Systems and Computing*, pages 689–702. Springer, 2020. doi: 10.1007/978-3-030-45688-7_68. URL https://doi.org/10.1007/978-3-030-45688-7_68.

[118] Rúben Pereira, José Braga de Vasconcelos, Álvaro Rocha, and Isaias Scalabrin Bianchi. Business process management heuristics in IT service management: a case study for incident management. *Comput. Math. Organ. Theory*, 27(3):264–301, 2021. doi: 10.1007/S10588-021-09331-2. URL https://doi.org/10.1007/s10588-021-09331-2.

[119] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani, editors, *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 701–710. ACM, 2014. doi: 10.1145/2623330.2623732.

[120] Bryan Perozzi, Vivek Kulkarni, Haochen Chen, and Steven Skiena. Don't walk, skip! online learning of multi-scale network embeddings. In *International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, ASONAM '17, page 258–265, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349932. doi: 10.1145/3110025.3110086.

[121] Maja Pesic, Helen Schonenberg, and Wil M. P. van der Aalst. DECLARE: full support for loosely-structured processes. In *EDOC*, pages 287–300, 2007.

[122] Peter Pfeiffer, Johannes Lahann, and Peter Fettke. Multivariate business process representation learning utilizing gramian angular fields and convolutional neural networks. In *BPM*, volume 12875, pages 327–344. Springer, 2021.

[123] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Hugo Larochelle. Improving reproducibility in machine learning research(a report from the neurips 2019 reproducibility program). *J. Mach. Learn. Res.*, 22: 164:1–164:20, 2021. URL `https://jmlr.org/papers/v22/20-303.html`.

[124] Mirko Polato, Alessandro Sperduti, Andrea Burattin, and Massimiliano de Leoni. Time and activity sequence prediction of business process instances. *Computing*, 100(9):1005–1031, 2018.

[125] Carol Pollard and Aileen Cater-Steel. Justifications, strategies, and critical success factors in successful ITIL implementations in U.S. and australian companies: An exploratory study. *Inf. Syst. Manag.*, 26(2):164–175, 2009. doi: 10.1080/10580530902797540. URL `https://doi.org/10.1080/10580530902797540`.

[126] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *International Conference on Web Search and Data Mining (WSDM)*, WSDM '18, page 459–467, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355810. doi: 10.1145/3159652.3159706.

[127] Efren Rama-Maneiro, Juan Vidal, and Manuel Lama. Deep learning for predictive business process monitoring: Review and benchmark. *IEEE TSC*, pages 1–1, 2021.

[128] Efrén Rama-Maneiro, Fabio Patrizi, Juan Carlos Vidal, and Manuel Lama. Towards learning the optimal sampling strategy for suffix prediction in predictive monitoring. In Giancarlo Guizzardi, Flávia Maria Santoro, Haralambos Mouratidis, and Pnina Soffer, editors, *Advanced Information Systems Engineering - 36th International Conference, CAiSE 2024, Limassol, Cyprus, June 3-7, 2024, Proceedings*, volume 14663 of *Lecture Notes in Computer Science*, pages 215–230. Springer, 2024. doi: 10.1007/978-3-031-61057-8_13. URL https://doi.org/10.1007/978-3-031-61057-8_13.

[129] Vikas Raunak, Amr Sharaf, Yiren Wang, Hany Hassan Awadalla, and Arul Menezes. Leveraging GPT-4 for automatic translation post-editing. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 12009–12024. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-EMNLP.804. URL https://doi.org/10.18653/v1/2023.findings-emnlp.804.

[130] Aleksandra Revina. Assessing process suitability for ai-based automation. research idea and design. In Witold Abramowicz and Adrian Paschke, editors, *Business Information Systems Workshops - BIS 2018 International Workshops, Berlin, Germany, July 18-20, 2018, Revised Papers*, volume 339 of *Lecture Notes in Business Information Processing*, pages 697–706. Springer, 2018. doi: 10.1007/978-3-030-04849-5_59. URL https://doi.org/10.1007/978-3-030-04849-5_59.

[131] Nina Rizun, Aleksandra Revina, and Vera G. Meister. Analyzing content of tasks in business process management. blending task execution and organization perspectives. *Comput. Ind.*, 130:103463, 2021. doi: 10.1016/J.COMPIND.2021.103463. URL https://doi.org/10.1016/j.compind.2021.103463.

[132] Nina Rizun, Aleksandra Revina, and Vera G. Meister. Assessing business process complexity based on textual data: Evidence from ITIL IT ticket processing. *Bus. Process. Manag. J.*, 27(7):1966–1998, 2021. doi: 10.1108/BPMJ-04-2021-0217. URL https://doi.org/10.1108/BPMJ-04-2021-0217.

[133] Williams Rizzi, Luca Simonetto, Chiara Di Francescomarino, Chiara Ghidini, Tõnis Kasekamp, and Fabrizio Maria Maggi. Nirdizati 2.0: New features and redesigned backend. In *Proceedings of the Dissertation Award, Doctoral Consortium, and Demonstration Track at BPM 2019 co-located with 17th International Conference on Business Process Management, BPM 2019, Vienna, Austria, September 1-6, 2019*, pages 154–158, 2019. URL `http://ceur-ws.org/Vol-2420/paperDT8.pdf`.

[134] Marcello La Rosa, Hajo A. Reijers, Wil M. P. van der Aalst, Remco M. Dijkman, Jan Mendling, Marlon Dumas, and Luciano García-Bañuelos. APROMORE: an advanced process model repository. *Expert Syst. Appl.*, 38(6):7029–7040, 2011.

[135] Benedek Rozemberczki and Rik Sarkar. Fast sequence-based embedding with diffusion graphs. *CoRR*, abs/2001.07463, 2020.

[136] Anne Rozinat, R. S. Mans, Minseok Song, and Wil M. P. van der Aalst. Discovering simulation models. *Inf. Syst.*, 34(3):305–327, 2009.

[137] Nick Russell, ter Ahm Arthur Hofstede, Wil M.P. van der Aalst, and Na Nataliya Mulyar. *Workflow control-flow patterns: a revised view.* 2006.

[138] S. Salehi and A. Schmeink. Data-centric green artificial intelligence: A survey. *IEEE Transactions on Artificial Intelligence*, 5:1973–1989, 2024. doi: 10.1109/TAI.2023.3315272.

[139] Mohammadreza Fani Sani, Juan J. Garza Gonzalez, Sebastiaan J. van Zelst, and Wil M. P. van der Aalst. Conformance checking approximation using simulation. In *ICPM*, pages 105–112. IEEE, 2020.

[140] Shubhra Kanti Karmaker Santu, Md. Mahadi Hassan, Micah J. Smith, Lei Xu, Chengxiang Zhai, and Kalyan Veeramachaneni. Automl to date and beyond: Challenges and opportunities. *ACM Comput. Surv.*, 54(8):175:1–175:36, 2022. doi: 10.1145/3470918. URL `https://doi.org/10.1145/3470918`.

[141] Yutaka Sasaki et al. The truth of the f-measure. *Teach tutor mater*, 1(5): 1–5, 2007.

[142] Clemens Schreiber. Automated sustainability compliance checking using process mining and formal logic. In Ruzanna Chitchyan, Daniel Schien, Ana Moreira, and Benoît Combemale, editors, *ICT4S 2020: 7th International*

Conference on ICT for Sustainability, Bristol, United Kingdom, June 21-27, 2020, pages 181–184. ACM, 2020. doi: 10.1145/3401335.3401355. URL https://doi.org/10.1145/3401335.3401355.

[143] Alexander Seeliger, Stefan Luettgen, Timo Nolle, and Max Mühlhäuser. Learning of process representations using recurrent neural networks. In Marcello La Rosa, Shazia W. Sadiq, and Ernest Teniente, editors, *Advanced Information Systems Engineering - 33rd International Conference, CAiSE 2021, Melbourne, VIC, Australia, June 28 - July 2, 2021, Proceedings*, volume 12751 of *Lecture Notes in Computer Science*, pages 109–124. Springer, 2021. doi: 10.1007/978-3-030-79382-1_7. URL https://doi.org/10.1007/978-3-030-79382-1_7.

[144] Arik Senderovich, Chiara Di Francescomarino, Chiara Ghidini, Kerwin Jorbina, and Fabrizio Maria Maggi. Intra and inter-case features in predictive process monitoring: A tale of two dimensions. In Josep Carmona, Gregor Engels, and Akhil Kumar, editors, *Business Process Management - 15th International Conference, BPM 2017, Barcelona, Spain, September 10-15, 2017, Proceedings*, volume 10445 of *Lecture Notes in Computer Science*, pages 306–323. Springer, 2017. doi: 10.1007/978-3-319-65000-5_18. URL https://doi.org/10.1007/978-3-319-65000-5_18.

[145] Arik Senderovich, Chiara Di Francescomarino, and Fabrizio Maria Maggi. From knowledge-driven to data-driven inter-case feature encoding in predictive process monitoring. *Inf. Syst.*, 84:255–264, 2019. doi: 10.1016/J.IS.2019.01.007. URL https://doi.org/10.1016/j.is.2019.01.007.

[146] Gaurav Shekhar. Instructor led training and certification - ITIL V4 foundation. In Deepak Khazanchi, Harvey P. Siy, George Grispos, and Tenace Kwaku Setor, editors, *SIGITE '20: The 21st Annual Conference on Information Technology Education, Virtual Event, USA, October 7-9, 2020*, page 355. ACM, 2020. doi: 10.1145/3368308.3415453. URL https://doi.org/10.1145/3368308.3415453.

[147] Renuka Sindhgatta, Catarina Moreira, Chun Ouyang, and Alistair Barros. Exploring interpretable predictive models for business processes. In Dirk Fahland, Chiara Ghidini, Jörg Becker, and Marlon Dumas, editors, *Business Process Management - 18th International Conference, BPM 2020, Seville, Spain, September 13-18, 2020, Proceedings*, volume 12168 of *Lecture Notes in Computer Science*, pages 257–272. Springer, 2020. doi: 10.1007/978-3-030-58666-9_15. URL https://doi.org/10.1007/978-3-030-58666-9_15.

[148] Dominique Sommers, Vlado Menkovski, and Dirk Fahland. Supervised learning of process discovery techniques using graph neural networks. *Inf. Syst.*, 115:102209, 2023. doi: 10.1016/J.IS.2023.102209. URL `https://doi.org/10.1016/j.is.2023.102209`.

[149] Minseok Song and Wil M. P. van der Aalst. Towards comprehensive support for organizational mining. *Decis. Support Syst.*, 46(1):300–317, 2008.

[150] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *ArXiv*, abs/1906.02243, 2019. URL `https://api.semanticscholar.org/CorpusID:174802812`.

[151] Dennis L. Sun and Cédric Févotte. Alternating direction method of multipliers for non-negative matrix factorization with the beta-divergence. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6201–6205, 2014. doi: 10.1109/ICASSP.2014.6854796.

[152] Ajaya K. Swain and Valeria R. Garza. Key factors in achieving service level agreements (SLA) for information technology (IT) incident resolution. *Inf. Syst. Frontiers*, 25(2):819–834, 2023. doi: 10.1007/s10796-022-10266-5.

[153] Gabriel M. Tavares, Rafael Seidi Oyamada, Sylvio Barbon, and Paolo Ceravolo. Trace encoding in process mining: A survey and benchmarking. *Eng. Appl. Artif. Intell.*, 126(Part D):107028, 2023. doi: 10.1016/J.ENGAPPAI.2023.107028. URL `https://doi.org/10.1016/j.engappai.2023.107028`.

[154] Gabriel Marques Tavares, Sylvio Barbon Junior, Ernesto Damiani, and Paolo Ceravolo. Selecting optimal trace clustering pipelines with meta-learning. In *Intelli.Sys.*, 2022.

[155] Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. Predictive business process monitoring with LSTM neural networks. In *CAiSE*, volume 10253, pages 477–492. Springer, 2017.

[156] Farbod Taymouri, Marcello La Rosa, and Sarah M. Erfani. A deep adversarial model for suffix and remaining time prediction of event sequences. In *Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021, Virtual Event, April 29 - May 1, 2021*, pages 522–530. SIAM, 2021.

[157] Irene Teinemaa, Niek Tax, Massimiliano de Leoni, Marlon Dumas, and Fabrizio Maria Maggi. Alarm-based prescriptive process monitoring. In

Mathias Weske, Marco Montali, Ingo Weber, and Jan vom Brocke, editors, *Business Process Management Forum - BPM Forum 2018, Sydney, NSW, Australia, September 9-14, 2018, Proceedings*, volume 329 of *Lecture Notes in Business Information Processing*, pages 91–107. Springer, 2018. doi: 10.1007/978-3-319-98651-7_6. URL https://doi.org/10.1007/978-3-319-98651-7_6.

[158] Irene Teinemaa, Marlon Dumas, Marcello La Rosa, and Fabrizio Maria Maggi. Outcome-oriented predictive process monitoring: Review and benchmark. *ACM Trans. Knowl. Discov. Data*, 13(2):17:1–17:57, 2019. doi: 10.1145/3301300. URL https://doi.org/10.1145/3301300.

[159] Tanja Tornede, Alexander Tornede, Jonas Hanselle, Felix Mohr, Marcel Wever, and Eyke Hüllermeier. Towards green automated machine learning: Status quo and future directions. *J. Artif. Intell. Res.*, 77:427–457, 2023. doi: 10.1613/JAIR.1.14340. URL https://doi.org/10.1613/jair.1.14340.

[160] Leo Torres, Kevin S. Chan, and Tina Eliassi-Rad. GLEE: geometric laplacian eigenmap embedding. *J. Complex Networks*, 8(2), 2020. doi: 10.1093/comnet/cnaa007.

[161] Wil M. P. van der Aalst. Business process simulation survival guide. In *Handbook on Business Process Management, 2nd Ed*, International Handbooks on Information Systems, pages 337–370. Springer, 2015.

[162] Wil M. P. van der Aalst. *Process Mining - Data Science in Action, Second Edition*. Springer, 2016.

[163] Wil M. P. van der Aalst. On the pareto principle in process mining, task mining, and robotic process automation. In Slimane Hammoudi, Christoph Quix, and Jorge Bernardino, editors, *Proceedings of the 9th International Conference on Data Science, Technology and Applications, DATA 2020, Lieusaint, Paris, France, July 7-9, 2020*, pages 5–12. SciTePress, 2020.

[164] Wil M. P. van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–1142, 2004. doi: 10.1109/TKDE.2004.47. URL https://doi.org/10.1109/TKDE.2004.47.

[165] Wil M. P. van der Aalst, Kees M. van Hee, Arthur H. M. ter Hofstede, Natalia Sidorova, H. M. W. Verbeek, Marc Voorhoeve, and Moe Thandar Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal Aspects Comput.*, 23(3):333–363, 2011.

doi: 10.1007/S00165-010-0161-4. URL https://doi.org/10.1007/s00165-010-0161-4.

[166] Boudewijn F. van Dongen, Ana Karla A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and Wil M. P. van der Aalst. The prom framework: A new era in process mining tool support. In *ICATPN*, volume 3536 of *LNCS*, pages 444–454. Springer, 2005.

[167] Boudewijn F. van Dongen, Ana Karla Alves de Medeiros, and Lijie Wen. Process mining: Overview and outlook of petri net discovery algorithms. *Trans. Petri Nets Other Model. Concurr.*, 2:225–242, 2009. doi: 10.1007/978-3-642-00899-3_13. URL https://doi.org/10.1007/978-3-642-00899-3_13.

[168] Mathieu van Luijken, István Ketykó, and Felix Mannhardt. An experiment on transfer learning for suffix prediction on event logs. In Jochen De Weerdt and Luise Pufahl, editors, *Business Process Management Workshops - BPM 2023 International Workshops, Utrecht, The Netherlands, September 11-15, 2023, Revised Selected Papers*, volume 492 of *Lecture Notes in Business Information Processing*, pages 31–43. Springer, 2023. doi: 10.1007/978-3-031-50974-2_3. URL https://doi.org/10.1007/978-3-031-50974-2_3.

[169] Joaquin Vanschoren. Meta-learning: A survey. *CoRR*, abs/1810.03548, 2018. URL http://arxiv.org/abs/1810.03548.

[170] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[171] Ishwar Venugopal, Jessica Töllich, Michael Fairbank, and Ansgar Scherp. A comparison of deep-learning methods for analysing and predicting business processes. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021. doi: 10.1109/IJCNN52387.2021.9533742.

[172] H. M. W. Verbeek and Renata Medeiros de Carvalho. Log skeletons: A classification approach to process discovery. *CoRR*, abs/1806.08247, 2018.

[173] Ilya Verenich, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi, and Irene Teinemaa. Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. *ACM Trans. Intell. Syst. Technol.*, 2019.

[174] Mark von Rosing, Stephen White, Fred Cummins, and Henk de Man. Business process model and notation - BPMN. In *The Complete Business Process Handbook*, pages 429–453. Morgan Kaufmann/Elsevier, 2015.

[175] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL `http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html`.

[176] Sholom M. Weiss, Nitin Indurkhya, and Tong Zhang. *Fundamentals of Predictive Text Mining, Second Edition*. Texts in Computer Science. Springer, 2015. ISBN 978-1-4471-6749-5. doi: 10.1007/978-1-4471-6750-1.

[177] Hans Weytjens and Jochen De Weerdt. Process outcome prediction: CNN vs. LSTM (with attention). In Adela del-Río-Ortega, Henrik Leopold, and Flávia Maria Santoro, editors, *Business Process Management Workshops - BPM 2020 International Workshops, Seville, Spain, September 13-18, 2020, Revised Selected Papers*, volume 397 of *Lecture Notes in Business Information Processing*, pages 321–333. Springer, 2020. doi: 10.1007/978-3-030-66498-5_24. URL `https://doi.org/10.1007/978-3-030-66498-5_24`.

[178] Hans Weytjens and Jochen De Weerdt. Creating unbiased public benchmark datasets with data leakage prevention for predictive process monitoring. In *BPM*, volume 436, pages 18–29. Springer, 2021.

[179] Bemali Wickramanayake, Chun Ouyang, Yue Xu, and Catarina Moreira. Generating multi-level explanations for process outcome predictions. *Eng. Appl. Artif. Intell.*, 125:106678, 2023. doi: 10.1016/J.ENGAPPAI.2023.106678. URL `https://doi.org/10.1016/j.engappai.2023.106678`.

[180] Peter Y. H. Wong and Jeremy Gibbons. A process semantics for BPMN. In Shaoying Liu, T. S. E. Maibaum, and Keijiro Araki, editors, *Formal*

Methods and Software Engineering, 10th International Conference on Formal Engineering Methods, ICFEM 2008, Kitakyushu-City, Japan, October 27-31, 2008. Proceedings, volume 5256 of *Lecture Notes in Computer Science*, pages 355–374. Springer, 2008. doi: 10.1007/978-3-540-88194-0_22. URL https://doi.org/10.1007/978-3-540-88194-0_22.

[181] Brecht Wuyts, Hans Weytjens, Seppe vanden Broucke, and Jochen De Weerdt. Dylopro: Profiling the dynamics of event logs. In *BPM*, volume 14159 of *LNCS*, pages 146–162. Springer, 2023.

[182] Brecht Wuyts, Seppe Vanden Broucke, and Jochen De Weerdt. Sutran: an encoder-decoder transformer for full-context-aware suffix prediction of business processes. In *2024 6th International Conference on Process Mining (ICPM)*, pages 17–24, 2024. doi: 10.1109/ICPM63005.2024.10680671.

[183] Dingqi Yang, Paolo Rosso, Bin Li, and Philippe Cudre-Mauroux. Nodesketch: Highly-efficient graph embeddings via recursive sketching. In *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, KDD '19, page 1162–1172, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330951.

[184] Fareed Zandkarimi, Jana-Rebecca Rehse, Pouya Soudmand, and Hartmut Hoehle. A generic framework for trace clustering in process mining. In Boudewijn F. van Dongen, Marco Montali, and Moe Thandar Wynn, editors, *2nd International Conference on Process Mining, ICPM 2020, Padua, Italy, October 4-9, 2020*, pages 177–184. IEEE, 2020. doi: 10.1109/ICPM49681.2020.00034. URL https://doi.org/10.1109/ICPM49681.2020.00034.

Acknowledgements

La borsa di dottorato è stata cofinanziata con risorse del
Programma di finanziamento PON REACT-EU
Azione IV.5 – Dottorati su tematiche green
Questa tesi riflette solo i punti di vista e le opinioni degli autori, né l'Unione Europea né la Commissione Europea possono essere considerate responsabili per essi.

The doctoral scholarship was co-funded with resources in the frame of the
PON REACT-EU financing Program
Action IV.5 – Doctorates on green topics
This thesis reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

Author's Publications

1. Encoding Techniques for Digital Trace Data. S Barbon Jr., RS Oyamada, GM Tavares, P Ceravolo. Digital Trace Data Research in Information Systems: Foundations, Methods, and Applications, 2024 (to appear).

2. CoSMo: a Framework for Implementing Conditioned Process Simulation Models. RS Oyamada, GM Tavares, P Ceravolo. International Conference on Business Process Management (BPM), 2024. URL: https://doi.org/10.1007/978-3-031-70396-6_19

3. Enhancing Predictive Process Monitoring with Time-related Feature Engineering. RS Oyamada, GM Tavares, S Barbon Jr., P Ceravolo. International Conference on Advanced Information Systems Engineering (CAiSE), 2024. URL: https://doi.org/10.1007/978-3-031-61057-8_5

4. Trace Encoding in Process Mining: a survey and benchmarking. GM Tavares, RS Oyamada, S Barbon Jr., P Ceravolo. Engineering Applications of Artificial Intelligence, 2023. URL: https://doi.org/10.1016/j.engappai.2023.107028

5. The Dataset-Similarity-Based Approach to Select Datasets for Evaluation in Similarity Retrieval. M Matiazzo, V Castro-Silva, RS Oyamada and DS Kaster. Similarity Search and Applications (SISAP), 2023. URL: https://link.springer.com/chapter/10.1007/978-3-031-46994-7_11

6. A Scikit-learn Extension Dedicated to Process Mining Purposes (DEMO Paper). RS Oyamada, GM Tavares, S Barbon Jr., P Ceravolo. International Conference on Cooperative Information Systems (CoopIS) Demo Track, 2023. URL: https://ceur-ws.org/Vol-3552/paper-3.pdf

7. FEEED: Feature Extraction from Event Data. A Maldonado, GM Tavares, RS Oyamada, P Ceravolo, T Seidl. ICPM Doctoral Consortium/Demo, 2023. URL: https://ceur-ws.org/Vol-3648/paper_6598.pdf

8. A meta-learning configuration framework for graph-based similarity search indexes. RS Oyamada, LC Shimomura, S Barbon Jr., DS Kaster. Information Systems, 2023. URL: https://www.sciencedirect.com/science/article/abs/pii/S0306437922001016

9. A survey based on graph-based methods for similarity searches in metrics spaces. LC Shimomura, RS Oyamada, MR Vieira, DS Kaster. Information Systems, 2021. URL: https://www.sciencedirect.com/science/article/abs/pii/S0306437920300181

10. Towards Proximity Graph Auto-Configuration: an Approach Based on Meta-learning. RS Oyamada, LC Shimomura, S Barbon Jr., DS Kaster. European Conference on Advances in Databases and Information Systems (ADBIS), 2020. URL: https://link.springer.com/chapter/10.1007/978-3-030-54832-2_9