Utrecht University

ProRail

Graduate School of Natural Sciences
Master Artificial Intelligence

Enhancing Railway Operations: Rule-Based Event Log Extraction for ProRail Traffic Control

Utrecht University Master Thesis

Thijs van den Berg
1790064

ProRail Verkeersleiding
Afdeling Ontwikkeling Logistiek

Company supervisor:
J. Blom, MSc

University supervisors:
Dr. M.A. Mitici,
Prof. dr. M.M. Dastani

Version 1.1

Tuesday 16th July, 2024

t.vandenberg9@students.uu.nl

Abstract

Current research in process mining and event detection offers excellent interfaces and algorithms for discovering and improving real-world processes. However, these methods assume that the very activities that make up the processes are already captured in data. Algorithms capable of extracting event log data from generic tabular data sources are limited and often require significant manual effort or input from domain experts. A generic, easy-to-use approach that can transform tabular data into event log data is currently lacking.

This thesis addresses the gap through a case study conducted in collaboration with the Dutch railway infrastructure manager ProRail. ProRail oversees the maintenance, traffic control, management, and expansion of the Dutch railway network, with safety as its top priority. To ensure and enhance safety, it is crucial to understand the operational processes on the railway network, such as driving trains. Currently, ProRail does not automatically and in real-time monitor these user processes.

The main research question addressed by this thesis is: *'To what extent is it possible to automatically detect user processes in the data of ProRail?'*

This thesis proposes a comprehensive solution by:

1. Formalising a methodology to convert any generic tabular data source into event log data.
2. Formalising operational processes and their activities as process models.
3. Implementing a rule-based system (RBS) to extract activities from data.
4. Matching extracted activity sequences to predefined process models using the longest common subsequence (LCS) and token-based replay (TBR) algorithms.

The performance of the proof of concept (PoC) is assessed using the macro-averaged $\mathcal{F}_1$-score of fitness and precision. This metric accounts for the imbalance in the number of instances per process model.

The RBS extracted over $70,000$ activities from $2,506$ process instances (train journeys) present in the data. The created dataset annotated by the RBS serves as a ground truth for future research, such as training supervised learning models. The TBR algorithm found a single best process model for $2,448$ out of $2,506$ train journeys, while the LCS algorithm found $1,156$ single best matches. The macro-averaged $\mathcal{F}_1$-score of the LCS algorithm is 0.120 ± 0.087, while the TBR algorithm scores 0.104 ± 0.048.

A two-sided paired t-test ($t = 0.564$, $p = 0.629$) indicates no significant performance difference between the LCS and TBR algorithms. However, the TBR algorithm is preferred as it matches more than twice as many activity sequences to process models and supports multiple paths, loops, and parallel activities within a process model, which will be beneficial for future improvements to the processes. Inter-annotator agreement between two domain experts resulted in a Cohen's Kappa coefficient of $\kappa = 0.49$, indicating moderate agreement, while the agreement between each algorithm-expert pair ranged from poor to slight with κ-values between -0.06 and 0.09.

To conclude, it is possible to automatically detect user processes in the data of ProRail to some extent. Future improvements include refining current process models, incorporating additional data sources to capture activities currently unable to be implemented, and enhancing the RBS by allowing a percentage of rules to be true for an activity rather than requiring all rules to be true. The annotation of more data by the experts will further improve the quality of the constructed event log dataset. More research is needed to improve the ordering of activities within an event and the detection of multiple processes within a single train journey. The use of semantic similarity measures to match activities to dataset columns is very promising and should be further explored.

Keywords: process mining, event log extraction, sequence mining, event detection, longest common subsequence, token-based replay, ProRail, railways

Contents

Preface

You are reading the master thesis *'Enhancing Railway Operations: Rule-Based Event Log Extraction for ProRail Traffic Control'*. This master thesis is part of the master's program in Artificial Intelligence (AI) at Utrecht University (UU).

Exploring process mining, a field that was new to me, has been an exciting and educational journey. The focus of this thesis is on extracting processes and their activities from unlabelled tabular data, a topic that should interest both academics and professionals in AI, process mining, and railway operations.

I am deeply grateful to everyone at ProRail for their invaluable insights, suggestions, feedback, and engaging discussions. Special thanks go to my supervisor Jehudi Blom. It has been an honour to learn all the ins and outs of the Dutch railway system from you. I would also like to thank my other colleagues at ProRail, especially Rosa, Harm-Jaap, René, Astrid, and Eline, among many others. Additionally, I am grateful to Dr. Mitici from Utrecht University for her guidance and support throughout this project.

As this thesis marks the end of my studies (for now, who knows what the future holds!), I would like to thank my family and friends for their support and encouragement. A special mention goes to Casper Smet and Stan Meyberg, two friends with whom I have worked together on many different courses and projects throughout both the bachelor's and master's programs. I am proud of what we have achieved together.

Thijs van den Berg
Veenendaal, Tuesday 16th July, 2024

1 Introduction

Chapter 1 provides an introduction to the problem that this thesis tries to find an answer to and to the graduation company. The aim of this thesis is defined in section 1.3. The main research question and accompanying sub-questions are defined in sections 1.4 and 1.5, respectively.

1.1 Context

This thesis project is conducted in collaboration with ProRail B.V. ProRail is the Dutch national railway infrastructure manager. Safety on and along the railway track is central to ProRail's operations. The following is a summary of the core tasks of the company (ProRail, 2024):

- Maintaining existing railway tracks
- Managing and maintaining railway stations
- Construction of new railway tracks and stations
- Informing passenger and freight operators
- Train traffic control
- Allocation and distribution of capacity on the railway tracks

The research is conducted specifically within the subdivision Logistics Development (Dutch: *Ontwikkeling van de Logistiek*) of the division Traffic Control (Dutch: *Verkeersleiding*). The team consists of ±30 Full-time equivalent (FTE).

In 2023, ProRail had ±5.071 employees and managed 399 train stations divided across 7.002 kilometres of railway track. In 2023, the cumulative kilometres travelled by trains in the Netherlands amounted to 152 million (ProRail, 2024).

1.2 Problem Definition

Driving trains entails a multitude of processes that need to be followed to ensure safety on and alongside the railway tracks. These operational processes are known as user processes or gebruikersprocessen (GPs) in Dutch. Each GP describes the activities that need to be conducted by the train operator, the traffic controller (treindienstleider or Trdl in short) and the systems they communicate with. The GPs can be seen as a set of agreements and operational rules to which all stakeholders must adhere (Operationeel Kenniscentrum ERTMS [OKE], 2023). They prescribe how the stakeholders, such as train drivers and the Trdl, must interact with the system to minimise risk to those using the transport system.

All processes relevant for this thesis are described in ProRail Assetmanagement (ProRail AM, 2023). This document is designed for the railway track between the Dutch cities of Amsterdam (Asd) and Utrecht (Ut). The processes describe situations such as a short stop (GP-6) or parking a train (GP-7). They also describe higher-risk situations such as driving on a slippery railway track (GP-12) and the procedure after unintendedly driving through a red signal (GP-37).

Currently, incidents are reported after the fact and only when they have been the cause of a safety breach or caused significant delays. The associated consequences and possible risks are subsequently identified. Using the reports produced, lessons are drawn for the future. The incident reports cover whether the appropriate steps have been followed for the processes applicable to the incident.

While some events[1], such as splitting a train into two separate trains, may be detected automatically, the procedures that describe how such an event needs to be handled are not monitored or reported by an automated system. There is currently no system which monitors if and when processes are executed. This has multiple downsides:

[1] This includes but is not limited to, incidents.

- No statistics on how often processes are executed (incorrectly)
- No data-driven risk analysis available for each process
- Error-prone and time-consuming manual incident reporting
- Possibility that not all incidents are currently reported
  - I.e. incidents that were deemed insignificant

These downsides can greatly impact the safety and efficiency of the company and all stakeholders involved.

The original problem definition by ProRail can be found in Appendix 6.1.

1.3 Aim and Scope

This thesis project aims to develop a general formalisation and methodology for a system that automatically extracts predefined processes from tabular data. The proposed method is furthermore implemented as a PoC for ProRail.

Once processes are found in the data, they can be monitored and reported. Monitoring the processes makes it possible to uncover whether and in what way they are executed. The processes can be reviewed and tested for safety based on real life scenarios found in the data. Furthermore, unforeseen scenarios and risks can be discovered and exposed. Possible findings based on the automatic extraction of processes from data include:

- Discovery of unsafe or unknown shortcuts taken by the stakeholders
- Edge cases in the train protection software
- Finding structural errors or missing activities in the description of current processes
- Confirming or reassessing the calculated risk for the various processes

Based on these findings, the activities involving each process can be modified to adhere to the new scenarios uncovered in data. Furthermore, new findings and irregularities can be shared with train operators and traffic controllers. This way, a feedback loop emerges of continuous improvement to the described processes, the staff, and safety in general.

1.3.1 Scope

This thesis is the first of many steps towards the improvement and monitoring of operational processes. Subsequent steps, such as improving the existing operational processes and the implementation of a feedback loop, are out-of-scope for this project.

The thesis is specifically focused on developing a general formalisation and methodology and testing this with a proof of concept (PoC) and not on developing a production-ready application. Showcasing the PoC's potential on a subset of processes allows ProRail to evaluate its performance.

Alongside the railway track between Amsterdam and Utrecht, the processes described in ProRail AM (2023) also apply to the *Hanzelijn*: the track between the cities Lelystad (Lls) and Zwolle (Zl). However, this track is outside the scope of the thesis project. The specifications and suppliers of the trackside equipment differ from those of the track between Amsterdam and Utrecht.

The operational processes across other railway tracks in the Netherlands differ in various degrees. A developed PoC is however highly applicable to other collections of GPs since the data source used is homogeneous across the Netherlands. Furthermore, the developed theory and methodology can be applied to other domains with similar input data formats.

1.4 Research Question

Based on the problem definition, aim and scope of the thesis, the following main research question is formulated:

To what extent is it possible to automatically detect user processes in the data of ProRail?

1.5 Sub-questions

To answer the main research question, the following sub-questions are formulated:

RQ1 What data sources provide predictive information about the user processes?
RQ2 How can the description of a user process and its activities be encoded into a formal definition?
RQ3 How can the formal definition be used to extract a sequence of activities from data?
RQ4 Which algorithm is best in identifying user processes in the retrieved sequences of activities using the formal definitions?
RQ5 What evaluation metric is best to determine the performance of the system?

The available data is examined and formalised in the first research question (RQ). Translating the predefined user processes into a general and formal definition is covered in RQ2. The formal definition can then be used to extract a sequence of activities from the data, as covered in RQ3. RQ4 examines the classification of retrieved sequences of activities to processes using a variety of algorithms. To determine the overall performance of the PoC, suitable evaluation metrics are required. This is discussed in RQ5.

1.6 Readers Guide

Section 2.1 gives a general overview of the Dutch railway system. This section provides additional context to the problem at hand and introduces various concepts relevant to the thesis. Section 2.2 is a deeper dive into the operational processes regarding this thesis. The data used for the PoC is presented in section 2.3.[2] The scientific basis of this study is described in section 2.4. Based on this literature review, the research gap is presented in section 2.5. Section 2.6 gives an overview of the ethical considerations that are made for this thesis.

Chapter 3 describes the various steps taken to answer each research question. These steps include the data collection and the development of the PoC along with various formalisations and definitions. The chapter concludes with the methods used to evaluate the PoC.

Chapter 4 presents the results obtained for each of the sub-questions. The results were obtained through the literature review in the theoretical framework and the methodology constructed in Chapter 3.

We start Chapter 5 with the findings on each of the sub-questions and provide an answer to the main question in section 5.1. Section 5.2 is dedicated to the discussion. We conclude the chapter with suggestions for further research in section 5.3.

[2] Any processing of the data will however be described in the methodology in Chapter 3.

2 Theoretical Framework

Section 2.1 gives a general overview of the Dutch railway system. This section provides additional context to the problem at hand and introduces various concepts relevant to the thesis. Section 2.2 is a deeper dive into the operational processes regarding this thesis. The data used for the PoC is presented in section 2.3.[1] The scientific basis of this study is described in section 2.4. Based on this literature review, the research gap is presented in section 2.5. Section 2.6 gives an overview of the ethical considerations that are made for this thesis.

2.1 The Dutch Railway System

The railway track between Amsterdam and Utrecht is a dual signalling track, which means that two signalling systems are used: both NS'54 and ERTMS. A dual signalling system allows two types of trains to be run on the same track: trains using the NS'54 system and trains using ERTMS.

Figure 2.1: A passenger and freight train travelling side by side on the dual signalling track between Amsterdam and Utrecht (ProRail ERTMS Integratie Lab, 2020, p. 5).

2.1.1 NS'54 and ATB

The NS'54 signalling system, as specified by Rijksoverheid (2023), consists of light signals and signs that direct train traffic. The standard was first implemented in 1955 (Middelraad, 2000, p. 22) and is still in use today. Almost the entire Dutch railway network is equipped with the NS'54 signalling system.

This signalling system is often used in combination with automatische treinbeïnvloeding (ATB), Dutch for automatic train control. There are several versions of ATB, which prevent trains from exceeding the speed limit or passing a red signal. If a train exceeds the speed indicated by the NS'54 system, ATB will intervene and apply an emergency brake. The train will then automatically come to a complete stop.

[1] Any processing of the data will however be described in the methodology in Chapter 3.

2.1.2 ERTMS and ETCS

For historical reasons, each Member State of the EU has its own unique and proprietary National Train Control (NTC) system. ATB is the NTC system of the Netherlands. International trains therefore need to be equipped with several NTC systems, one for each country the train passes through on its journey. In addition, the train operator needs to be properly trained and certified for all the different signalling systems they may encounter.

The European Rail Traffic Management System (ERTMS) is a set of specifications for both the signalling system and the train control system (ProRail, 2019). The specifications are designed by the European Union (EU) and enable the interoperability of trains across the EU. By streamlining the signalling and automatic train control systems across all Member States, train operation will become more efficient, more flexible, cheaper and, above all, safer.

Part of the ERTMS specification is the European Train Control System (ETCS) (European Union Agency for Railways [ERA], 2023). This will replace the NTC of all participating countries. Replacing ATB with ETCS will require major modifications to both the tracks and trains in the Netherlands. According to the latest planning, ERTMS is expected to be implemented on all tracks in the Netherlands in 2050 (ERTMS NL, 2022). The current status and distribution of automatic train control systems in the Netherlands can be found in Appendix 6.2.

2.1.3 Trains Driving Under ETCS

Currently, only a limited number of trains can run under the ETCS regime. It is important for the data selection process in section 3.1 to know which trains are equipped with ETCS. The GPs used for this thesis are only applicable to trains equipped with ETCS. The following trains are known to be equipped with ETCS:

- Sprinter Nieuwe Generatie (SNG) (Nederlandse Spoorwegen [NS], 2023b)
  - Scheduled to run between Asd and Ut in the 2023 and 2024 timetables
- Intercity Nieuwe Generatie (ICNG) (NS, 2023a)
  - Not scheduled to run between Asd and Ut in the 2023 and 2024 timetables
- International freight and passenger trains that pass through the Netherlands
  - These trains are often equipped with ETCS to ensure interoperability across multiple countries
- Older generations of trains from NS that are currently being retrofitted with ETCS (NS, 2022)

Occasionally, other trains are also equipped with ETCS such as trains for trackside maintenance and testing. In the last quarter of 2023 (ProRail, 2022), a project called 'Ervaringsrijden', Dutch for 'Experience Driving', was launched. During this project, train operators were trained to drive using ETCS. After completing their training, the train operators regularly drive using ETCS in order to gain and maintain their experience (NS, 2022). The project is carried out on the track between Amsterdam and Utrecht and on the Hanzelijn.

2.2 User Processes

This section provides an in-depth overview of the GPs and further elaborates on them.

2.2.1 Definition of a User Process

In total, there are forty GPs described in ProRail AM (2023). A table of all forty processes and their (translated) names can be found in Appendix 6.3.

A GP consists of a text describing when the process is applicable, a set of prerequisites, and a sequence diagram containing all activities of the process. This sequence diagram only visualises a single flow of the process. Some processes have accompanying notes that describe possible deviations and the actions to be taken in such cases. The sequence diagram in Figure 2.2 shows

the first four out of fifteen activities that are part of GP-1. As the diagram originates from ProRail AM (2023), the text is in Dutch. It does however provide a clear overview of the interactions between the actors involved in the process.

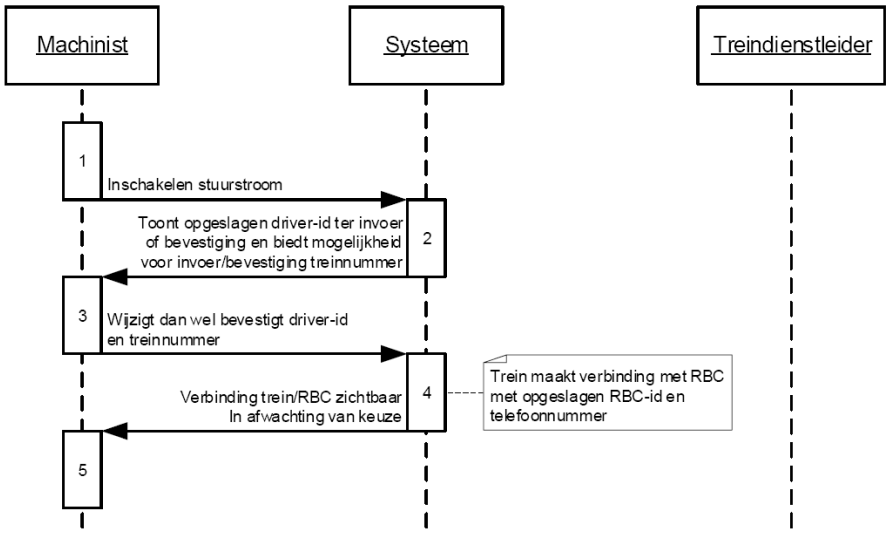

Figure 2.2: First four out of fifteen activities of GP-1: *Heading towards a normally set route with known train position* from ProRail AM (2023). An arrow from one actor to another indicates an interaction between the two. The arrow is annotated with the activity that takes place between the two actors. Notes can be added to the diagram for additional information, as can be seen on the right side of the fourth activity.

The processes described will need to be revised once the track is eventually upgraded from dual signalling to an ETCS-only system. Not all described processes are relevant for the Amsterdam – Utrecht railway track. For example, some processes describe situations around railway crossings or tunnels, both of which are not present between Asd – Ut.

2.2.1.1 Interaction Between Actors

Every activity of a process describes the communication from one actor to another actor. The actor can be either a Trdl, a train operator, or the system. The system refers to all equipment with which the Trdl and train operator can interact. Furthermore, the Trdls and train operators can interact with each other via phone calls.

Interaction is not necessarily between one Trdl and one train operator. For example, if a train is split into two separate trains (GP-51), two train operators are involved. Sometimes the rail carrier is also involved in a process. For example, if the freight or passenger carrier notifies their train operator of a slippery track (GP-12). Furthermore, the boundary between the dual signalling track and ATB can coincide with the boundary of two traffic control posts (GP-12 & GP-15). In these cases, not one but two Trdls are involved.

2.2.1.2 Prerequisites

Most processes contain a set of prerequisites. These are conditions that are assumed to be true at the start of the process. For instance, GP-15 *Transition from Level 2 to level NTC ATB*, assumes the following prerequisites:

- A train is driving in a dual signalling protected area under Level 2 (L2), mode FS.
- The train needs to enter an ATB-EG protected area.

2.3 Data

ProRail's largest data warehouse is called Sherlock. It brings together over 250 data sources from 90 different systems. This data includes both infrastructure and traffic information from the past, present, and future. Data is stored in tabular form and consists of 733 columns, totalling over 125 Terabyte (TB) of data. Sherlock was originally designed to investigate the cause of delays and the true arrival times of trains (ProRail, 2023, p. 274). Today, the data warehouse has evolved into a data provider for a wide range of use cases, including this thesis.

Each row in the tabular data represents a step in the timetable of each train. These timetable steps are called dienstregelpunten (DRPs). A DRP is a geographical point on the track. Typical locations for a DRP to be situated include railway stations, bridges, and rail junctions (Infrasite, 2021). A train can perform various *DRP activities*. Each new DRP activity at a DRP is logged as a new entry, the DRP activities include:

- (Short) arrival at DRP
- (Short) departure from DRP
- Passage of DRP
- Shunting at DRP

Information that is logged while at a DRP includes the planned and actual time the DRP activity happened. The track between Amsterdam and Utrecht consists of 22 DRPs which can be seen in Appendix 6.4.

Each train journey is assigned a train number. This number is unique for that day and remains the same for the entire journey. On each journey, the train passes several DRPs. An excerpt of the data from Sherlock is shown in Table 2.1.

Train Number	**DRP**	**DRP activity**	**Planned**	**Actual**	...
5732	Vspa	Passage	18-12-2023 11:30:00	18-12-2023 11:30:13	...
5739	Dmnz	Short Departure	18-12-2023 11:29:30	18-12-2023 11:30:30	...
7339	Bkla	Passage	18-12-2023 11:28:42	18-12-2023 11:30:33	...
220	Ut	Arrival	18-12-2023 10:59:42	18-12-2023 11:30:35	...
3139	Bkl	Passage	18-12-2023 11:30:42	18-12-2023 11:30:45	...
3139	Bkla	Passage	18-12-2023 11:30:54	18-12-2023 11:30:57	...
3934	Aco	Passage	18-12-2023 11:31:18	18-12-2023 11:30:58	...
...	...	...	...	...	...

Table 2.1: Adapted excerpt of the data from Sherlock. The first column contains the unique identifier for a train journey. The 'DRP' column contains the unique identifier for the current DRP. The 'DRP activity' is the type of DRP activity that was performed while at the DRP. The 'Planned' and 'Actual' columns contain the planned and actual time of the DRP activity. The actual data contains more columns and rows than shown here.

2.4 Literature Review

This section provides a literature review of the relevant topics for this thesis. We start this section with the introduction of process mining in subsection 2.4.1. We then continue with introducing event detection in subsection 2.4.2 and conclude with the theoretical background of sequence matching in subsection 2.4.3.

2.4.1 Process Mining

Process mining focuses on discovering, monitoring, and improving real processes. Van der Aalst et al. (2012) provides a process mining framework to extract knowledge from event log data. An event log is a sequential record of events that stores activities, i.e. steps of a process, and is related to a specific case, i.e. a process instance Van der Aalst et al. (2012, p. 174). Additional data such as timestamps may also be stored in the event log. An example event log can be seen in Table 2.2. The framework defines the following concepts:

- **Process model:** A model consisting of activities that are ordered in time and formally describe a process
- **Activity:** An event from an event log; A well-defined step in some process (Van der Aalst, 2010)
- **Case:** An instance of a process model
- **Resource:** The actor that performs the activity. Also referred to as the originator or performer (Van Dongen et al., 2005; Weijters et al., 2006)
- **Timestamp:** The time at which the activity was performed
- **Data elements**: Additional data that is associated with the activity

Case ID	Activity ID	Resource	Timestamp
1	A	John	09-03-2004 15:01
3	A	Sue	09-03-2004 16:03
3	B	Carol	09-03-2004 16:07
1	B	Mike	09-03-2004 18:25
1	C	John	10-03-2004 09:23

Table 2.2: An example event log, adopted from Weijters et al. (2006). Entries with the same Case ID belong to the same process instance. The 'Resource' is the actor initiating the event described in the column 'Activity ID'. The activity was performed at the timestamp described in the column 'Timestamp'.

Process mining techniques can be divided into three categories (Van der Aalst et al., 2012):

- **Discovery:** Discover a process model based on event log data, without any prior formal knowledge of the process.
- **Conformance checking**: Comparing whether the event log data and the process model are in line with each other.
- **Enhancement:** Improving the process model based on the event log data.

In the subsequent sections, we will first discuss the visualisation of process models in subsubsection 2.4.1.1. We then continue with an overview of the three categories of process mining mentioned above.

2.4.1.1 Visualising process models

As textual descriptions of processes can often be ambiguous, it is common to visualise process models graphically (Dumas et al., 2018, p. 16). This is often done using Petri Nets (Petri, 1962) or the Business Process Model and Notation (BPMN) standard (Object Management Group [OMG], 2011). An example of a process model in BPMN can be seen in Figure 2.3.

BPMN differs from the sequence diagrams of the GPs in several ways. Firstly, BPMN is more expressive. It allows for the modelling of multiple start and end activities, multiple paths, and loops. The current sequence diagrams only model a single flow of each process. Furthermore, variations and exceptions are described in separate notes and are not part of the sequence diagrams. This could lead to inconsistencies between the sequence diagrams and the accompanying notes (Van der Aa et al., 2017).

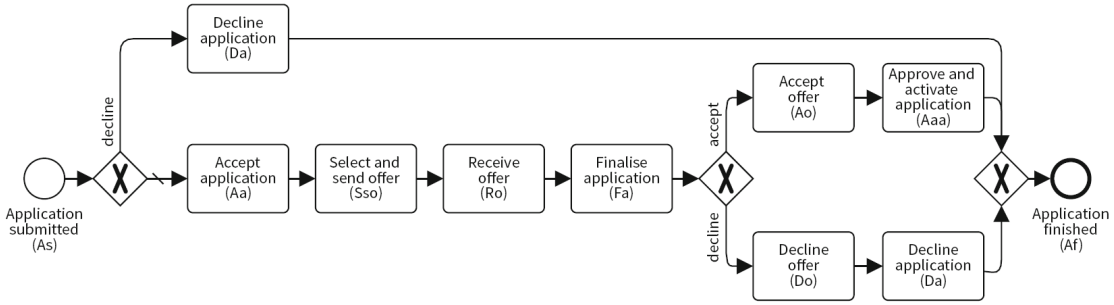

Figure 2.3: Example of a BPMN process model, extracted from Carmona et al. (2018, p. 24). The process starts with the 'Application submitted (As)' activity on the left. It ends at the right with the 'Application finished (Af)' event. Multiple paths can be observed.

Secondly, BPMN is more formal as it has a well-defined syntax and semantics that is easier for computers to understand. BPMN can be parsed and interpreted by a computer, whereas the sequence diagrams, in their current pdf form, cannot. This makes the BPMN standard more suitable for automated analysis.

Petri nets are constructed with places, transitions, arcs and tokens. The arcs connect places to transitions and vice versa. Arcs are directional edges and never connect places to places or transitions to transitions. The place at the start of a process is often referred to as the source place. The sink place is the place at the end of a process. Places that have an arc to a transition are its input places, and places that come out of a transition are its output places.

A token can be created in the source place at the start of the process. A token can also be created for any output place if the connected transition fires. The token at an input place is consumed when a transition fires. A transition can only fire when all input places have sufficient tokens. A sufficient amount of tokens is defined by the multiplicity of the arc. The multiplicity of an arc is the number of tokens needed for that arc to satisfy the transition. If multiple transitions can be fired, they are fired in a non-deterministic order. This is one of the reasons why Petri nets are often used to model concurrent systems (Peterson, 1977). Petri nets furthermore "provide a comprehensive mathematical foundation" and also offer a "plethora of formal results and analysis techniques" according to Carmona et al. (2018, p. 109). An example of a Petri net can be seen in Figure 2.4.

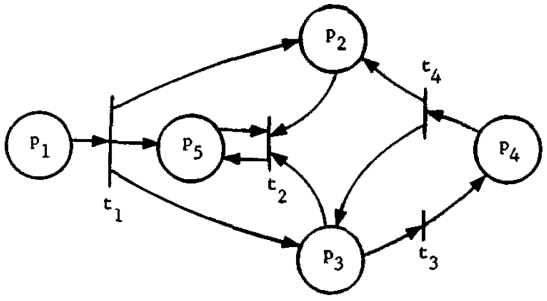

Figure 2.4: Example of a Petri net from Peterson (1977, p. 235). The Petri shown is defined as $C = (P, T, I, O)$ where $P = \{p_1, p_2, p_3, p_4, p_5\}$, $T = \{t_1, t_2, t_3, t_4\}$, $I(t_1) = \{p_1\}$, $I(t_2) = \{p_2, p_3, p_5\}$, $I(t_3) = \{p_3\}$, $I(t_4) = \{p_4\}$, $O(t_1) = \{p_2, p_3, p_5\}$, $O(t_2) = \{p_5\}$, $O(t_3) = \{p_4\}$, $O(t_4) = \{p_2, p_3\}$.

Formally, a Petri net, as defined by Peterson (1977, pp. 234–235), is a tuple $C = (P, T, I, O)$ where P is a set of places, T is a set of transitions, I is an input function and O is an output function. The input function I defines the set of input places for any transition $t_j \in T$. Likewise,

the output function O defines the set of output places for any transition t_j. The set of input and output places of a transition t_j are defined as $I(t_j)$ and $O(t_j)$ respectively. As a place can never be a transition and vice versa, the sets P and T are disjoint. Petri nets are thus not only directed graphs but also bipartite graphs as two distinct groups of nodes exist: places and transitions (Berti & van der Aalst, 2019).

Instead of the input and output functions I and O, we can define the Petri net as $C = (P, T, F)$ where F is a set of arcs $F \subseteq (P \times T) \cup (T \times P)$. This definition can be easily extended to include the multiplicity of each arc: $W : F \to \mathbb{N}$. In this case, the Petri net is defined as $C = (P, T, F, W)$ where W is the multiset of arcs F. The count of each arc in the multiset is the weight of the arc (Berti & van der Aalst, 2019). Peterson (1977, p. 246) refers to this as *generalized Petri Nets* and is visualised by drawing multiple arcs between places and transitions. In this case, the number of arcs between a place and a transition indicates the weight of the arc.

2.4.1.2 Discovery

Process discovery algorithms focus on extracting a process model from event log data (Van der Aalst, 2010; Van der Aalst et al., 2012) and enable the discovery of new information such as alternative paths (Van der Aalst et al., 2004). This is often done without any a priori knowledge of the process model itself. However, the activities that make up the processes are often already identified within the data. Process discovery thus focuses on finding the processes that consist of sequences of activities and not on finding the activities themselves.

The α-algorithm by Van der Aalst et al. (2004) was one of the first process discovery algorithms ever proposed. One of the limitations of the algorithm is that it cannot detect short loops of length one and two (Van der Aalst et al., 2004, p. 1138). This should pose no problem for the thesis as none of the processes covered in this thesis contain loops.

HeuristicMiner by Weijters et al. (2006) is a process discovery algorithm that uses heuristics to handle noise. It does this by focusing on the primary behaviour of processes and not on details and exceptions. The algorithm tries to discover the different roles that resources can have within a process model. This way, relationships between different resources become more clear which adds to the predictive power of the algorithm.

Măruşter et al. (2006) propose a rule-based approach to process discovery. The algorithm automatically induces two rule sets from data to predict the relations between various activities. The first rule set tries to detect causal relationships between activities whereas the second rule set tries to detect both parallel and exclusive relationships. Subsequently, the model applies the α-algorithm by Van der Aalst et al. (2004) to construct a final process model.

2.4.1.3 Conformance checking

Conformance checking is the process of verifying whether an event log and a process model are in line with each other. Following Carmona et al. (2018), conformance checking tries to investigate the following: *"How do the modelled behaviour of a process and its recorded behaviour relate to each other?"*. Rozinat and van der Aalst (2006) describe two metrics to measure the conformance between event log data and a process model:

- **Fitness:** the extent to which the log traces can be associated with execution paths specified by the process model
- **Appropriateness:** the degree of accuracy in which the process model describes the observed behaviour, combined with the degree of clarity in which it is represented
  - Split into (1) structural appropriateness and (2) behavioural appropriateness

In this definition, log traces are the sequences of activities that make up a process instance in the event logs. The execution paths are the possible paths that can be traversed in the process model. Rozinat and van der Aalst (2006) state that fitness could be measured with string distance measures, such as described below in subsection 2.4.3, but that they have a limited applicability.

String distance measures will show poor performance when a process model contains parallels or loops (Rozinat & van der Aalst, 2006).

Appropriateness focuses on minimising both the structure as well as the behaviour of a process model. A minimal structure will result in clearer behavior as a too-detailed structure will be hard to interpret. Minimal behaviour will ensure that the process model closely follows the *actual* behaviour of the process in real life, i.e. in the data (Rozinat & van der Aalst, 2008).

Carmona et al. (2018) considers two quality metrics to quantify the conformance between a process model and event log data, similar to the definition of Rozinat and van der Aalst (2006):

- **Fitness:** Has the recorded behaviour been modelled?
- **Precision:** Has the modelled behaviour been recorded?

An ideal process model will have a high degree of both fitness and precision. In this act of balancing, as mentioned by Carmona et al. (2018, p. 43), a similarity can be observed to precision, recall and their harmonic mean: the F_1-score (Chinchor & Sundheim, 1993; Stein Dani et al., 2023; Van der Aa et al., 2017). Fitness is defined as the fraction of log traces allowed by the process model:

$$\text{Fitness} = \frac{|L \cap M|}{|L|} \tag{2.1}$$

Here, L is the recorded behaviour in the event log data. M is the modelled behaviour of the process model. Fitness falls within the range of $[0, 1]$. A value of 1 indicates that all behaviour of the event log data is captured by the process model, i.e. $L = M$. A value of 0 indicates that none of the recorded behaviour is captured by the model (Carmona et al., 2018, p. 46), i.e. no overlap between L and M.

Precision is defined as the fraction of modelled behaviour that is also recorded in the event log data. It is defined as:

$$\text{Precision} = \frac{|L \cap M|}{|M|} \tag{2.2}$$

Precision too falls within the range of $[0, 1]$. A value of 1 indicates that all behaviour of the process model is captured in the event log data. I.e., all possible paths in the process model are present in the event log data. A value of 0 indicates that none of the behaviour of the model is captured in the data.

A process model that achieves both a high fitness and a high precision score is not guaranteed to perfectly capture the actual real life process. An example of an overfitted situation is visualised by Carmona et al. (2018) in Figure 2.5. In the case of overfitting, a process model perfectly captures the log data behaviour and vice versa but does only minimally capture real process behaviour.

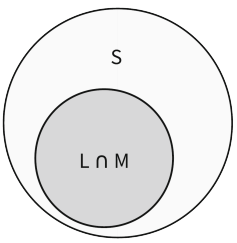

Figure 2.5: An example of overfitting. The behaviour of the log data L and the process model M are perfectly aligned. They do however only capture parts of the actual process S. Extracted from Carmona et al. (2018, p. 56).

A popular approach to determine the conformance between an event log and a process model is to investigate their alignment (Carmona et al., 2018; Van der Aalst, 2012). A perfect alignment

is found when the log trace can be *exactly* followed by a path in the process model. Alignment algorithms try to find the path with the least amount of deviations: the optimal alignment.

An example of a perfect alignment can be seen in Table 2.3. A move where the activity of the log trace and process model are identical is called a 'synchronous move' (Carmona et al., 2018).

Log trace	T	R	A	I	N	S
Process model path	T	R	A	I	N	S

Table 2.3: A perfect alignment between a log trace and a path in the process model. Each letter indicates an activity. Six synchronous moves can be observed.

A move where an activity of the log trace is not executed when it should have according to the process model is referred to as a 'model move'. Likewise, an activity executed in the log trace but not in the process model is referred to as a 'log move'. An example alignment with both model and log moves can be seen in Table 2.4.

Log trace	T	R	A	I	≫	N	S
Process model path	T	R	A	≫	M	≫	S

Table 2.4: An alignment between a log trace and a path in a process model. This alignment contains four synchronous moves, two log moves, and one model move. Deviations are indicated with ≫, following the conventions of Carmona et al. (2018).

Instead of using model and log moves to detect deviations, Rozinat and van der Aalst (2008) elaborates on a method called token-based replay (TBR). This algorithm tries to replay a log trace on the Petri Net representation of a process model. In doing so, it keeps track of the number of produced, consumed, missed and remaining tokens.

At the start of a process instance, a token is created for the source place. Thus, increasing the number of produced tokens from zero to one. The tokens of the input places of a transition are consumed once the transition fires, increasing the number of consumed tokens. A new token is produced for each output place of the fired transition. It could occur that a transition should have fired according to the replayed trace but could not as not enough tokens were present in any of its input places. In this case, missing tokens are inserted and the count of missing tokens is increased accordingly. Tokens that are produced but not consumed after the process instance has ended are considered remaining tokens.

To measure the conformance of a log trace to a process model based on the TBR algorithm, Berti and van der Aalst (2019) have defined the following metric:

$$\text{fitness}_{PN} = \frac{1}{2}(1 - \frac{m}{c}) + \frac{1}{2}(1 - \frac{r}{p}) \tag{2.3}$$

Here, m is the total number of missing tokens after the replay of a log trace on a process model, c is the number of consumed tokens, r is the number of remaining tokens, and p is the number of produced tokens. The fitness metric falls within the range of $[0, 1]$. A perfect score of 1 indicates that all produced tokens are consumed and no tokens are missing or remaining.

2.4.1.4 Enhancement

Whereas conformance checking only investigates the alignment between a process model and an event log, process enhancement goes one step further and tries to improve or extend an existing model based on data. Often, this is done by adding additional information to the process model, such as timestamps or other relevant data stored in the event logs (Van der Aalst et al., 2012).

Enhancing the current processes is out-of-scope for this thesis. Process enhancement algorithms are however very relevant for future research.

2.4.1.5 Event log extraction

Process discovery, conformance and enhancement algorithms all rely on event log data in which activities are already annotated. Making data suitable for process mining and transforming it in event log data often requires substantial amounts of work (Stein Dani et al., 2022). A series of steps that describe the human effort required to transform raw data into event log data is described by Stein Dani et al. (2022). This includes steps such as 'Assess Data Quality', 'Define Activity Names' and 'Anonymize Data' and is split into the following five topics:

- Context and Scope Definition
- Data Source Assessment
- Attribute Selection
- Data Source Extraction
- Event Log Assessment

These guidelines help to transform data into event log data in a structured and reproducible way. By examining the role of humans in event log extraction, it becomes clearer which parts of the extraction process can be automated (Stein Dani et al., 2022). Automated event log extraction has the potential to save time and reduce the risk of human error.

Stein Dani et al. (2023) automatically matches an existing process model to tables in a database using natural language processing (NLP) techniques based on various syntactic similarity scores such as the Levenshtein distance and Jaccard index.[2] Matches are then evaluated against the author's gold standard mappings using the F_1 score. Stein Dani et al. (2023) state that although automated matching between different representations is common in research areas outside of process mining, they are the first to perform automated matching between data and a process model.

Whereas Stein Dani et al. (2023) is interested in matching activities of a process model to database tables, this thesis is interested in matching activities to columns contained within a single table. As suggested by Stein Dani et al. (2023), instead of matching on syntactic similarity, semantic similarity scores such as using the transformer neural network architecture (Vaswani et al., 2017) could potentially be used to improve performance.

A literature review on event log extraction by Dakic et al. (2020) shows that 80% of the proposed methods "require extensive programming knowledge". The authors conclude that only three of the methods reviewed (Buijs et al., 2010; González López de Murillas et al., 2019; Rodrıguez et al., 2012) have the potential to be used on any generic dataset. However, these methods still require significant manual input from domain experts.

2.4.2 Event Detection

Event detection, also referred to as anomaly detection or outlier detection, has been extensively applied to log data (Landauer et al., 2023; Papers With Code, 2024; Zhang et al., 2019). It focuses on finding events that are different from regular, expected behaviour and flow in data. The events can be anomalies, incidents, or any other data entry or log different from standard operation. Event detection is most relevant for processes that describe incidents or abnormalities.

Often, event detection methods perform some kind of classification and require labelled data to train a model. Since for this research, events are not yet identified in the dataset, supervised learning methods are not suitable.

A straightforward way to find events that does not require labelled data is to check if conditions relevant to the event are met. This can be done via a rule-based system (RBS) and is often referred to as an expert system (Preece et al., 1992). If the conditions are met, the event is found. If the conditions are not met, the event is not found. This would mean that for each event, a set of conditions needs to be defined, either by a domain expert or by an algorithm.

Many RBSs consist of two parts: (1) a rule base or knowledge base and (2) an inference engine (Masri et al., 2019). The knowledge base is a representation of specific domain knowledge via

[2] Both of which will be introduced in subsection 2.4.3.

rules by which events can be found. An inference engine can derive new facts or rules from an existing knowledge base and add these to or remove facts from the knowledge base (Rattanasawad et al., 2018). The inference engine takes actions based on input data and the rules defined in the knowledge base.

Unsupervised approaches to event detection include Lin et al. (2016) for instance. The authors cluster logs to make the identification of anomalies easier via the use of a knowledge base that checks for reoccurring log sequences. Xu et al. (2009) clusters anomalies via Principal Component Analysis. The proposed method is however designed for raw and unstructured textual console logs. Du et al. (2017) uses LSTMs to artificially generate the next log based on previous logs and compare this to the actual next log. If the difference between the two is too large, an anomaly is detected. Lu et al. (2019) clusters log traces of hospital data using sample patient data provided by domain experts and frequent sequence patterns.

Clustering algorithms will require additional postprocessing to link retrieved clusters to specific events or processes and require some sort of human input. The processes in this thesis do not only cover anomalies or incidents but also describe day-to-day operations. As such, algorithms that can only detect anomalies are not suitable. These algorithms should also alert on common, regular events.

2.4.3 Sequence Matching

Finding the best match of a sequence within a set of target sequences is a common problem in computer science. It is often referred to as (approximate) string matching (Hall & Dowling, 1980; Van der Loo et al., 2014) or calculating string similarity (Alberga, 1967; Cheatham & Hitzler, 2013). In essence, a string consisting of characters and a process consisting of activities are both an ordered sequence of elements.

This section describes several alternatives to process mining to match a retrieved activity sequence to the correct process. Van der Aalst (2010, p. 1) however states that "(...) simple techniques such as sequence mining are unable to capture the underlying process adequately". Although maybe not among the best performers, sequence matching algorithms are still relevant as they can be used as a baseline to compare the performance of various process mining algorithms. As the algorithms described are relatively simple, their interpretability and ease of implementation are also beneficial for this thesis.

A popular way to find the best match is to calculate the longest common subsequence (LCS) between a query sequence and a database with target sequences. The target sequence corresponding to the LCS with the most elements can be regarded as the best match. The LCS is the longest, ordered, sequence of elements that is present in both input sequences. The elements of the LCS do however not need to be consecutive in the input sequences, as long as they are in the same order. For instance, the LCS of two sequences (T, R, A, I, N, S) and (T, R, A, M, S) is (T, R, A, S). Many LCS algorithms have a time complexity of $O(n^2)$ or $O(n \log n)$ (Hunt & Szymanski, 1977).

The problem of sequence matching is often described as the minimum number of elements that need to be changed to transform one sequence into another. This is often referred to as the edit distance (Cormode & Muthukrishnan, 2007) or the Levenshtein distance (Levenshtein et al., 1966). It has been used for spelling checkers (Chaabi & Ataa Allah, 2022; Santoso et al., 2019) and optical character recognition (OCR) (Haldar & Mukhopadhyay, 2011), amongst other use cases. Possible operations to transform one sequence into another include:

- Insertion of an element
- Deletion of an element
- Substitution of an element

For instance, to transform the sequence (T, R, A, I, N) to (T, R, A, M), one deletion (I) and one substitution $(N \rightarrow M)$ is needed. The Levenshtein distance between these two sequences is thus 2. The lower the distance, the better the match between two sequences. Later improvements allow for the transpositions of two elements (Damerau, 1964) or add a weighting mechanism to favour similar elements (Haldar & Mukhopadhyay, 2011). In the case of OCR, similar-looking elements

can be the characters 'U' and 'V' (Haldar & Mukhopadhyay, 2011). Concerning this thesis, similar elements can be activities that have similar semantics. Cormode and Muthukrishnan (2007) propose an improvement by approximating the distance and adding the possibility of moving complete subsequences from one position to another. This algorithm has a time complexity of $O(n \log n)$.

Other popular methods to calculate the similarity between two sequences include the Hamming distance (Hamming, 1950) and the Jaccard index (Jaccard, 1912). The Hamming distance is the number of positions at which two equal-length sequences differ. This algorithm however only works for sequences of the same length. The Jaccard index or the 'coefficient of community' does have support for unequal-length sequences. It is defined as the intersection of two sets divided by the union of the two sets. The resulting similarity score falls within the range of $[0, 1]$. A value of 0 means that the two sets are completely dissimilar, while a value of 1 means that they are identical. As the Jaccard index utilises set operations, it does not capture the characteristics of sequences with duplicate elements or where order is important. For instance, the Jaccard index of the sequences (A, B) and (A, B, A, B, A, B) is 1 even though one sequence has three times as many elements as the other.

2.5 Research Gap

Despite advancements in process mining and event detection, a significant gap remains in the automation of event log data extraction from generic tabular data sources. Many software products, such as ProM (Van Dongen et al., 2005) and Disco (Günther & Rozinat, 2012), provide excellent ready-to-go user interfaces for analysing event log data using process mining. However, they assume that the event log data itself is readily available, making existing process mining techniques such as conformance checking not directly applicable to generic tabular data.

Consequently, the development of algorithms and interfaces that can easily convert generic tabular data into event log data has been minimal, often requiring substantial manual input and domain-specific expertise. The Process Mining Manifesto, written by the author of the seminal work on process mining, further highlights this issue by stating that "process mining is a relatively new paradigm and most of the currently available tools are still rather immature" (Van der Aalst et al., 2012).

For the Dutch railway infrastructure manager ProRail, there is a critical need for automated, real-time monitoring of operational processes to enhance safety. ProRail currently cannot automatically detect and monitor the execution of operational processes without extensive manual work, causing a blind spot for data-driven insights into these processes.

To address this gap, this thesis proposes a comprehensive methodology by:

- Formalising a methodology to convert generic tabular data into event log data.
- Formalising operational processes and their activities as process models.
- Implementing a PoC to extract activities from data using the defined formalisms
- Matching extracted activity sequences to predefined process models using various algorithms.
- Assessing the performance of the PoC using quantitative metrics and domain expert annotations.

By automating the annotation of log data as much as possible, this approach allows domain experts to focus on more complex and ambiguous cases where the PoC is not able to automatically annotate the data correctly. This assists in reducing the time and effort required to transform tabular data into event log data, making process mining techniques more accessible to a wider audience.

2.6 Ethical Considerations

This section discusses the ethical considerations made for the thesis.

2.6.1 Automation Bias

According to Goddard et al. (2011), automation bias is the tendency to over-rely on automation. This can lead to not questioning decisions and advice of an AI system, resulting in accidentally agreeing to incorrect or suboptimal decisions.

AI-suggested improvements to the processes described in the case study should be critically evaluated before being implemented, especially considering the extremely safety-critical nature of ProRail's operations. The potential for automation bias should be considered when evaluating these improvements. The PoC should be used as a tool to aid the identification and refinement of process models, not as a replacement for the domain experts who create and improve the processes.

2.6.2 Implicitly Evaluating Employee Performance

When investigating if processes are executed correctly, it is important to consider the ethical implications for the various actors involved. If the PoC detects that certain employees are deviating from a process description, it is important to consider the potential consequences of this information.

Possible consequences involve reprimanding or even firing the employee. The PoC could also be used to improve the performance of the employee by providing feedback on how to execute the process correctly. The PoC could of course be wrong in its assessment of the employee's performance, which could lead to unfair consequences for the employee.

Possible mitigations include the anonymisation of data used by the PoC. Hence making it impossible to trace the data back to a specific employee. Having a human-in-the-loop, where the PoC's suggestions are reviewed by a human before any action is taken, is another possible mitigation. This would allow for the human to (hopefully) correct any mistakes made by the PoC. One needs to be aware that the problem of automation bias, as discussed in subsection 2.6.1, could still occur in this case.

2.6.3 Environmental Impact

Training and evaluating AI models can have a significant environmental impact as it often requires large amounts of computational resources and thus electricity (Lacoste et al., 2019). Following the philosophy of Occam's razor (Duignan, 2024), of two algorithms that perform equally well, the simpler algorithm should be chosen. A simpler (but equally performant) algorithm is likely to have a smaller environmental impact.

2.6.4 Data Breach

As large amounts of data are used during this thesis, there is a risk of an unintentional data breach. Caution should be taken to ensure that no sensitive data is breached during the research. Risks emerging from a data breach include reputation damage, legal consequences, and financial loss to ProRail. Possible mitigations include anonymising data used in this thesis and ensuring that data is stored securely and locally.

2.6.5 Ethics and Privacy Quickscan

The Ethics and Privacy Quick Scan of the Utrecht University Research Institute of Information and Computing Sciences was conducted (Utrecht University [UU], 2023). It classified this research as low-risk with no fuller ethics review or privacy assessment required.

3 Methodology

Chapter 3 describes the various steps taken to answer each research question. These steps include the data collection and the development of the PoC along with various formalisations and definitions. The chapter concludes with the methods used to evaluate the PoC.

3.1 What data sources provide predictive information about the user processes?

For this sub-question, the research focuses on formalising a general format for tabular log data. A proper data formalisation provides a general approach for future research on event log extraction. We furthermore describe the process of selecting the most suitable data source for the PoC of the formalisms.

3.1.1 Data Formalisation

Let us formally define the input dataset as a sequence of two-dimensional matrices $\mathcal{M} = [M_1, M_2, \ldots, M_i]$. Each matrix M_i has dimensions $j \times k$, where j is the number of rows and k is the number of columns. Each matrix is the event log of a process instance whose activities are not yet identified. Each $M_i \in \mathcal{M}$ has the same number of columns k but can have a different number of rows j.

Each row of a matrix is a feature vector $F_j = [f_{j,1}, f_{j,2}, \ldots, f_{j,k}]$ and can be seen as a single event in the event log. The feature vectors are ordered in time, F_1 being the first row and F_j the last row of a matrix.

The k columns of a matrix M_i are represented as $C(M_i) = \{c_1, c_2, \ldots, c_k\}$. The descriptions of the columns are represented as $D(\mathcal{M}) = \{d(c_1), d(c_2), \ldots, d(c_k)\}$, where $d(c_k)$ is the textual description of the k-th column in each matrix. The column descriptions define the attributes of the event log, such as the resources, timestamps, and other data elements defined in subsection 2.4.1.

An example of a matrix M_i can be seen in Eq. (3.1).

$$M_i = \begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,k} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,k} \\ \cdots & \cdots & \ddots & \cdots \\ f_{j,1} & f_{j,2} & \cdots & f_{j,k} \end{bmatrix} \tag{3.1}$$

Each feature $f_{j,k}$ is a value stored in column c_k at timestep j in a matrix. The value $f_{j,k}$ can be of any type, such as numerical, categorical or textual.

Since activities are not yet found in the dataset, their associated processes can also not yet be identified. It is therefore possible for a single row to contain zero to ∞ activities. This allows a matrix to contain zero to ∞ activities and thus zero to ∞ processes. Ideally, a process will not start until the previous process has finished. However, this cannot be guaranteed as the data is not yet enriched with annotated activities and thus processes.

3.1.2 Data Selection

For practical and computational reasons, the PoC does not use all of the data available in Sherlock. A subset of the available data is selected based on the criteria defined below.

- The data must cover the period from 1 September 2023 to 31 December 2023.
  - As during this period the project *'Ervaringsrijden'* has been active in which ETCS has been used (subsection 2.1.3).
- The data may only include the 22 DRPs mentioned in Appendix 6.4.

  - As these DRPs cover the entire route between Asd and Ut.
  - Parts of a train journey outside this subset of DRPs are deemed irrelevant.
- Only train journeys are included for which ETCS was enabled during the passage of at least one of the DRPs mentioned in Appendix 6.4.

3.2 How can the description of a user process and its activities be encoded into a formal definition?

GPs are described in natural language and are visualised using sequence diagrams. In order to better understand the flow of these processes and to identify recurring activities, the sequence diagrams of the processes are converted into a structured format. BPMN diagrams are the format of choice to visualise process models for this thesis. Converting sequence diagrams into BPMN diagrams is the first step in formalising the GPs. They help to (1) identify similar activities across the different GPs and (2) capture alternative flows that are currently only described in the accompanying notes of ProRail AM (2023).

A subset of the forty GPs is used for the PoC. The three processes below are chosen in collaboration with domain experts:

- GP-3: Departure with unknown train position
- GP-5: Passing a stop signal without MA
- GP-9: Turning or reversing a train

3.2.1 Formalising Activities Across Process Models

A unique type of activity can be part of zero to ∞ process models. For example, the activity 'Inschakelen stuurstroom', Dutch for 'Switch on control current', is present in eight different processes. Ideally, we can easily identify the same activity across all eight process models and use a consistent notation. This is not possible with the sequence diagrams as activities are inconsistently described in natural language. Furthermore, activities are not guaranteed to be consistent across the different sequence diagrams as activities are not labelled with a unique ID.

Each activity consists of the interaction from one resource to another. For example, the aforementioned activity 'Switch on control current' is an interaction of the train operator to the system. We label the activity as MS_{id}, where M stands for 'train operator', S stands for 'system' and id is the unique identifier for this specific activity. Semantically equal activities are numbered the same across all process models. Following this reasoning for all possible combinations of resources, the scheme in Table 3.1 is created.

		Receiving Resource		
		Train operator (machinist)	System (systeem)	Traffic controller (Trdl)
Sending Resource	Train operator (machinist)	MM	MS	MT
	System (systeem)	SM	SS	ST
	Traffic controller (Trdl)	TM	TS	TT

Table 3.1: Scheme to model the interaction between different resources. The initiating resource is on the left, the receiving resource is on the top. The labels are a combination of the first letter of the initiating resource and the first letter of the receiving resource.

3.2.2 Formalisation of a User Process

Based on the descriptions of ProRail AM (2023), as elaborated in section 2.2, and in the literature study of section 2.4, a GP is defined to have the following properties:

- A GP is defined as the combination of a sequence of activities and a set of prerequisites.
- A sequence of activities consists of activities defined in Table 3.1. It is ordered in time.
  - E.g. $MS_1 \rightarrow ST_4 \rightarrow MS_2 \rightarrow TM_2 \rightarrow \dots$
- Each GP has a set of prerequisites.
  - E.g. {*'The train is driving in L2'*, *'The position of the train is unknown'*}
- Each GP has a unique identifier.
  - E.g. GP-1, GP-2, GP-3, ...
- The combination of the sequence of activities and set of prerequisites is unique to each GP.
  - I.e. no two GPs have the same sequence of activities *and* the same set of prerequisites.

More formally, each GP, denoted as G_{id}, has a corresponding ordered sequence of activities, denoted as A_{id} and a set of prerequisites, denoted as P_{id}. The id of any G_{id} is a positive integer where id is the unique identifier of the process model.

If $id \in \mathcal{D}$, the id falls within the domain of all unique identifiers of the GPs described in ProRail AM (2023). Since ProRail AM (2023) consists of forty GPs, $|\mathcal{D}| = 40$. Furthermore, $A_{\mathcal{D}}$ is the superset of all activities present in ProRail AM (2023) and $P_{\mathcal{D}}$ is the superset of all prerequisites in ProRail AM (2023).

3.2.2.1 Formalisation of a Sequence of Activities

A sequence of activities A_{id} of G_{id} is ordered in time. It is denoted as follows:

$$A_{id} = (a_1, a_2, \dots, a_{n-1}, a_n) \tag{3.2}$$

where n is the number of activities in G_{id}. A sequence of activities always has a length of $0 < n < \infty$.

An activity a_i refers to any of the activities that follow from the resource scheme in Table 3.1. For instance, $a_i = MS_1$ is a valid activity and $A_{id} = (MS_1, ST_4, MS_2, TM_2)$ is a valid sequence of activities for G_{id}.

Since a sequence of activities is ordered in time, a_i happens before a_j if $i < j$. It is thus not a set but rather a tuple or a sequence. Within the GPs in $\mathcal{D}$, the sequence of activities is unique to each GP. This is formalised as follows:

$$\forall x \forall y [(x \in \mathcal{D} \land y \in \mathcal{D} \land x \neq y) \rightarrow A_x \neq A_y \rightarrow G_x \neq G_y] \tag{3.3}$$

3.2.2.2 Formalisation of a Set of Prerequisites

The set of prerequisites P_{id} belonging to G_{id} is a set of propositions. A proposition is a statement that is either true or false. The prerequisites are denoted as follows:

$$P_{id} = \{p_1, p_2, \dots, p_{n-1}, p_n\} \tag{3.4}$$

where p_i is the i-th prerequisite of G_{id} and n is the total number of prerequisites of G_{id}. Unlike the sequence of activities, the prerequisites are not ordered. It is thus defined as a set. Furthermore, the set of prerequisites can be empty. The empty set $P_{id} = \emptyset$ means that there are no prerequisites applicable for G_{id}. If any of the propositions in P_{id} are false, the corresponding process is automatically not applicable.

3.2.2.3 Equality of Two User Processes

Two processes are equal if and only if they have the same sequence of activities and the same set of prerequisites. This can be formalised as follows:

$$\forall x \forall y[(P_x = P_y \wedge A_x = A_y) \rightarrow (G_x \leftrightarrow G_y)] \tag{3.5}$$

Note that we do not take $\mathcal{D}$ into account in Eq. (3.5). The equation is useful to determine whether a retrieved sequence of activities and set of prerequisites from data is equivalent to a process from ProRail AM (2023). Currently, no two processes in $\mathcal{D}$ are equal.

3.3 How can the formal definition be used to extract a sequence of activities from data?

To find and annotate activities in data, we follow the process mining definitions in subsection 2.4.1 and the formalisations in subsection 3.1.1 and subsection 3.2.2.

As described in section 2.3, data is stored in a tabular form. Each row in the data represents all the information about one step of a particular train journey. If we extract the data for train 220 on Monday the 18$^{\text{th}}$ of December 2023, we retrieve all data from the start until the end of that train journey. An extract of this journey can be seen in Table 3.2.

Train Number	Track	DRP	DRP activity	Planned	Actual
220	5	Ut	Arrival	18-12-2023 10:59:42	18-12-2023 11:30:35
220	5	Ut	Departure	18-12-2023 11:02:30	18-12-2023 11:31:52
220	1	Utzl	Passage	18-12-2023 11:04:30	18-12-2023 11:33:52
220	801	Mas	Passage	18-12-2023 11:07:18	18-12-2023 11:36:22
220	AD1	Bkla	Passage	18-12-2023 11:09:54	18-12-2023 11:38:26
220	1	Bkl	Passage	18-12-2023 11:10:06	18-12-2023 11:38:37
220	AC1	Aco	Passage	18-12-2023 11:16:12	18-12-2023 11:43:41
220	671	Ac	Passage	18-12-2023 11:16:48	18-12-2023 11:44:14

Table 3.2: Adapted extract of the data from Sherlock, similar to Table 2.1. This excerpt contains the data of a train journey with train 220 on Monday the 18$^{\text{th}}$ of December 2023. The actual data contains more columns and rows than shown here.

Relating the above table to the formalisations of subsection 3.1.1, each matrix M_i represents a unique train journey i where each step of the journey is a row of that matrix. As noted in section 2.3, Sherlock consists of a total of 733 columns. Each row or feature vector is therefore of length $k = 733$. Example features, as shown in Table 3.2, include the train number, the DRP the train is at and the DRP activity. These features can change over time, the rows are thus ordered in time as defined in subsection 3.1.1.

3.3.1 Rule-Based System

To find and annotate all activities that have occurred within a matrix M_i, an RBS as described in subsection 2.4.2 is used. RBSs are a straightforward way of annotating data.

There are however some drawbacks to an RBS. Firstly, it is not very efficient. Even though simple and fast if-then statements are used, the dataset has to be iterated once for each activity to check if its conditions are met. This can be very computationally demanding, especially when the dataset is very large.

Secondly, the code to check all conditions will consist of a large number of if-then statements. This can be both a positive and a negative feature. The use of if-then statements makes the

reasoning of a system explainable to the user. The reasoning can also be compared and verified with the reasoning of domain experts, adding a layer of validation to the system. However, an RBS could be difficult to maintain and scale, particularly when the number of activities and their complexity increases.

The RBS is described in the pseudocode of Algorithm 1. The system can be used for any sequence of matrices $\mathcal{M}$ and any superset of activities $A_{\mathcal{D}}$. The system tries to identify a set of activities for each feature vector of a matrix $M_i \in \mathcal{M}$. The system automatically proceeds to the next activity a_{n+1} if any rule of activity a_n is false for the given feature vector. Thus, all rules for a given activity must be true for the activity to be retrieved. This increases the efficiency of the system as it reduces the number of rules that need to be checked. The rules for an activity a_n are defined as $R(a_n, M_i, F_j) = \{r_1(M_i, F_j), r_2(M_i, F_j), \ldots, r_m(M_i, F_j)\}$.

If all rules of an activity are evaluated as true, the activity is automatically appended to the list of retrieved activities A_{F_j} belonging to row F_j. In this way, the system can be used to find all activities for each row in a matrix M_i, stored as A_{M_i}. As a last step, the system returns a list of lists of retrieved activities $A_{\mathcal{M}}$ for all matrices $M_i \in \mathcal{M}$.

Algorithm 1 Rule-Based System

$\mathcal{M} \leftarrow [M_1, M_2, \ldots, M_i]$ ▷ Dataset as a sequence of matrices
$A_{\mathcal{D}} \leftarrow \{a_1, a_2, \ldots, a_{|A_{\mathcal{D}}|}\}$ ▷ All unique activities in the domain $\mathcal{D}$
$A_{\mathcal{M}} \leftarrow [\ldots]$ ▷ Empty list of retrieved activities for dataset $\mathcal{M}$
for all matrices $M_i \in \mathcal{M}$ **do**
    $A_{M_i} \leftarrow [\ldots]$ ▷ Empty list of retrieved activities for matrix M_i
    for all feature vectors $F_j \in M_i$ **do**
        $A_{F_j} \leftarrow \{\ldots\}$ ▷ Empty set of retrieved activities for row F_j
        for all activities $a_n \in A_{\mathcal{D}}$ **do**
            $R(a_n) \leftarrow$ all rules necessary to retrieve activity a_n from F_j
            for all rules $r_m(M_i, F_j) \in R(a_n, M_i, F_j)$ **do**
                if rule $r_m(M_i, F_j)$ is **true then**
                    continue ▷ Continue to next rule $r_{m+1}(M_i, F_j)$
                else
                    break ▷ Continue to next activity a_{n+1}
                end if
            end for
            Append activity a_n to A_{F_j} ▷ Since no rule of a_n is false for F_j
        end for
        Append A_{F_j} to A_{M_i} ▷ All activities for feature vector F_j
    end for
    Append A_{M_i} to $A_{\mathcal{M}}$ ▷ All activities for matrix M_i
end for
return $A_{\mathcal{M}}$ ▷ All activities for dataset $\mathcal{M}$

To determine whether a rule is true for a given feature vector F_j, data from other vectors may be required. It is therefore essential to provide the rule-checking function with the entire matrix M_i and not just the vector F_j. A rule $r_m(M_i, F_j)$ can thus be seen as a function that takes the entire matrix M_i and the target row F_j as input. This way, the rule-checking function can access all data of matrix M_i but is still focused on determining whether the rule is applicable for a particular vector F_j.

For the PoC, the rules associated with each activity in $A_{\mathcal{D}}$ are defined in collaboration with domain experts. Dataset $\mathcal{M}$ refers to the Sherlock dataset where each train journey is a matrix M_i.

Let us consider activity MS_{10} = *'Acknowledge mode OS'* as an example. This activity describes the timestamp at which the train operator acknowledges ETCS mode OS. The exact timestamp of this event is not stored in the data but can be approximated by identifying the column that

stores the current ETCS mode of the train. If the current ETCS mode is equal to OS and is different from the previous entry, we can conclude that the activity has occurred since the last feature vector. More formally, the rule can be defined as follows:

$$r_1(M_i, F_j) = \begin{cases} \text{True} & \textbf{if } F_{j,290} = \text{“OS”} \\ \text{False} & \text{otherwise} \end{cases}$$

$$r_2(M_i, F_j) = \begin{cases} \text{True} & \textbf{if } F_{j,290} \neq F_{j-1,290} \\ \text{False} & \text{otherwise} \end{cases}$$

$$R(MS_{10}, M_i, F_j) = \{r_1(M_i, F_j) \wedge r_1(M_i, F_j)\} \tag{3.6}$$

In this example, Eq. (3.6) assumes that column c_{290} stores the current ETCS mode of the train. Rulesets can be defined for all activities in the domain $A_{\mathcal{D}}$ in a similar way.

3.4 Which algorithm is best in identifying user processes in the retrieved sequences of activities using the formal definitions?

The activity sequences gathered from data using the methods described in section 3.3 are not necessarily exact matches to any of the activity sequences specified by the process models. This section investigates how to match a retrieved sequence to those of the GPs. Both sequence mining (subsection 2.4.3) and process mining (subsection 2.4.1) techniques are considered and discussed in this section.

Suppose that we want to check if G_{id} is applicable for a retrieved activity sequence $A_{\text{retrieved}}$. An ideal match would then be when $A_{\text{retrieved}} = A_{id}$, both in content as well as in order. The possibility of an exact match is however slim as there is a high chance not all specified activities are retrieved, due to data limitations for instance. Likewise, there is a high chance that not all retrieved activities for a matrix M_i belong to the same process as a matrix captures a multitude of events and not necessarily a single process. The goal is thus to find the best match for $A_{\text{retrieved}}$ from the three previously chosen process models.

3.4.1 Sequence Matching

Many different sequence matching algorithms can be used to compare two sequences, as discussed in section 2.4. The most promising sequence matching method investigated is the LCS algorithm as it takes the order of elements into account. The resulting subsequence found by the LCS algorithm furthermore generates valuable insights for future research into investigating the described processes.

Other methods researched, such as the Levenshtein distance, only result in a single number: the distance between two sequences. The Jaccard index furthermore does not capture duplicate elements or the order of elements and the Hamming distance only works for sequences of equal length. The LCS algorithm is thus the most informative and promising sequence matching method for this research. An example to find the best match for a retrieved sequence is given in Table 3.3.

As can be seen in Table 3.3, two of the example process models achieved the same rank and are thus considered to be equally good matches for $A_{\text{retrieved}}$. This tie is problematic since we want to determine the *single* best match. A tie indicates the inability of the algorithm to distinguish between process models. We annotate retrieved sequences as ambiguous in cases where the algorithm is uncertain or unable to determine a single best match.

This method is preferred over forcing the algorithm to strictly select from the three implemented process models for multiple reasons. Firstly, there is a chance that the process captured in the retrieved sequence is not one of the three chosen processes. This would force the algorithm to make

Process Model	A_{id}	LCS	Rank
GP-X	$A_X = (ST_5, MS_1, ST_3, MS_2, TM_2)$	(MS_1, MS_2, TM_2)	#1
GP-Y	$A_Y = (MS_1, ST_4, MT_3, TM_2)$	(MS_1, ST_4, TM_2)	#1
GP-Z	$A_Z = (ST_4, MS_3, TM_2, MS_1)$	(ST_4, TM_2)	#2

Table 3.3: Example of using the LCS method to find the best match for an example sequence $A_{\text{retrieved}} = (MS_1, ST_4, MS_2, TM_2)$. The rank is based on the length of the LCS. The longest LCS is considered the best match.

an incorrect decision as it cannot select the correct process model. Integrating a second algorithm as a tie-breaker would furthermore make it more difficult to investigate the results achieved by the LCS algorithm. The results would then be skewed by the results of the tie-breaker algorithm making it harder to interpret.

3.4.2 Process Mining

Conformance checking algorithms are used to investigate deviations between data and a process model. As we have a collection of process models to choose from rather than a single process model, we do not know which process model we should compare the data to, adding an extra layer of complexity.

The alignment method is not used for this research. The number of synchronous moves in the alignment of any two sequences is equal to the length of their LCS, making it too similar to the proposed method in subsection 3.4.1. For example, let Z be the LCS of sequences X and Y. Each element of Z refers to indices i and j in X and Y respectively, where $X[i] = Y[j]$. Each subsequent element in Z refers to indices k and l in X and Y respectively with $k > i$ and $l > j$, meaning that the order of elements is preserved. Again, $X[k] = Y[l]$ means that another synchronous move has occurred. Since $X[i] = Y[j]$ holds for every element in Z, the number of synchronous moves is equal to the length of Z. Thus, no additional information is gained by using the alignment method as opposed to the LCS method.

We however compare the LCS to the TBR algorithm. A retrieved sequence of activities is replayed on the Petri net of each process model. The pair with the highest fitness_{PN} score, using Eq. (2.3), is then considered the best match. In the case of a tie, we annotate the retrieved sequence as ambiguous for the same reasons as discussed in subsection 3.4.1.

The Petri net of a process model is constructed by utilising the formal definitions of a GP, as established in subsubsection 3.2.2.1. Each activity a_i in the ordered sequence A_{id} is represented by a transition t_i in T_{id} with $T_{id} = \{t_1, t_2, \ldots, t_n\}$ where $n = |A_{id}|$. Places are inserted for each transition, this includes before the first and after the last transition. This results in a set of places P_{id} such that $P_{id} = \{p_1, p_2, \ldots, p_m\}$ where $m = |T| + 1$. The input and output functions for each t_i in T_{id} are defined as $I_{id}(t_i) = \{p_i\}$ and $O_{id}(t_i) = \{p_{i+1}\}$ respectively. The Petri net of G_{id} is thus defined as $PN_{id} = (P_{id}, T_{id}, I_{id}, O_{id})$. Petri nets of each process model are visualised in Appendix 6.5.

3.5 What evaluation metric is best to determine the performance of the system?

In this section, we discuss various evaluation metrics used to assess the performance of the PoC. We also discuss the reasoning behind the choice of these metrics.

Determining proper evaluation metrics to assess the performance of a system is important for multiple reasons. Firstly, an improper metric can lead to a biased estimate of a system's performance. An overestimation of the performance of a system can lead to a false sense of security and trust, as discussed in subsection 2.6.1.

Secondly, the choice of evaluation metric can influence the development of a system. If the evaluation metric is not properly chosen, a system could be optimised for the wrong goal. This can lead to an overfitted system that performs well according to the chosen metric, but poorly in real-world scenarios. The metric of choice should thus be aligned with the world it is trying to represent and the goals of the stakeholders. A proper evaluation metric furthermore helps to set a baseline for future research. An important factor for this is to make the evaluation process reproducible and transparent.

3.5.1 Quantifying Matching Algorithms

We compare the performance of the LCS algorithm to the TBR algorithm following the fitness[1] and precision definitions of Eq. (2.1) and Eq. (2.2), discussed in subsubsection 2.4.1.3. We define $|L \cap M|$ as the number of synchronous moves of two sequences, as implemented via the alignment algorithm of subsubsection 2.4.1.3. This way, $|L \cap M|$ is equal to the number of activities that are perfectly aligned and captures the shared behaviour between the retrieved and actual sequence. Furthermore, we define $|L|$ as the length of the retrieved sequence. Since only a single path exists for each of the three currently implemented process models, $|M|$ is defined as the number of activities in the process model. Once a process model is extended to include multiple paths, $|M|$ will be defined as the number of activities of the traversed path in the process model.

By definition, $|\emptyset| \leq |L \cap M| \leq |L|$ and $|\emptyset| \leq |L \cap M| \leq |M|$ meaning that both measures have a range of $[0, 1]$. This makes the results easily understandable for the stakeholders, as they are intuitive and easy to interpret. This method also makes it possible to compare the performance of the two algorithms generically, as fitness and precision are calculated in the same way for both algorithms.

As mentioned in subsubsection 2.4.1.3, an optimal algorithm should have both a high fitness as well as a high precision. Similar to the well-known F_1-score within the field of classification (Chinchor & Sundheim, 1993; Naidu et al., 2023; Stein Dani et al., 2023; Van der Aa et al., 2017), we use the harmonic mean of fitness and precision to quantify the performance of the algorithms into a single number:

$$\mathcal{F}_1 = 2 \cdot \frac{\text{Fitness} \cdot \text{Precision}}{\text{Fitness} + \text{Precision}} \tag{3.7}$$

One needs to be cautious, as overfitting can still occur which leads to bad real-world performance, as presented in Figure 2.5.

For each of the two matching algorithms, we calculate the fitness, precision and $\mathcal{F}_1$-score for every retrieved sequence and their matched process model. We then calculate the macro-averaged fitness, precision and $\mathcal{F}_1$-scores per process model. This way, we obtain a mean fitness, precision and $\mathcal{F}_1$-score per process model, per algorithm.

We report the macro-average $\mathcal{F}_1$-score per algorithm by calculating the arithmetic mean of the three mean $\mathcal{F}_1$-scores. A macro-average $\mathcal{F}_1$-score is chosen over a micro-average $\mathcal{F}_1$-score as it gives equal weight to each class (Opitz & Burst, 2019). This is important due to class imbalance as the matched sequences of activities might not be equally distributed over the process models.

Retrieved sequences classified as ambiguous are not included in this evaluation as they are not matched to a process model and we want to determine the performance per process model. For each algorithm, we disclose the number of sequences classified as ambiguous.

3.5.2 Statistical Testing

The best algorithm is determined using the two-tailed paired t-test with a significance level of $\alpha = 0.05$. We test the difference between the two algorithms based on the distribution of the macro-average $\mathcal{F}_1$-score. This is again to ensure that each process model has equal weight as the

[1] Please note that this definition of fitness differs from the definition of fitness_{PN} in Eq. (2.3).

2,506 retrieved sequences are most likely not uniformly distributed over the three process models. The null and alternative hypotheses are defined as follows:

- H_0: There is no significant difference between the macro-average $\mathcal{F}_1$-score of the LCS algorithm and the TBR algorithm.
- H_1: There is a significant difference between the macro-average $\mathcal{F}_1$-score of the LCS algorithm and the TBR algorithm.

If the null hypothesis is rejected, we can conclude that there is a significant difference between the two algorithms. The algorithm with the highest macro-average $\mathcal{F}_1$-score is then chosen as the best algorithm.

3.5.3 Domain Expert Evaluation

To further investigate the difficulty of the problem at hand, two domain experts are tasked with annotating a randomly selected subset of the data used for this thesis. Both domain experts are provided with the same subset of train journeys and are tasked with determining which process model is most suitable for each train journey. The experts are permitted to discuss their results with each other to determine the best-fitting process model. This is intended to simulate how they would perform the task if it were part of their regular duties, without the help of an AI system.

We report the inter-annotator agreement (IAA) between both annotators, as well as the IAA between each algorithm-annotator pair and the IAA between the two algorithms. The IAA is a concept to determine the level of agreement between two annotators. For this, we use Cohen's Kappa (κ) coefficient to determine the IAA. Cohen's Kappa coefficient is popular within multiple research fields including healthcare (Bonnyman et al., 2012; McHugh, 2012), psychology (Min et al., 2023) and NLP (Artstein & Poesio, 2008; Dahlmeier et al., 2013).

This measure is a better indicator than the percentage of agreement as it corrects for the agreement that can be expected by chance (Cohen, 1960). Cohen's Kappa coefficient is defined as follows:

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{3.8}$$

Here, p_o is the proportion of agreement observed and p_e is the proportion of agreement expected by chance. This means that $p_o - p_e$ is the proportion of agreement that is not due to chance (Cohen, 1960, p. 40).

Cohen's Kappa coefficient ranges from -1 to 1. A κ of 1 indicates perfect agreement. A value $0 \leq \kappa < 1$ indicates agreement better than chance. A value of $\kappa = 0$ is an agreement equal to chance and a value of $-1 \leq \kappa < 0$ is an agreement worse than chance.

A popular interpretation of the κ coefficient is that of Landis and Koch (1977). This interpretation is shown in Table 3.4. The domain experts are experts in their field and are thus expected to have a high level of agreement.

Kappa Statistic	**Strength of Agreement**
< 0.00	Poor
0.00 – 0.20	Slight
0.21 – 0.40	Fair
0.41 – 0.60	Moderate
0.61 – 0.80	Substantial
0.81 – 1.00	Almost Perfect

Table 3.4: Interpretation of Cohen's Kappa coefficient by Landis and Koch (1977, p. 165).

If an algorithm classifies a sample as ambiguous, we report the estimated probability for each of the three process models. This is to be more insightful than just reporting that the sample is classified as ambiguous. The probability is calculated by applying the softmax function to the score calculated for each process model. For the LCS algorithm, the scores are the lengths of the LCS of each process model. For the TBR algorithm, these are the achieved $fitness_{PN}$ scores. The softmax function normalises a collection of numbers such that its sum becomes 1. Each value is then interpreted as the probability of the corresponding process model being the best fit. The softmax function is defined as follows:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{3.9}$$

Here, x_i is the score for process model i. The total number of scores is defined as n, three in our case. As we use this method to investigate ambiguous classifications, multiple process models will achieve an equal maximum probability. I.e., if one process model outperforms all others, the sample would not have been classified as ambiguous in the first place.

4 Results

Chapter 4 presents the results obtained for each of the sub-questions. The results were obtained through the literature review in the theoretical framework and the methodology constructed in Chapter 3.

4.1 What data sources provide predictive information about the user processes?

As mentioned in subsection 2.1.3, only a limited number of trains run under the ETCS regime. A relatively small amount of data is thus available compared to traditional ATB-trains. On average, 270.7 events ($SD = 72.9$) involving 20.5 ETCS train journeys ($SD = 5.8$) occurred daily on the track between Amsterdam and Utrecht between the 1st of September 2023 and the 31st of December 2023. An average train journey consists of 13.2 events ($SD = 2.3$). 33,025 events covering $|\mathcal{M}| = 2,506$ matrices were recorded in total. The number of events and train journeys per day can be seen in Figure 4.1.

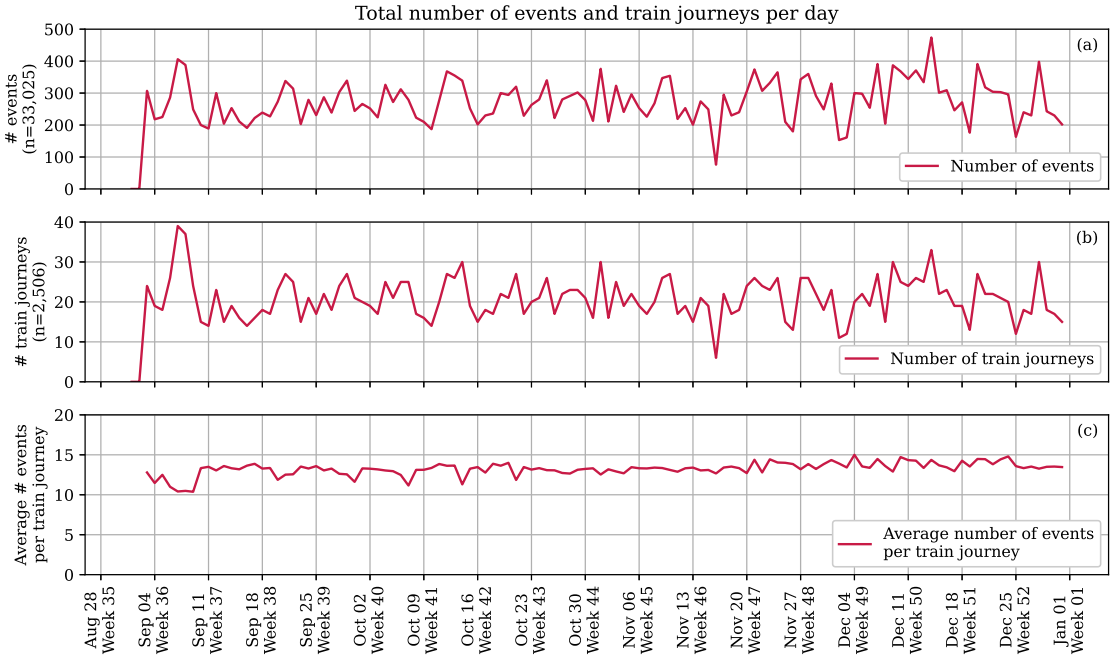

Figure 4.1: Total number of events (a) and journeys (b) per day covering trains driving (parts of) their journey under ETCS. Subfigure (c) shows the mean number of events per train journey per day. Vertical grid lines indicate the Monday of each week.

The distribution of the number of events and train journeys is further visualised in the box plots in Figure 6.5 of the appendix. In addition, the distribution among the DRPs is visualised in Figure 6.6 of the appendix. Data for the 1st and 2nd of September 2023 is missing, as track maintenance was carried out on these days. On the 16th of November 2023, German train drivers went on strike causing a significant reduction in international train traffic.

Descriptions of the $k = 733$ columns include $d(c_3) =$ *'basic.drp: Dienstregelpunt'*, which captures the current DRP. The feature value is a string and can be any single of the 22 DRPs

mentioned in Appendix 6.4. Other examples include c_{11} which is the planned time of the event, c_5 which is the actual time of the event and c_{290} which is a list of ETCS modes the train has been in since the last event.

4.2 How can the description of a user process and its activities be encoded into a formal definition?

Following the formal definition of a process and its activities in section 3.2, all activities of the three selected process models are manually annotated with a unique label. The annotated activities, grouped per process model, can be found in Appendix 6.7. A summary of the distribution of activities among the three selected processes is shown in Table 4.1.

Process Model	# Unique Activities	Total # of Activities
GP-3	23	32
GP-5	14	14
GP-9	18	19

Table 4.1: Number of unique activities present in each of the three process models and the total number of activities per process model.

GP-3 contains both the most activities as well as the most unique activities, 32 and 23, respectively. Equal activities in different processes are assigned the same ID. For example, the activity *'Wijzigt dan wel bevestigt driver-id en treinnummer'* of GP-3 and *'Bevestigt of wijzigt driver-id en treinnummer'* of GP-9 are both assigned to activitiy MS_2. Both describe the same activity *'Confirms or changes driver id and train number'* and thus have the same identifier. Merging similar activities across processes is done in agreement with domain experts. BPMN diagrams of the three process models can be found in Appendix 6.8.

4.2.1 Annotation results

In total, the three process models consist of 65 activities and are manually grouped and ID'ed into 43 unique activities. Their distribution can be seen in Figure 4.2(a). The distribution of the six identified combinations of initiating and receiving resources can be seen in Figure 4.2(b). Activity types SS, MM and TT are not present. The most common combination is SM, which covers 28(43%) of the 65 activities. The MT and TM combinations both only occur once.

The distribution of the initiating and receiving resources can be seen in Figure 4.2(c) and Figure 4.2(d), respectively. The system is both the most common initiating and receiving resource, with 33 and 30 occurrences, respectively. The Trdl is the least common initiating and receiving resource.

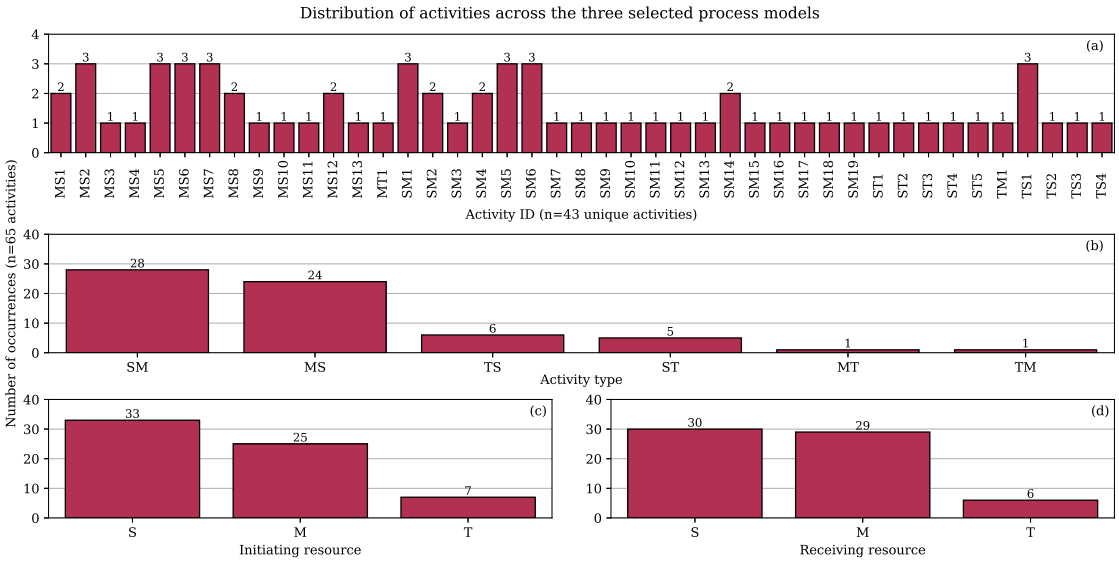

Figure 4.2: Distributions of (a) the 43 unique activities identified, (b) the six combinations of initiating and receiving resources, (c) the initiating and (d) receiving resources.

4.3 How can the formal definition be used to extract a sequence of activities from data?

The activities and their associated rules are implemented in collaboration with domain experts. The rules for each activity have been finetuned until an agreement was reached with domain experts. Due to data limitations, it was not possible to implement all activities. In total, 27 (63%) of the 43 activities have been implemented. All rules of the RBS can be found in Appendix 6.9. All implemented activities can be found in Appendix 6.10.

The remaining 16 activities (37%) belong to the Start of Mission (SoM) or the stoptonend sein (STS) route procedure. The SoM entails all the starting procedures that need to be carried out before the train can start its journey. The data captured during SoM activities is only stored locally on the train and is therefore not present in the dataset used for this research. The STS-route, Dutch for 'stop signal route', is a route granted by the Trdl that the train operator can take to pass a red signal. STS-route information is also not present in the data.

In total, 73,591 activities have been identified in the dataset. The activities are present in 29,234 of the 33,025 events. Activities have not been found for 3,791 events. Only one of the 2,506 matrices did not contain any activities. 19,598 events included more than one activity. The distribution of the number of activities per event can be seen in Figure 4.3(a).

The top five most common sets of activities per event are shown in Figure 4.3(b). The most common set is $\{SM_{15},\}$, which is present in 9,156 events. The most common set with more than one activity is $\{SM_{15}, ST_1, TS_1\}$ with 8,038 occurrences.

Figure 4.4 is similar to Figure 4.2, but now shows the distribution of the activities retrieved from data. The distribution of the six unique activities is shown in Figure 4.4(a). The most common activity is SM_{15}, which is present in 23,815 events. MS_{11} and SM_{12} are the least common activities, with two occurrences each.

Figure 4.4(b) shows the distribution among the retrieved pairs of initiating and receiving resources. The most common pair is SM, which covers 36,594 activities. The least common combination is MT, which occurs 30 times.

The system is the most common initiating resource, with 54,455 occurrences, as shown in Figure 4.4(c). The train operator is the most common receiving resource, with 36,658 occurrences,

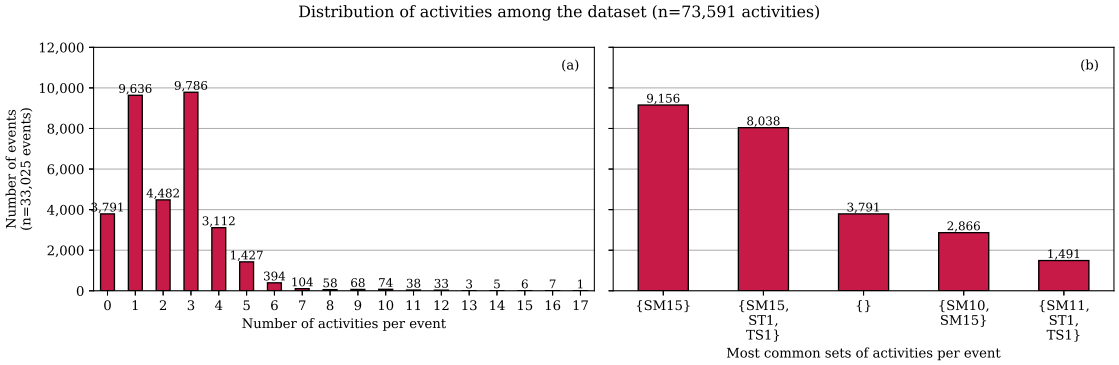

Figure 4.3: (a) Distribution of number of activities per event. (b) Distribution of the top five most common sets of activities per event.

as shown in Figure 4.4(d).

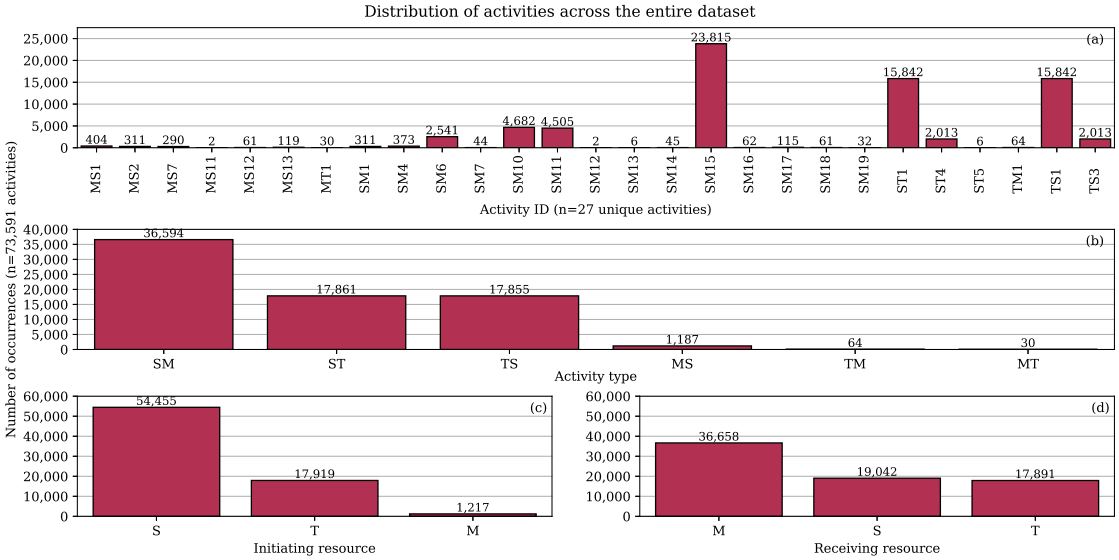

Figure 4.4: Distribution of (a) the six unique activities retrieved from data, (b) the pairs of initiating and receiving resources, (c) the initiating resources, and (d) the receiving resources.

4.4 Which algorithm is best in identifying user processes in the retrieved sequences of activities using the formal definitions?

The results of the sequence matching algorithm are presented in subsection 4.4.1. This is followed by the results achieved by the process mining algorithm in subsection 4.4.2.

4.4.1 Sequence Matching

The LCS algorithm can identify a single best match for 1,156 of the 2,506 matrices. The remaining 1,350 matrices are classified as ambiguous, as multiple best matches are found for these matrices.

Process Model	Count	Mean LCS length	SD of LCS length
GP-3	1,110	5.088	1.780
GP-5	0	N/A	N/A
GP-9	46	4.652	1.464
Ambiguous	1350	3.529	0.708
Total	2,506	4.240	1.518

Table 4.2: Table summarising the number of matches divided across four categories. The Standard Deviation and mean LCS length per category are also shown. The mean and SD across all 2,506 matrices is shown in the last row.

As can be seen in Table 4.2, GP-3 is the most common among the three process models, with 1,110 matches. Most matrices are however classified as ambiguous, with a mean LCS length of 3.529. No matches are found for GP-5. Of the three process models, GP-3 has the longest mean LCS and biggest SD. A visualisation of the distribution of the length of the LCSs per process model can be found in Figure 6.10 of the appendix.

4.4.2 Process Mining

For 2,448 of the 2,506 matrices, the TBR algorithm is able to determine a single best match. No best match is found for the remaining 58 matrices.

Process Model	Count	Mean fitness$_{PN}$ score	SD of fitness$_{PN}$ score
GP-3	2,344	0.478	0.059
GP-5	20	0.399	0.070
GP-9	84	0.411	0.052
Ambiguous	58	0.092	0.161
Total	2,506	0.466	0.086

Table 4.3: Table summarising the number of matches divided across four categories. The Standard Deviation and mean fitness$_{PN}$ score per category are also shown. The mean and SD across all 2,506 matrices is shown in the last row.

As can be seen in Table 4.3, GP-3 has both the most matches and the highest mean fitness$_{PN}$ score. GP-5 achieved the lowest mean fitness$_{PN}$ and the lowest number of matches of the three process models. A total of 43 of the 58 matrices annotated as ambiguous scored a fitness$_{PN}$ of 0.0. A visualisation of the distribution of the fitness$_{PN}$ score per process model can be found in Figure 6.11 of the appendix.

4.5 What evaluation metric is best to determine the performance of the system?

For RQ4, all 2,506 matrices have been classified by the LCS algorithm and the TBR algorithm. As proposed in subsection 3.5.1, the macro-averaged $\mathcal{F}_1$-score is used to compare the classifications of the two algorithms.

The results achieved for the LCS and TBR algorithms are shown in Table 4.4 and Table 4.5, respectively. The LCS algorithm achieves a macro-averaged $\mathcal{F}_1$-score of 0.120 ± 0.087. Since no

matrices were classified as GP-5, fitness and precision could not be calculated for this process model. We thus regard the mean $\mathcal{F}_1$-score of GP-5 as 0.0. GP-9 achieves a higher mean $\mathcal{F}_1$-score of 0.202 than GP-3 while also having a lower spread.

Process Model	Mean Fitness	Mean Precision	Mean $\mathcal{F}_1$-score
GP-3 ($n = 1110$)	0.174 ± 0.102	0.159 ± 0.056	0.158 ± 0.051
GP-5 ($n = 0$)	N/A	N/A	0.0
GP-9 ($n = 46$)	0.187 ± 0.074	0.245 ± 0.077	0.202 ± 0.048
		Macro-averaged $\mathcal{F}_1$:	0.120 ± 0.087

Table 4.4: Fitness, Precision and $\mathcal{F}_1$-score achieved by the LCS algorithm.

The TBR algorithm achieves a macro-averaged $\mathcal{F}_1$-score of 0.104 ± 0.049. What can be seen is that GP-5 achieves the lowest mean $\mathcal{F}_1$-score of 0.036 ± 0.040, while GP-9 and GP-3 have very similar mean $\mathcal{F}_1$-scores of 0.140 ± 0.026 and 0.137 ± 0.042, respectively. Distributions of the $\mathcal{F}_1$-scores among GPs for both algorithms can be found in Appendix 6.12.

Process Model	Mean Fitness	Mean Precision	Mean $\mathcal{F}_1$-score
GP-3 ($n = 2344$)	0.151 ± 0.075	0.132 ± 0.047	0.137 ± 0.042
GP-5 ($n = 20$)	0.027 ± 0.032	0.057 ± 0.064	0.036 ± 0.040
GP-9 ($n = 84$)	0.120 ± 0.038	0.178 ± 0.037	0.140 ± 0.026
		Macro-averaged $\mathcal{F}_1$:	0.104 ± 0.048

Table 4.5: Fitness, Precision and $\mathcal{F}_1$-score achieved by the TBR algorithm.

The two-tailed paired t-test, as described in subsection 3.5.2, shows that there is no significant difference between the macro-averaged $\mathcal{F}_1$-scores of the LCS algorithm and the TBR algorithm ($t = 0.564$, $p = 0.629$). We thus fail to reject the null hypothesis with a significance level of $\alpha = 0.05$.

4.5.1 Domain Expert Evaluation

Both domain experts have annotated the same set of 18 train journeys following the procedures described in subsection 3.5.3. As can be seen in the main diagonal of the confusion matrix in Figure 4.5(a), the domain experts agreed on 13 of the 18 (72%) samples annotated. This results in a Kappa coefficient of $\kappa = 0.49$, a moderate agreement according to Table 3.4.

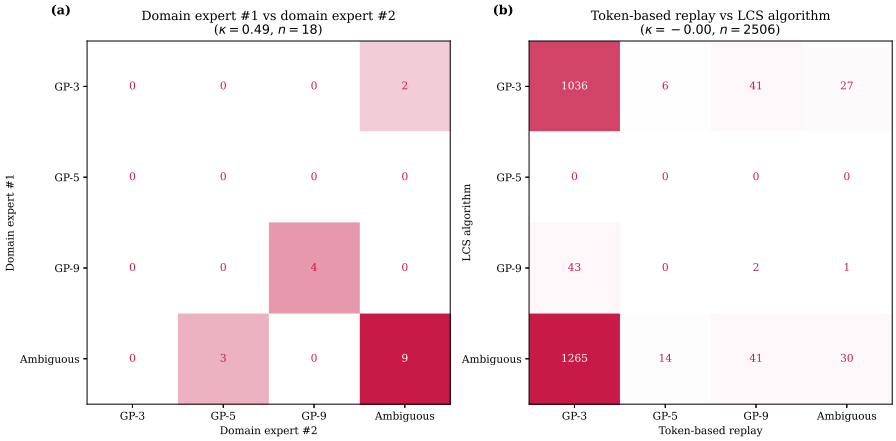

Figure 4.5: Confusion matrices showing the IAA between (a) the domain experts and between (b) the algorithms.

Figure 4.5(b) shows the agreement between the LCS and TBR algorithms for all 2,506 process instances. They achieved a coefficient of $\kappa < 0.0$ and agreed on 1,068 of the 2,506(43%) classifications. A poor agreement following the interpretation of Landis and Koch (1977).

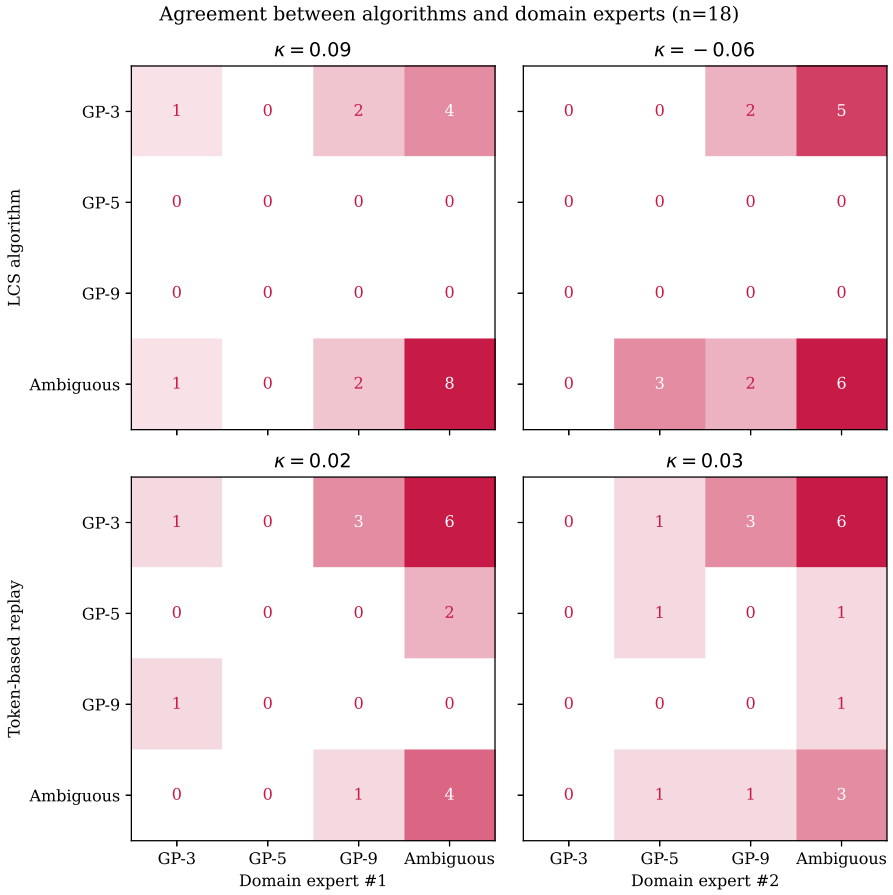

Figure 4.6: Confusion matrices showing the IAA between each algorithm-expert pair. The algorithms are displayed in the rows, while the domain experts are displayed in the columns.

Comparisons between each algorithm-expert pair are shown in Figure 4.6. As the experts annotated 18 of the 2,506 samples, the LCS and TBR algorithms have been evaluated on the same set of 18 samples for this figure. The LCS algorithm achieves a coefficient of $\kappa = 0.09$ with the first domain expert and an $\kappa = -0.06$ in combination with the second expert. The TBR algorithm achieves Kappa scores of $\kappa = 0.02$ and $\kappa = 0.03$ with domain experts #1 and #2, respectively. Both algorithms thus have a slight agreement with domain expert #1, based on Table 3.4. The LCS algorithm has a poor agreement with domain expert #2, while the TBR algorithm has a slight agreement with domain expert #2. All samples, including their classification by the experts and the algorithms, can be found in Appendix 6.13.

5 Conclusions

We start Chapter 5 with the findings on each of the sub-questions and provide an answer to the main question in section 5.1. Section 5.2 is dedicated to the discussion. We conclude the chapter with suggestions for further research in section 5.3.

5.1 Findings

For this section, we first report and summarise the findings for each of the sub-questions. We then provide an answer to the main research question.

In RQ1, we have presented a formalisation for any general tabular data source. This formalisation is applied to a use case in collaboration with ProRail using its largest data warehouse Sherlock. Applying the formalisation on a real dataset substantiates the applicability of the formalisation to domains with similar input data formats. The data used for this thesis contains $2,506$ train journeys distributed over more than $70,000$ events that cover the last four months of 2023. It forms the basis for the PoC developed for this thesis.

The formalisation of a gebruikersproces (GP) as a process model is defined in RQ2. This formalisation is applied to identify 43 unique activities in three of the processes described by ProRail AM (2023). It furthermore enables the conversion of these processes to BPMN diagrams which can be easily extended to capture more paths and variations in the future. A scheme is devised that captures the interaction between traffic controllers, train operators and the system. The resource scheme can be easily extended to accompany new resources. A majority of the activities captured by the process models entail interactions between the system and the train operator.

To retrieve the activities hidden in data, a rule-based system (RBS) is implemented for RQ3 that follows the formalisations defined in RQ1 and RQ2. An RBS is chosen as it is a simple and effective way to extract activities from data. Furthermore, no labelled dataset was available, making supervised learning methods infeasible. The annotated dataset created by the RBS does however enable supervised learning methods to be used in the future. A total of 27 out of the 43 activities are implemented in the RBS. The remaining 16 activities could not be implemented due to data limitations. The RBS has extracted more than $70,000$ activities from the $2,506$ train journeys present in the data. Whereas the system was the most common receiving resource for the process models in Figure 4.2, the train operator is the most common receiving resource in the data. This is visualised in Figure 4.4 and might be an indication of deviations between the process model and the data. One reason could be that many of the 16 missing activities are related to interactions from the train operator and Trdl to the system, creating a blind spot. MT and TM activities are still the least common activities in the data, which is in line with the described process models.

For RQ4, two methods are used to match retrieved sequences of activities to process models: the longest common subsequence (LCS) algorithm and the token-based replay (TBR) process mining technique. The TBR algorithm found a single best match for $2,448$ out of the $2,506$ retrieved sequences of activities instead of only $1,156$ for the LCS algorithm. Activity sequences where an algorithm is incapable of determining a single best match are classified as ambiguous. The LCS algorithm is relatively fast and easily implementable while still providing more insights than other string-matching algorithms such as the Jaccard index and the Levenshtein distance. The TBR algorithm allows for multiple paths to be traversed within a process model. This can be beneficial for the future if the current process models are extended. The LCS algorithm determined GP-3 to be the most common process model and found zero matches for GP-5. The TBR algorithm also identified GP-3 to be the most common process model, finding only 20 and 84 matches for GP-5 and GP-9 respectively. As both algorithms found the most matches for GP-3, it is likely that this is either the most common process in the data or the most accurately represented. Likewise, as

GP-5 is the least common process according to both algorithms, it is possible that GP-5 and its activities are not accurately represented. This could again be due to unimplemented activities or an indication of deviations between the process model and the data.

The performance of the PoC is assessed in RQ5 using the macro-averaged $\mathcal{F}_1$-score of fitness (Eq. (2.1)) and precision (Eq. (2.2)) of the three process models. This measure is chosen as it takes class imbalance into account and makes it possible to use an overarching metric to compare the two algorithms. The macro-average $\mathcal{F}_1$-score of the LCS algorithm is 0.120 ± 0.087 and is heavily influenced by not matching any sequences to GP-5. The TBR algorithm scores 0.104 ± 0.048. It scored lower than the LCS algorithm for both GP-3 and GP-9. A two-tailed paired t-test with a significance level of $\alpha = 0.05$ concluded that there is no significant difference between the macro-average $\mathcal{F}_1$-scores of the two algorithms ($t = 0.564$, $p = 0.629$). The null hypothesis is thus not rejected.

A total of 18 train journeys have been annotated by two domain experts to investigate the difficulty of the task at hand. Their inter-annotator agreement (IAA) is assessed using Cohen's Kappa coefficient and is found to be $\kappa = 0.49$. This indicates a moderate agreement based on Landis and Koch (1977). A poor to slight agreement is found for each algorithm-expert pair with values ranging between $\kappa = -0.06$ and $\kappa = 0.09$. The agreement between the algorithms resulted in a coefficient of $\kappa < 0.0$, indicating a poor agreement and a need for further investigation. The agreement between each algorithm-expert pair is expected to increase as the system is further refined. The agreement between the experts can be further substantiated once more data is annotated as only 18 of the $2,506$ train journeys have been annotated by the experts.

As introduced in section 1.4, this thesis aims to answer the following main research question:

To what extent is it possible to automatically detect user processes in the data of ProRail?

Based on Occam's razor (Duignan, 2024), one could argue that the LCS algorithm is preferred as it is simpler and faster than the TBR algorithm while equally performant based on the conducted t-test. The TBR algorithm is however preferred as it matched more than twice as many retrieved activity sequences to process models. It also allows for multiple paths to be traversed within a process model which will be beneficial if the current process models are extended in the future.

In conclusion, it is, to some extent, possible to automatically detect user processes in the data of ProRail. Multiple improvements can be made to the PoC to increase the performance of both the retrieval and the matching process. The RBS has made the creation of an annotated dataset possible which can be used to train supervised learning models in the future. Further improvements include extending the current dataset, making it possible to implement the remaining activities. This allows for a more accurate representation of each process model. Extending the process models with alternative paths will further exploit the capabilities of the TBR algorithm, distinguishing it from the LCS algorithm. The IAA between domain experts can be improved by refining the guidelines for the annotation process. The annotation of more data by domain experts will also increase the reliability of the IAA between domain experts and help in further substantiating the findings of this thesis.

5.2 Discussion

In this section, we discuss several limitations of the research conducted. We also discuss possible improvements to counter these limitations.

5.2.1 Annotation Scheme

The current scheme to annotate each possible resource-pair, described in Table 3.1, is unable to distinguish two resources of the same type. It is for instance not possible to distinguish between two Trdls, both annotated as T. The resource scheme would ID this activity as a TT activity.

Furthermore, if an activity happens more than once in a process, it is impossible to distinguish between the different occurrences. For example, activity TS_1 occurs twice in process model GP-9. Both times, the activity is labelled TS_1, making it impossible to distinguish between the two occurrences, except for the order in which they occur.

Considering that only a minority of the activities occur more than once in the same process model, the first drawback was acceptable for this PoC. In only a small number of all forty process models, two resources of the same type are present. Furthermore, no interaction between two resources of the same type takes place in the three chosen process models. The second disadvantage was therefore also acceptable for this PoC.

A downside to a more detailed notation of the activities would be that it can become very sparse. This could result in an algorithm not being able to learn from it and is a trade-off that should be considered when deciding onthe level of detail.

5.2.2 Process Models and BPMN Diagrams

The current set of investigated process models is limited to three of the forty GPs described for the Amsterdam – Utrecht railway track. Extending the number of implemented process models would provide a more complete overview of the performance of the proposed approach.

GP-8, "Driving over a normally set route", is one of the 37 process models not yet implemented. This process captures the regular operations of a train and will thus be very common. Implementing this process model would allow for the filtering out of regular operations, thus making it possible to focus on the three more advanced process models already implemented.

The BPMN diagrams, presented in Appendix 6.8, are simple representations of the process models described by ProRail. They only capture a single path through each process and do not capture loops or parallel paths currently only present in the accompanying notes. An additional risk to having accompanying notes beside the process models is that the two might not be consistent with each other (Van der Aa et al., 2017). This could lead to confusion and process models that incorrectly represent the real world. Extending the diagrams to capture all possible paths of a process model would especially be beneficial for the TBR algorithm as it inherently supports more advanced features such as loops and parallel paths.

5.2.3 Implemented Rules and Dataset Quality

Some activities share the exact same set of rules. For example, the two least common activities MS_{11} and SM_{12} both have the ruleset $\{r_{14}\}$. Although the two activities have different semantics, the current dataset does not contain sufficiently detailed information to distinguish between the two activities. This is a limitation of the current dataset.

As can be seen in Figure 4.3, more than $19,000$ of the $33,025$ events contain more than one activity. Each event of the dataset represents a new step of a train its timetable. This level of detail makes it difficult to determine the exact order of activities within an event. The possibly incorrect order of activities within an event could hurt the performance of both matching algorithms. It would be beneficial to develop a dataset with a greater level of detail that can capture the exact timestamp of each activity, thus allowing for a more precise ordering of activities.

Currently, the RBS only checks which activities are applicable for a matrix M_i. It does not check which prerequisites are met for a matrix. If not all prerequisites of a process model are met, the process model can never be applicable to the given matrix. Adding a prerequisites check would make the RBS more robust and precise, thus reducing the number of false positives per process model.

5.2.4 Matching Algorithms Evaluation

The evaluation of the matching algorithms was conducted using a two-tailed paired t-test. The data used for the t-test is possibly not normally distributed. As only three of the forty process models were used for the evaluation, the number of data points is limited. This means that the results of the t-test should be interpreted with caution.

The IAA between the two domain experts was calculated to be $\kappa = 0.49$. This indicates a moderate agreement according to Landis and Koch (1977). A possible explanation for an imperfect agreement could be that the experts were not provided with clear enough guidelines on how to annotate the data. This could have led to different interpretations of the data and thus a lower agreement. Furthermore, there is a possibility that a train journey contains more than one process. Both domain experts could thus have identified the correct process model, but for different parts of the same train journey. Providing clearer guidelines hopefully increases the agreement between domain experts and helps to gain more insight into the quality of the dataset.

5.3 Further Research

Several possibilities for future research are discussed in this section.

5.3.1 Extending the Rule-based System

The current pseudocode for the RBS, described in Algorithm 1, is an initial implementation of the system. It could be extended in many ways.

For example, for an activity a_n to be true, all associated rules $R(a_n)$ must be true. This could be extended to a system where the activity is true if a certain percentage of the rules are true. For example, if $\geq 50\%$ of the rules are true, then the activity could be considered true. This can be useful in cases where data is noisy or incomplete as it allows for some uncertainty in the data.

In addition, all rules of an activity are currently considered to be equally important as falsifying any rule results in the activity being false. This could be altered to a system where each rule has a weight associated with it. The weight could be based on the importance of the rule or the degree of certainty the domain expert has in the rule.

A further improvement would be to attach a timestamp to each retrieved activity. This would allow us to order the activities chronologically instead of relying on the order in which the activities are retrieved. This would furthermore help solve the issue presented in subsection 5.2.3 where we highlighted that the order of activities within an event is possibly incorrect. As mentioned in subsection 5.2.3, the level of detail of the current dataset makes it difficult to determine the exact timestamp for each activity and thus the exact order of activities within an event.

5.3.2 Dataset Creation and Supervised Learning

The data used in this research is limited to the data available in Sherlock. The dataset can be extended by adding data from other sources. For example, the onboard systems of the train store additional data about its SoM. This data could be used to implement the SoM activities currently missing in the RBS, as discussed in section 4.3.

The annotated dataset created with the PoC is a valuable resource for future research as it could be used as a ground truth. The dataset could, for instance, be used to train supervised learning models. This was previously not possible as no annotated data was available. The use of

domain experts adds a layer of validation to the annotated dataset and ensures that it is of high quality.

5.3.3 Improving Process Models

Improving the process models requires knowledge from domain experts to ensure that the processes are modelled correctly. This is a time-consuming task and requires a lot of effort from the experts. The use of process enhancement, as briefly introduced in subsubsection 2.4.1.4, could potentially be used to improve the efficiency of this process and improve the process models themselves.

The activities present in any $A_{\text{retrieved}}$ but not in the matched A_{id} are interesting for future research. These activities are not specified in the process model but are retrieved from data. This could be an indication of missing activities in the process model. Both conformance checking and process enhancement techniques, as discussed in subsection 2.4.1, could be used to investigate this further. Incomplete or inaccurate process models pose a risk to ProRail's operations and the safety of the railway network.

The other way around, activities present in A_{id} but not in $A_{\text{retrieved}}$, could also hold valuable information. These activities are specified in the process model but have not been found in the data. Activities not found in the data could indicate that the algorithm used is not functioning properly. This could be an indication of a bug, ill-defined logic rules, or a change in the input data format. Another possibility could be that the process model is incorrect, which is of interest for the domain experts to further investigate.

5.3.4 Semantic Similarity

The algorithms mentioned in subsection 2.4.2 calculate the similarity between two sequences using their syntactic difference. Semantic similarity algorithms go one step further and can find terms that are similar in meaning. This is particularly useful for synonyms or related terms with different spellings. For example, syntactic similarity algorithms perform well on matching the terms 'train' and 'tram' but perform poorly on matching 'train' with 'railway'. Semantic similarity algorithms therefore help to match not only similar terms but also related terms. Semantic similarity algorithms are not only beneficial for matching two sequences of activities but also for matching activities to columns or column descriptions of a dataset.

A proposal for future research is elaborated on in Appendix 6.14.

6 Appendix

6.1 Original Problem Definition From ProRail

"This research project focuses on event detection and classification in the context of railway operations. This is a collaborative project with ProRail (Traffic Control) in Utrecht. The aim of the project is to improve the reliability and accuracy of event detection and classification, through the application of machine learning techniques, so that operational personnel, such as traffic controllers and traffic managers, can use the detected events to initiate appropriate procedures. The project will also focus on the integration of these techniques into a prototype user interface, to test the tool that could facilitate the work of our train traffic operators.

The ideal candidate for this position must possess a strong analytical and problem-solving mindset, with proven experience in data analysis and ideally with programming languages such as Python, Java, or any that may be appropriate. Additionally, candidates should have a solid understanding of machine learning techniques, as well as a keen interest in applying these technologies to improve rail traffic control room operations."

6.2 Distribution of NTC Systems

Figure 6.1: Distribution of various NTC systems across the Dutch railway network (ProRail, 2023, p. 217). The dual signalling track between Amsterdam (Asd) and Utrecht (Ut) is indicated in blue and orange. Other railway tracks, including the track between Lelystad (Lls) and Zwolle (Zl), are out of the scope of this thesis.

6.3 All GPs for the Asd – Ut Track

Process	Name (Original)	Name (Translated)	Chapter (Translated)
GP-1	Oprijden naar een normaal ingestelde rijweg met bekende treinpositie	Heading towards a normally set route with known train position	Departure and arrival
GP-2	Oprijden naar een ROZ-rijweg met bekende treinpositie	Heading towards a ROZ route with known train position	
GP-3	Vertrek met onbekende treinpositie	Departure with unknown train position	
GP-4	Oprijden naar normale rijweg met wissels tussen trein en vertreksein	Heading towards normal route with switches between train and departure signal	
GP-43	Vertrek vanuit een niet ontrokken vrijgave gebied	Departure from an unused clearance area	
GP-62	Vertrek met de kop voorbij het sein wanneer er een ROZ-rijweg “over de trein heen” kan worden ingesteld	Departure with the head past the signal when an ROZ route ”over the train” can be set	
GP-68	Vertrek met de kop voorbij het sein wanneer er geen ROZ-rijweg “over de trein heen” kan worden ingesteld	Departure with the head past the signal when no ROZ route ”over the train” can be set	
GP-6	Korte stop	Short stop	
GP-201	Vertrek onder level NTC	Departure under level NTC	
GP-207	Oprijden naar een rijweg zonder radioverbinding	Heading towards a route without radio connection	
GP-7	Wegzetten van een trein	Putting a train away	
GP-8	Rijden over een normaal ingestelde rijweg	Driving over a normally set route	Driving the train
GP-10	Overgang ‘normaal rijden’ naar ‘rijden op zicht’	Change from ‘normal running’ to ‘running on sight’	
GP-11	Overgang ‘rijden op zicht’ naar ‘normaal rijden’	Change from ‘running on sight’ to ‘normal running’	
GP-12	Rijden op glad spoor	Driving on slippery track	
GP-202	Het rijden onder een tijdelijke snelheidsbeperking	Driving during a temporary speed restriction	
GP-5	De passage van een stoptonend sein zonder MA	Passing a stop signal without MA	Instructions
GP-203	Herroepen van een rijweg waarbij de trein tot stilstand komt voor het begin van de te herroepen rijweg	Revoking a route in which the train stops before the start of the route to be revoked	Revoking a route
GP-29	Herroepen van een rijweg waarbij de trein tot stilstand komt in de te herroepen rijweg	Revoking a route in which the train comes to a stop in the route to be revoked	
GP-67	Vrijmaken van een restrijweg onder een L2-trein	Clearing a residual route under an L2 train	

GP-9	Het keren of kopmaken van een trein	Turning or reversing a train	Shunting movements within a centrally controlled area
GP-51	Het splitsen van een trein	Splitting of a train	
GP-50	Het combineren van twee treinen	Combining two trains	
GP-13	Transitie van level NTC ATB naar Level 2	Transition from level NTC ATB to Level 2	Passage of special locations
GP-15	Transitie van Level 2 naar level NTC ATB	Transition from Level 2 to level NTC ATB	
GP-204	De passage van een helling door een zware goederentrein	Passing a slope by a heavy freight train	
GP-208	Inrijden van een niet ontrokken vrijgave gebied	Entering a non-denied clearance area	
GP-37	De afhandeling van een STS passage	Handling of an STS passage	Disruptions and irregularities
GP-36	De afhandeling van het uitvallen van de verbinding met het RBC	Handling of the failure of the connection to the RBC	
GP-31	De afhandeling van de treinenloop bij een tunnelincident	Handling of the train running during a tunnel incident	
GP-32	Herstel van de treinenloop na een tunnelincident	Recovery of train running after a tunnel incident	
GP-55	De afhandeling van een voorwaardelijke noodstop waarbij de trein stopt voor de nieuwe EoA	Handling a conditional emergency stop where the train stops before the new EoA	
GP-206	De passage van een gedoofd sein	The passage of a failing signal	
GP-35	De afhandeling van een ingreep als gevolg van een balisefout	Handling of an emergency brake due to a balise failure	
GP-34	Verder rijden met buiten bedrijf gesteld ETCS-systeem na treinstoring	Continuing to run with decommissioned ETCS system after train failure	
GP-209	De afhandeling van een remming als gevolg van een balisegroep inconsistentie door een trein onder level NTC	Handling of an intervention due to a balise group inconsistency by a train driving under level NTC	
GP-72	Vanaf het CBG een werkgebied inrijden met een lichtsein op de grens	Entering a working area from the CBG with a light signal at the border	Driving to and from a work area
GP-73	Vanaf het CBG een werkgebied inrijden zonder lichtsein op de grens	Entering a working area from the CBG without a light signal at the border	
GP-74	Vanaf een werkgebied het CBG inrijden met een lichtsein op de grens	Entering a CBG from a working area with a light signal at the border	
GP-75	Vanaf een werkgebied het CBG inrijden zonder een lichtsein op de grens	Entering a CBG from a working area without a light signal at the border	

Table 6.1: Table containing all GPs present in ProRail AM (2023).

6.4 All DRPs of the Asd – Ut Track

Ac	Aco	Asa	Asb	Asdar	Asdz	Ashd	Bkl	Bkla	Ddm	Dmnz
Dvaw	Dvaz	Dvd	Hmlba	Mas	Mdsa	Rai	Ut	Utma	Utzl	Vspa

Table 6.2: The 22 DRPs covering the complete track between Amsterdam and Utrecht. This includes connecting arches and transition areas.

6.5 Petri Nets of Process Models

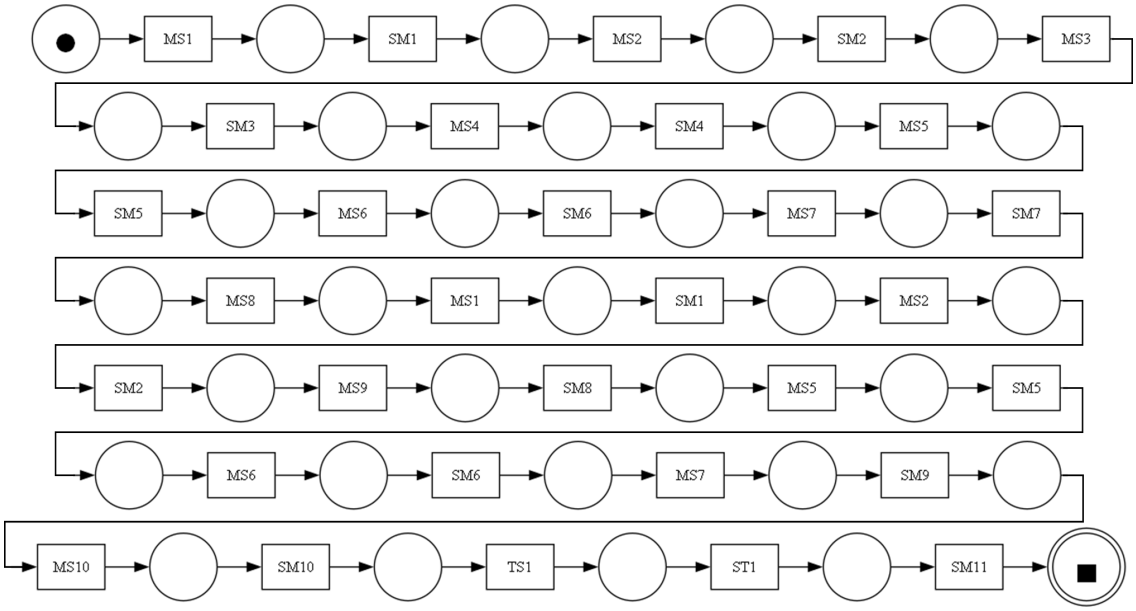

Figure 6.2: Petri net of GP-3. Transitions are indicated as rectangles, places as circles, and arcs as arrows. The source place is visible in the top-left corner. The sink place is in the bottom-right corner.

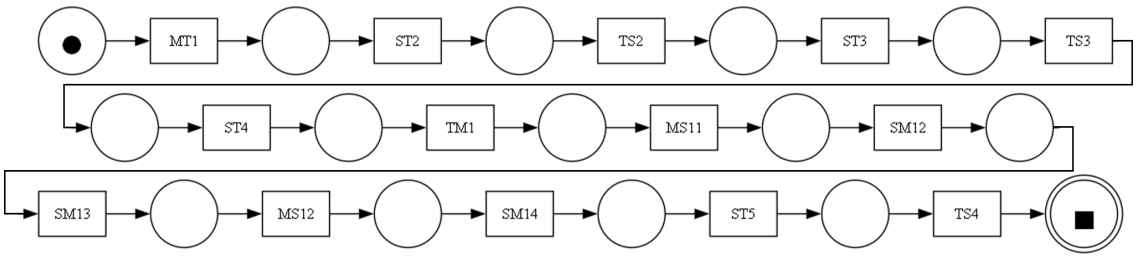

Figure 6.3: Petri net of GP-5.

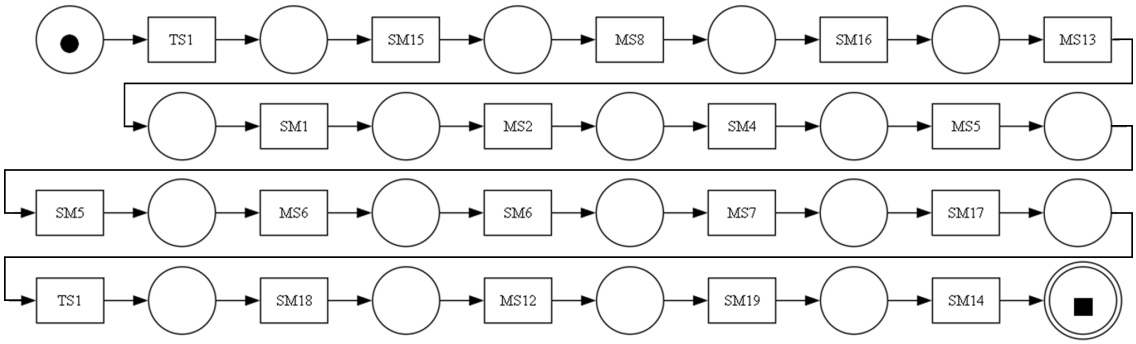

Figure 6.4: Petri net of GP-9.

6.6 Data Exploration Results

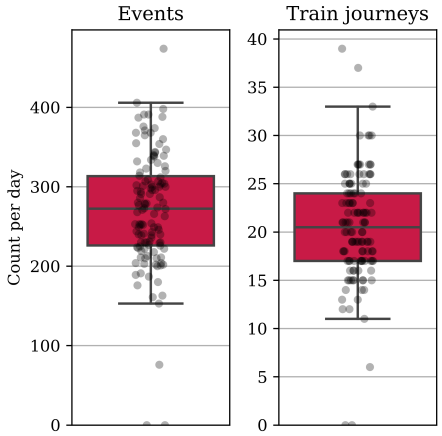

Figure 6.5: Spread of the number of events and train journeys per day.

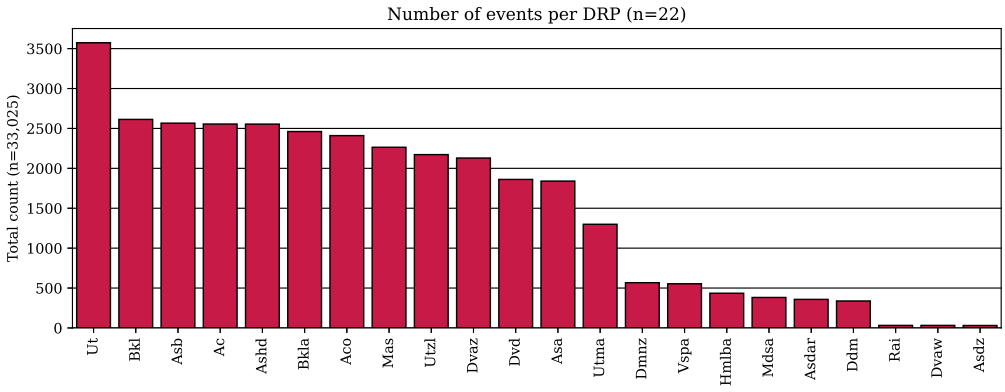

Figure 6.6: Spread of the number of events among the DRPs.

6.7 All Activities of GP-3, GP-5 and GP-9

GP ID	GP step	Activity ID	Activity Description
3	1	MS_1	Inschakelen stuurstroom
	2	SM_1	Toont opgeslagen driver-id ter invoer of bevestiging en biedt mogelijkheid voor invoer/bevestiging treinnummer
	3	MS_2	Wijzigt dan wel bevestigt driver-id en treinnummer
	4	SM_2	Toont opgeslagen level ter invoer of bevestiging
	5	MS_3	Kiest/bevestigt Level 2
	6	SM_3	Toont opgeslagen radioverbindingsgegevens
	7	MS_4	Bevestigt of wijzigt opgeslagen radioverbindingsgegevens
	8	SM_4	Verbinding trein/RBC zichtbaar in afwachting van keuze
	9	MS_5	Kiest invoer treingegevens
	10	SM_5	Toont opgeslagen treingegevens ter bevestiging en of wijziging
	11	MS_6	Wijzigt eventueel treingegevens en bevestigt daarna de treingegevens
	12	SM_6	In afwachting van start
	13	MS_7	Kiest start
	14	SM_7	Tekstmelding "Omschakelen ATB"
	15	MS_8	Uitschakelen stuurstroom
	17	MS_1	Inschakelen stuurstroom
	18	SM_1	Toont opgeslagen driver-id ter invoer of bevestiging en biedt mogelijkheid voor invoer/bevestiging treinnummer
	19	MS_2	Wijzigt dan wel bevestigt driver-id en treinnummer
	20	SM_2	Toont opgeslagen level ter invoer of bevestiging
	21	MS_9	Kiest level NTC met de STM voor ATB
	22	SM_8	Level NTC met STM voor ATB actief in afwachting van keuze
	23	MS_5	Kiest invoer treingegevens
	24	SM_5	Toont opgeslagen treingegevens ter bevestiging en of wijziging
	25	MS_6	Wijzigt eventueel treingegevens en bevestigt daarna de treingegevens
	26	SM_6	In afwachting van start
	27	MS_7	Kiest start
	28	SM_9	Stelt SN mode voor
	29	MS_{10}	Bevestigt SN mode
	30	SM_{10}	Schakelt naar mode SN
	32	TS_1	Opdracht om normale rijweg in te stellen van spoor A naar spoor B en verder
	33.1	ST_1	Rijweg is ingesteld
	33.2	SM_{11}	Neemt waar dat sein X uit de stand stop is
5	1	MT_1	Verzoekt toestemming sein X te mogen passeren (EI 1)
	2.1	ST_2	Controleert of STS-route van spoor A naar spoor B veilig kan worden ingesteld
	2.2	TS_2	Stelt STS-route in
	2.3	ST_3	Verifieert dat de STS-route beschikbaar is voor de trein
	2.4	TS_3	Stelt normale vervolgrijweg in van spoor B naar spoor C
	2.5	ST_4	Verifieert dat de vervolgrijweg beschikbaar is voor de trein
	2.6	TM_1	Toestemming om sein X te passeren (EI 1)
	5	MS_{11}	Activeert override-functie
	6	SM_{12}	Omschakeling naar SR en override indicatie
	8	SM_{13}	Omschakeling naar OS en OS-voorstel
	9	MS_{12}	Bevestigt OS
	11	SM_{14}	Omschakeling naar FS

	13	ST_5	Stelt vast dat de trein het eindpunt van de STS-route is gepasseerd
	14	TS_4	Herroept STS-route van spoor A naar B
9	1	TS_1	Opdracht om normale rijweg in te stellen van spoor A naar spoor B
	2	SM_{15}	FSMA verlengd tot sein Y
	4	MS_8	Schakelt de stuurstroom af
	5	SM_{16}	Trein in SB
	7	MS_{13}	Schakelt de stuurstroom aan de andere zijde van de trein in
	8	SM_1	Toont opgeslagen driver-id ter invoer of bevestiging en biedt mogelijkheid voor invoer/bevestiging treinnummer
	9	MS_2	Bevestigt of wijzigt driver-id en treinnummer
	10	SM_4	Verbinding met RBC zichtbaar in afwachting van keuze
	11	MS_5	Kiest invoer treingegevens
	12	SM_5	Toont opgeslagen treingegevens ter invoer of bevestiging
	13	MS_6	Wijzigt eventueel treingegevens en bevestigt daarna de treingegevens
	14	SM_6	In afwachting van start
	15	MS_7	Kiest start
	16	SM_{17}	Tekstmelding "Wacht"
	18	TS_1	Opdracht om normale rijweg in te stellen van spoor B naar spoor A
	19	SM_{18}	OS voorstel
	20	MS_{12}	Bevestigt OS
	21	SM_{19}	Omschakeling naar OS
	23	SM_{14}	Omschakeling naar FS

Table 6.3: All activities of GP-3, GP-5 and GP-9. Each activity is assigned a unique ID according to the resource scheme in Table 3.1. Equal activities have been assigned equal IDs. GP step refers to the step of the sequence diagram of the relevant GP as denoted in ProRail AM (2023).

6.8 BPMN Diagrams of GP-3, GP-5 and GP-9

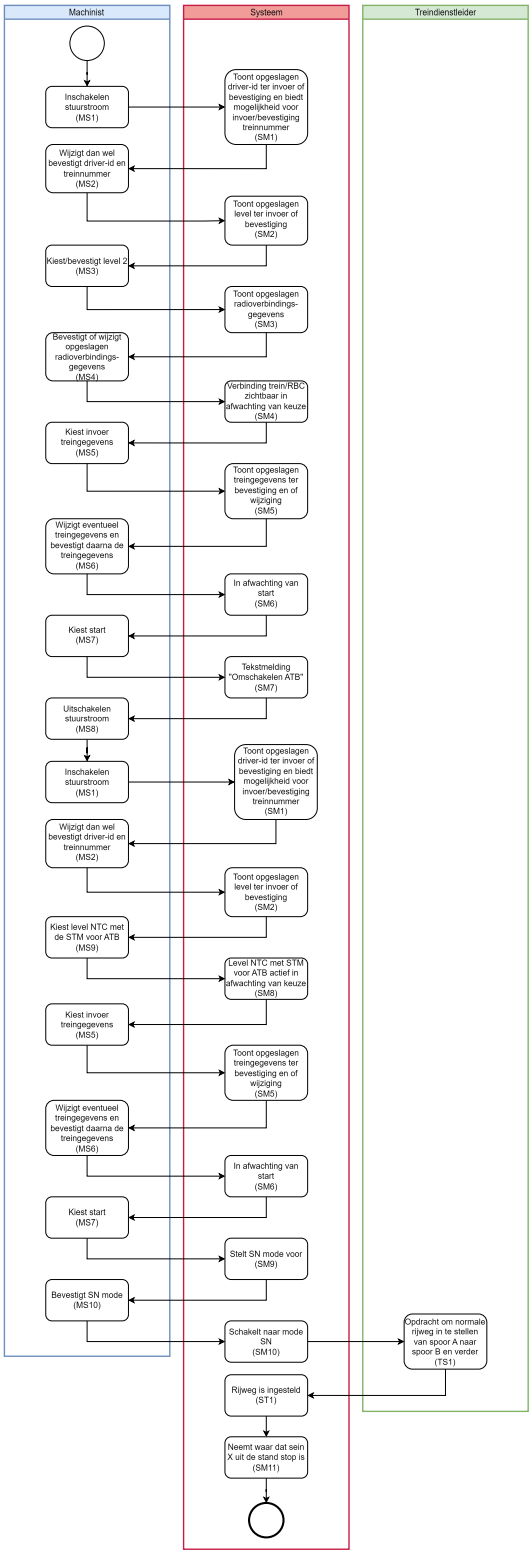

Figure 6.7: BPMN diagram of GP-3

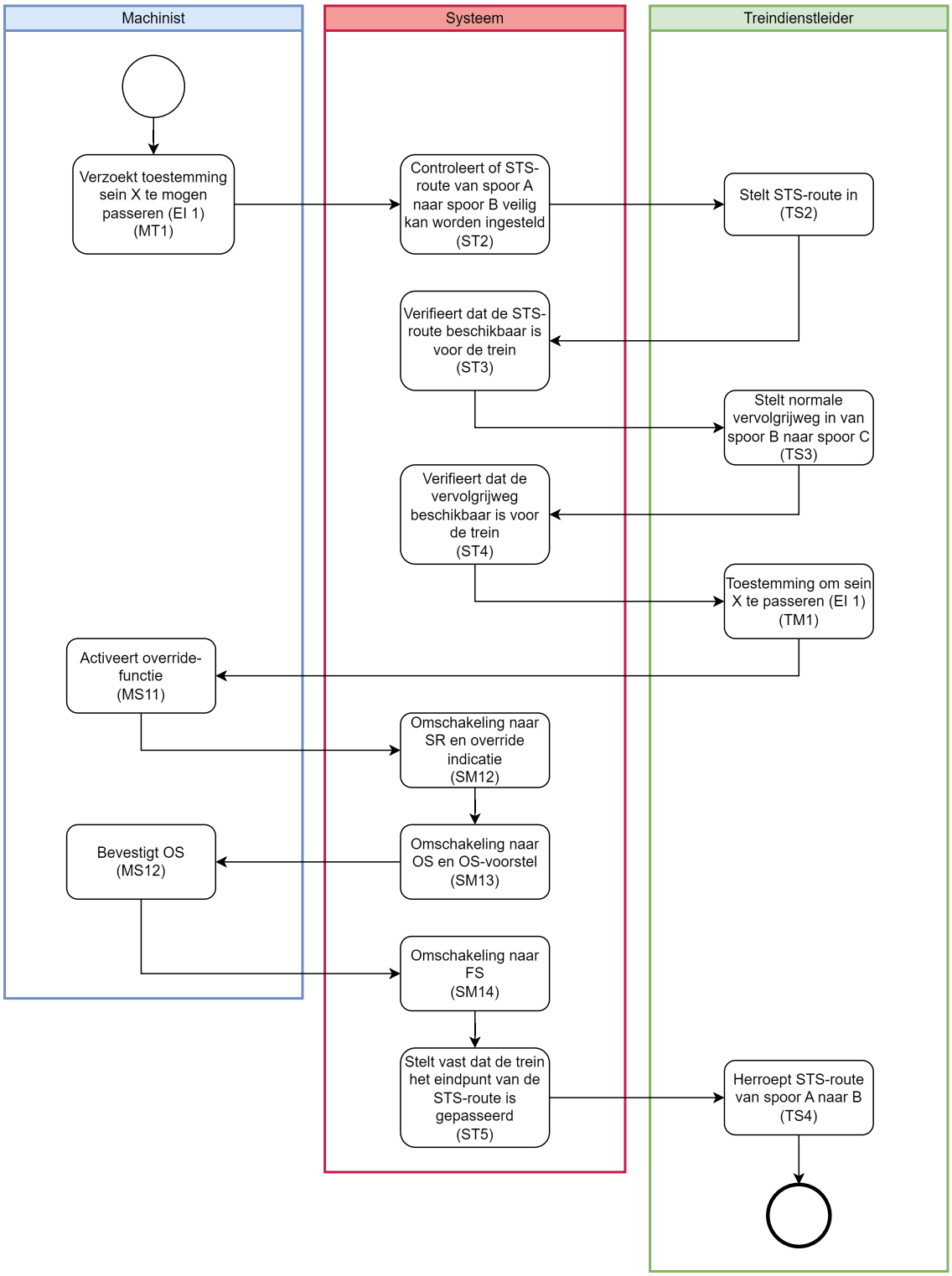

Figure 6.8: BPMN diagram of GP-5

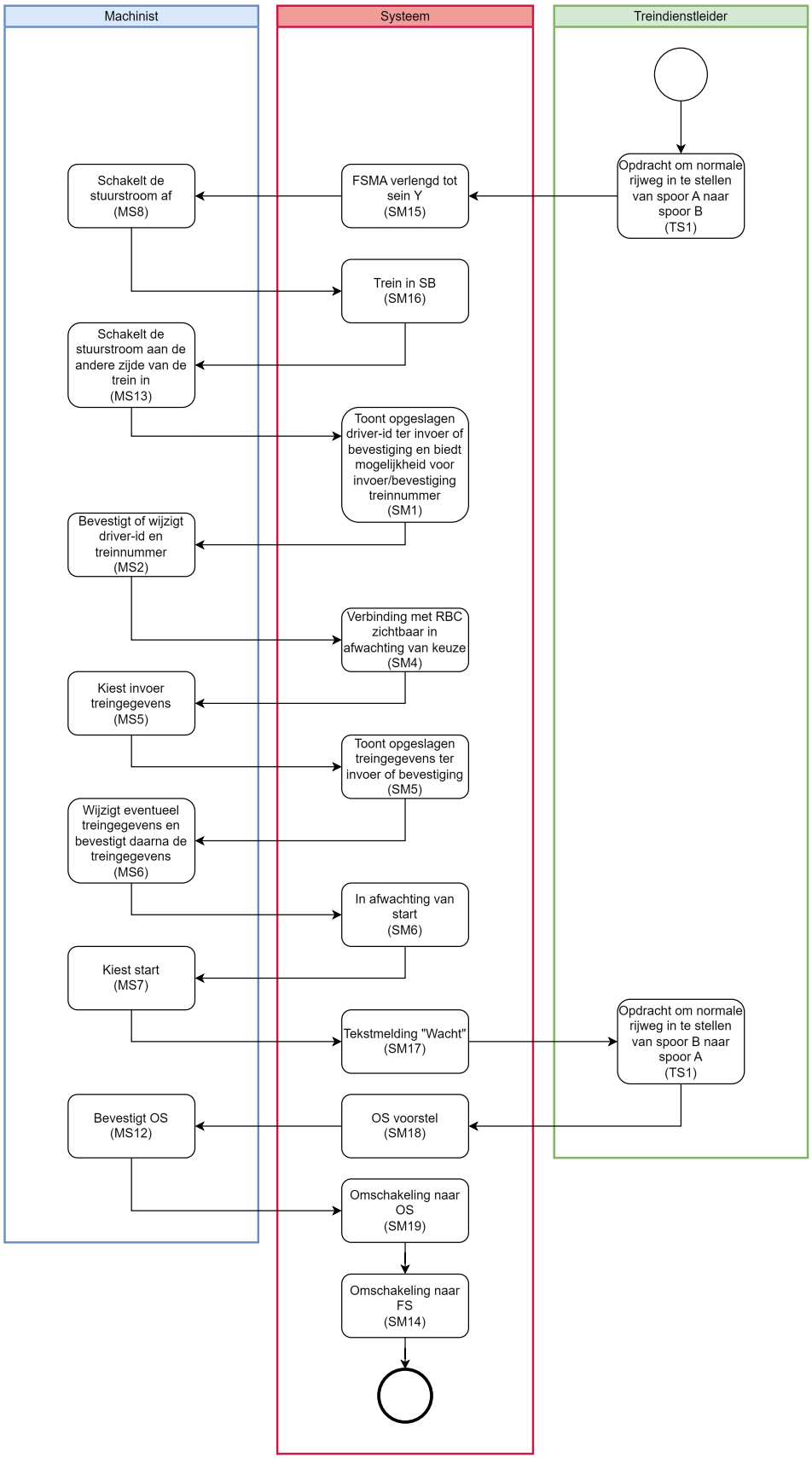

Figure 6.9: BPMN diagram of GP-9

6.9 All Implemented Rules of the RBS

Rule ID	Rule Name	Rule Description
r_1	RuleSB	True if 'qats_qpr_level' contains 'L2SB'
r_2	RuleMatTimestamp	True if it is the first occurrence of the current timestamp in 'mts_tijdstip_volgensmatnr'
r_3	RuleOr($r_1 \vee r_2$)	True if any of the rules in the set pass
r_4	RuleGSMRTimestamp	True if it is the first occurrence of the current timestamp in 'mts_tijdstip_registratie_volgenstreinnr'
r_5	RuleMcnReady	True if it is the first occurrence of the current timestamp in 'stipt_sub_pers_gereedtijd'
r_6	RuleFSPreviousSuffix	True if 'qats_qpr_level' in the previous event contains 'L2FS'
r_7	RuleOSPreviousSuffix	True if 'qats_qpr_level' in the previous event contains 'L2OS'
r_8	RuleOr($r_6 \vee r_7$)	True if any of the rules in the set pass
r_9	RuleSRPrefix	True if 'qats_qpr_level' starts with 'L2SR'
r_{10}	RuleAnd($r_8 \wedge r_9$)	True if all of the rules in the set pass
r_{11}	RuleFSSR	True if 'qats_qpr_level' contains 'L2FS,L2SR'
r_{12}	RuleOSSR	True if 'qats_qpr_level' contains 'L2OS,L2SR'
r_{13}	RuleOr($r_{11} \vee r_{12}$)	True if any of the rules in the set pass
r_{14}	RuleOr($r_{10} \vee r_{13}$)	True if any of the rules in the set pass
r_{15}	RuleNot($\neg r_7$)	Negates the result of the rule
r_{16}	RuleOSPrefix	True if 'qats_qpr_level' starts with 'L2OS'
r_{17}	RuleAnd($r_{15} \wedge r_{16}$)	True if all of the rules in the set pass
r_{18}	RuleNot($\neg r_{16}$)	Negates the result of the rule
r_{19}	RuleOS	True if 'qats_qpr_level' contains 'L2OS'
r_{20}	RuleAnd($r_{18} \wedge r_{19}$)	True if all of the rules in the set pass
r_{21}	RuleOr($r_{17} \vee r_{20}$)	True if any of the rules in the set pass
r_{22}	RuleDirectionChange	True if the direction of the train has changed
r_{23}	RuleGSMRDuration	True if a GSM-R call takes longer than 60 seconds
r_{24}	RuleUnplannedStop	True if the train made an unplanned stop
r_{25}	RuleMatReady	True if it is the first occurrence of the current timestamp in 'stipt_sub_matovergang_gereedtijd'
r_{26}	RuleDepartureDelayed	True if the cause of delay is 'vertrekken_starter'
r_{27}	RuleSNPreviousSuffix	True if 'qats_qpr_level' in the previous event ends with 'SN'
r_{28}	RuleNot($\neg r_{27}$)	Negates the result of the rule
r_{29}	RuleL2SNPrefix	True if the prefix of 'qats_qpr_level' is 'L2SN'
r_{30}	RuleSNPrefix	True if the prefix of 'qats_qpr_level' is 'SN'
r_{31}	RuleOr($r_{29} \vee r_{30}$)	True if any of the rules in the set pass
r_{32}	RuleAnd($r_{28} \wedge r_{31}$)	True if all of the rules in the set pass
r_{33}	RuleNot($\neg r_{29}$)	Negates the result of the rule
r_{34}	RuleNot($\neg r_{30}$)	Negates the result of the rule
r_{35}	RuleSN	True if 'qats_qpr_level' contains 'SN'
r_{36}	RuleAnd($r_{33} \wedge r_{34} \wedge r_{35}$)	True if all of the rules in the set pass
r_{37}	RuleOr($r_{32} \vee r_{36}$)	True if any of the rules in the set pass
r_{38}	RuleSignalSafe	True if it is the first occurrence of the current timestamp in 'trento_av_vertrekinrijsein_seinveilig'
r_{39}	RuleSRPreviousSuffix	True if 'qats_qpr_level' in the previous event ends with 'L2SR'

r_{40}	RuleAnd($r_{16} \wedge r_{39}$)	True if all of the rules in the set pass
r_{41}	RuleSROSCurrent	True if 'qats_qpr_level' contains 'L2SR,L2OS'
r_{42}	RuleOr($r_{40} \vee r_{41}$)	True if any of the rules in the set pass
r_{43}	RuleFSPrefix	True if 'qats_qpr_level' starts with 'L2FS'
r_{44}	RuleAnd($r_7 \wedge r_{43}$)	True if all of the rules in the set pass
r_{45}	RuleOSFSCurrent	True if 'qats_qpr_level' contains 'L2OS,L2FS'
r_{46}	RuleOr($r_{44} \vee r_{45}$)	True if any of the rules in the set pass
r_{47}	RuleFS	True if 'qats_qpr_level' contains 'L2FS'
r_{48}	RuleRouteDelayed	True if the cause of delay starts with 'rijweg'
r_{49}	RuleDeparture	True if 'basic_drp_act' is a departure activity
r_{50}	RuleSBPreviousSuffix	True if 'qats_qpr_level' in the previous event ends with 'L2SB'
r_{51}	RuleAnd($r_{16} \wedge r_{50}$)	True if all of the rules in the set pass
r_{52}	RuleSBOSCurrent	True if 'qats_qpr_level' contains 'L2SB,L2OS'
r_{53}	RuleOr($r_{51} \vee r_{52}$)	True if any of the rules in the set pass
r_{54}	RuleRouteNormal	True if 'prl_wpk_code' contains 'N'
r_{55}	RuleNormal-RouteByPPR	True if 'prl_wpk_code' contains 'N' and 'prl_wpk_ingestelddoor' contains 'PPR' at the same index
r_{56}	RuleNormalRouteBy-BIF	True if 'prl_wpk_code' contains 'N' and 'prl_wpk_ingestelddoor' contains 'BIF' at the same index
r_{57}	RuleOr($r_{55} \vee r_{56}$)	True if any of the rules in the set pass
r_{58}	RuleSignalRed	True if the signal was red when passed

Table 6.4: Table containing the ID, name and description of each implemented rule of the RBS. Rules may contain references to previously defined rules. The description contains the condition that needs to be met for the rule to be true and refers to the columns in the dataset.

6.10 All Implemented Activities of the RBS

Activity ID	Rules
MS_1	{r_3: RuleOr($r_1 \vee r_2$)}
MS_2	{r_4: RuleGSMRTimestamp}
MS_7	{r_5: RuleMcnReady}
MS_{11}	{r_{14}: RuleOr($r_{10} \vee r_{13}$)}
MS_{12}	{r_{21}: RuleOr($r_{17} \vee r_{20}$)}
MS_{13}	{r_{22}: RuleDirectionChange}
MT_1	{r_{23}: RuleGSMRDuration $\wedge$ r_{24}: RuleUnplannedStop}
SM_1	{r_4: RuleGSMRTimestamp}
SM_4	{r_2: RuleMatTimestamp}
SM_6	{r_{25}: RuleMatReady}
SM_7	{r_{26}: RuleDepartureDelayed}
SM_{10}	{r_{37}: RuleOr($r_{32} \vee r_{36}$)}
SM_{11}	{r_{38}: RuleSignalSafe}
SM_{12}	{r_{14}: RuleOr($r_{10} \vee r_{13}$)}
SM_{13}	{r_{42}: RuleOr($r_{40} \vee r_{41}$)}
SM_{14}	{r_{46}: RuleOr($r_{44} \vee r_{45}$)}
SM_{15}	{r_{47}: RuleFS}
SM_{16}	{r_1: RuleSB}
SM_{17}	{r_{48}: RuleRouteDelayed $\wedge$ r_{49}: RuleDeparture}
SM_{18}	{r_{21}: RuleOr($r_{17} \vee r_{20}$)}
SM_{19}	{r_{53}: RuleOr($r_{51} \vee r_{52}$)}
ST_1	{r_{54}: RuleRouteNormal}
ST_4	{r_{57}: RuleOr($r_{55} \vee r_{56}$)}
ST_5	{r_{42}: RuleOr($r_{40} \vee r_{41}$)}
TM_1	{r_{58}: RuleSignalRed}
TS_1	{r_{54}: RuleRouteNormal}
TS_3	{r_{57}: RuleOr($r_{55} \vee r_{56}$)}

Table 6.5: Table containing the ID and set of rules defined for each implemented activity of the RBS. The rules of an activity refer to the rules defined in Appendix 6.9.

6.11 Matching Algorithms Results

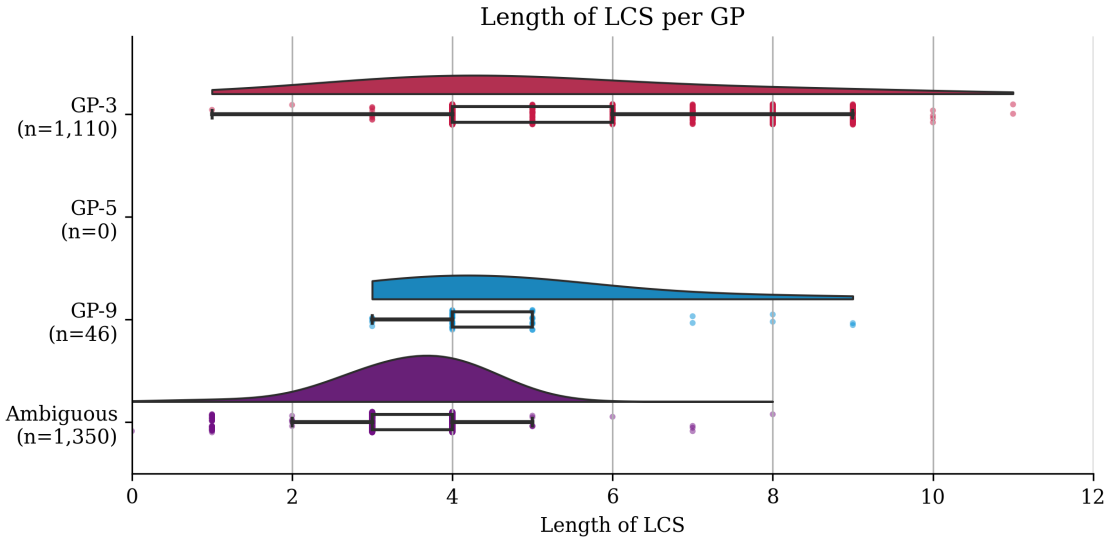

Figure 6.10: Raincloud plot visualising the distribution of the length of the LCSs among the three process models and sequences classified as ambiguous.

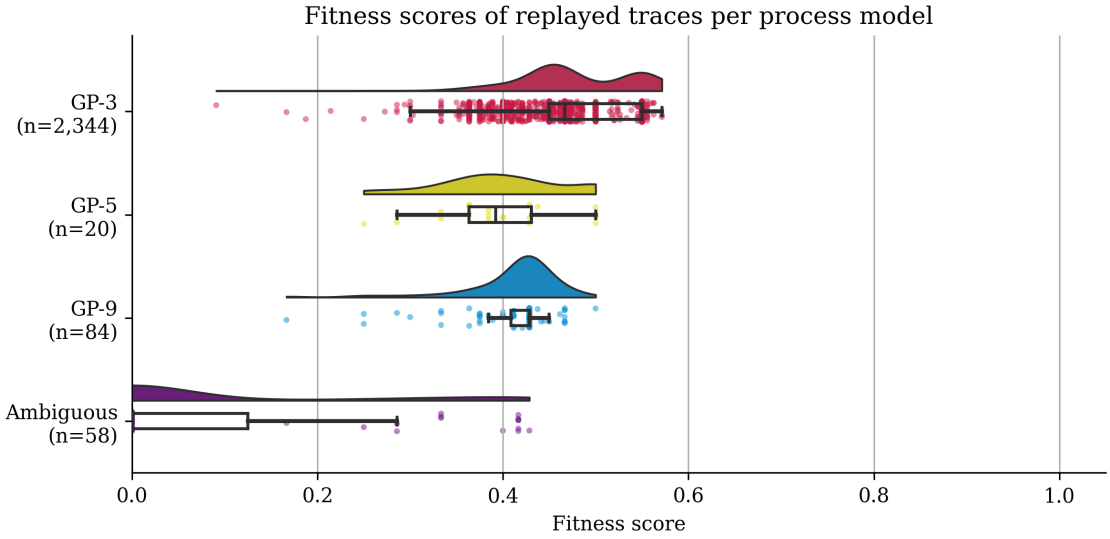

Figure 6.11: Raincloud plot visualising the distribution of the fitness$_{PN}$ score among the three process models and sequences classified as ambiguous.

6.12 F1-score Results

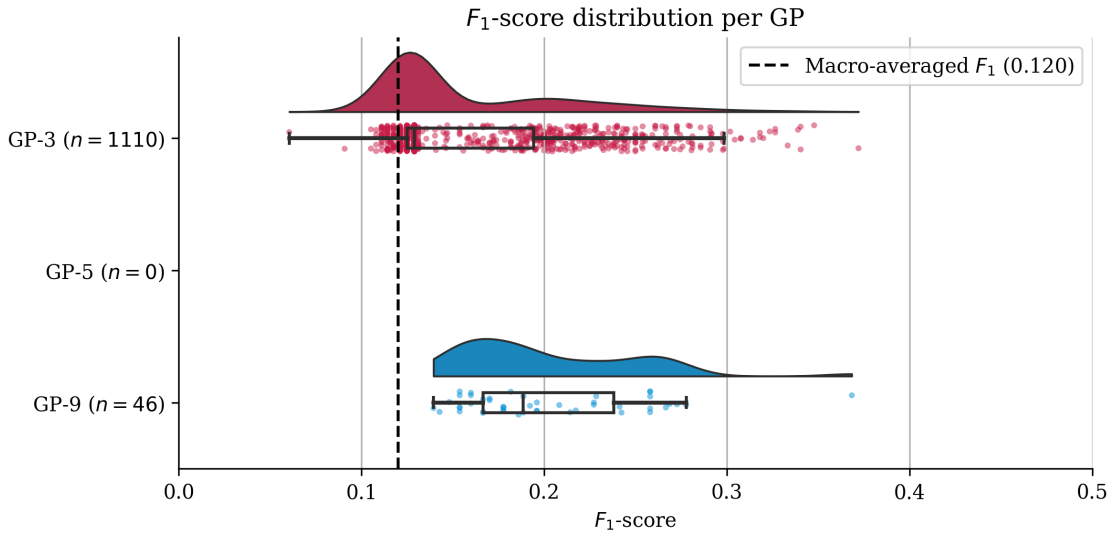

Figure 6.12: Raincloud plot visualising the distribution of the $\mathcal{F}_1$-score among the three process models using the LCS algorithm. No distribution can be shown for GP-5 as no matches were found for this process model.

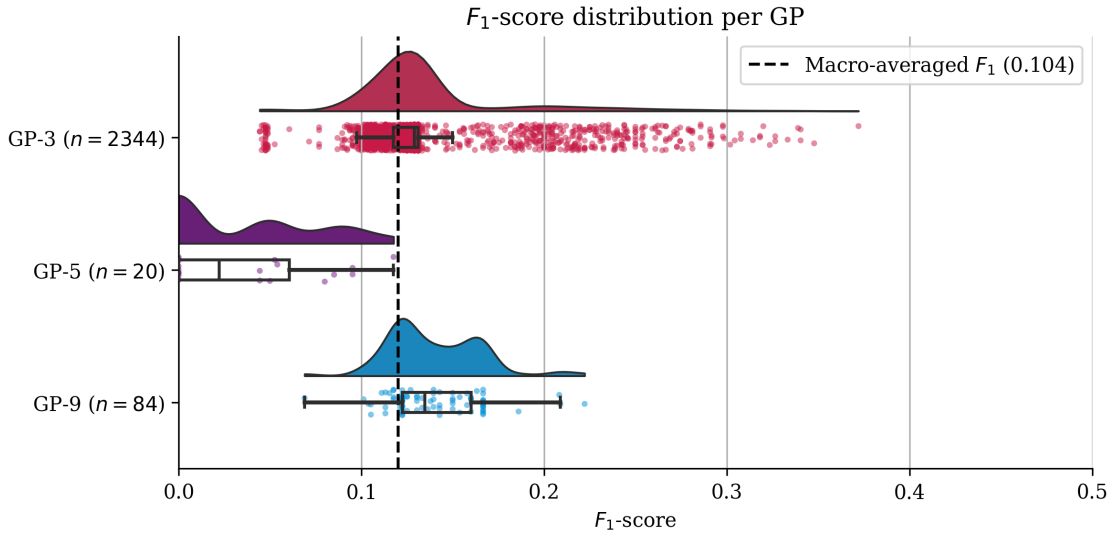

Figure 6.13: Raincloud plot visualising the distribution of the $\mathcal{F}_1$-score among the three process models using the TBR algorithm.

6.13 Expert Evaluation Results

Date	Train	Exp. 1	Exp. 2	LCS	TBR	Mode(s)
2023-09-06	45702	Amb.	GP-5	Amb. (GP-3: 49%, GP-5: 2%, GP-9: 49%)	GP-3	Amb.
2023-09-06	93244	Amb.	Amb.	GP-3	GP-5	Amb.
2023-09-08	932101	GP-9	GP-9	Amb. (GP-3: 33%, GP-5: 33%, GP-9: 33%)	Amb. (GP-3: 33%, GP-5: 33%, GP-9: 33%)	Amb., GP-9
2023-09-09	93229	GP-9	GP-9	GP-3	GP-3	GP-3, GP-9
2023-09-12	93205	GP-9	GP-9	Amb. (GP-3: 33%, GP-5: 33%, GP-9: 33%)	GP-3	GP-9
2023-10-02	127	Amb.	Amb.	Amb. (GP-3: 50%, GP-5: 1%, GP-9: 50%)	GP-3	Amb.
2023-10-05	43370	Amb.	GP-5	Amb. (GP-3: 42%, GP-5: 16%, GP-9: 42%)	GP-5	Amb., GP-5
2023-10-14	221	Amb.	Amb.	Amb. (GP-3: 50%, GP-5: 1%, GP-9: 50%)	Amb. (GP-3: 33%, GP-5: 33%, GP-9: 33%)	Amb.
2023-10-17	105	Amb.	Amb.	Amb. (GP-3: 50%, GP-5: 1%, GP-9: 50%)	Amb. (GP-3: 33%, GP-5: 33%, GP-9: 33%)	Amb.
2023-10-23	222	Amb.	Amb.	GP-3	Amb. (GP-3: 33%, GP-5: 33%, GP-9: 33%)	Amb.
2023-10-30	120	Amb.	Amb.	GP-3	GP-3	Amb., GP-3
2023-11-09	46256	GP-3	Amb.	Amb. (GP-3: 33%, GP-5: 33%, GP-9: 33%)	GP-3	Amb., GP-3
2023-11-27	41790	Amb.	GP-5	Amb. (GP-3: 42%, GP-5: 16%, GP-9: 42%)	Amb. (GP-3: 33%, GP-5: 33%, GP-9: 33%)	Amb.
2023-12-03	52411	GP-3	Amb.	GP-3	GP-9	GP-3
2023-12-12	44363	Amb.	Amb.	Amb. (GP-3: 49%, GP-5: 2%, GP-9: 49%)	GP-3	Amb.
2023-12-20	82301	GP-9	GP-9	GP-3	GP-3	GP-3, GP-9
2023-12-29	125	Amb.	Amb.	Amb. (GP-3: 49%, GP-5: 2%, GP-9: 49%)	GP-3	Amb.
2023-12-30	128	Amb.	Amb.	GP-3	GP-3	Amb., GP-3

Table 6.6: All samples annotated by both domain experts and the LCS and TBR algorithms. The softmax function is applied to the scores achieved for sequences classified as ambiguous. The mode or modes across the four classification results are shown in the last column.

6.14 Semantic Similarity

To calculate the semantic similarity between two sequences, the sequences need to be converted into numerical representations. Subsequently, the similarity between numerical representations can be calculated via various distance functions. For example, Reimers and Gurevych (2019) uses the Bidirectional Encoder Representations from Transformers (BERT) architecture of Devlin et al. (2018) to embed input strings in vector space. BERT is a transformer-based neural network that is pre-trained on a large corpus of text. The self-supervised network is trained to predict the next word in a sentence given the previous words, i.e. the context. In doing so, the network learns to use this context to place semantically similar words close together within the vector representation. Subsequently, it can convert a new input sequence into a vector representation as it has previously learned the context of the elements in that sequence.

Similar sequences will thus ideally have similar vectors and therefore a small distance. Reimers and Gurevych (2019) calculate the distance between two vectors using the cosine distance or cosine similarity measure (Li & Han, 2013). The cosine distance is defined as the dot product between two normalised vectors $\vec{u}$ and $\vec{v}$:

$$\text{cosine distance}(\vec{u}, \vec{v}) = \frac{\vec{u} \bullet \vec{v}}{||\vec{u}|| \cdot ||\vec{v}||} \tag{6.1}$$

Similar to the method proposed by Reimers and Gurevych (2019), are the word2vec and Global Vectors (GloVe) algorithms. Both algorithms convert words into a vector representation and calculate their similarity using the cosine distance. The word2vec algorithm by Mikolov et al. (2013) uses a two-layer neural network to represent words as vectors. The GloVe algorithm by Pennington et al. (2014) represents words as vectors using a co-occurrence matrix. Both algorithms are predecessors of BERT and have fewer parameters than BERT making them generally faster to train and use.

Since the development of BERT, many large language models (LLM) have been developed. Popular models include the multilingual XLM-RoBERTa model by Conneau et al. (2019), the 176 billion parameter BLOOM language model by Scao et al. (2023) that was trained on a supercomputer for 3.5 months, Meta's LLaMA model and the GPT family of models from Microsoft and OpenAI including the recent arrival of ChatGPT. These LLMs are all pre-trained on large corpora of text and can be fine-tuned on smaller datasets to improve performance. The LLMs are often used for various NLP tasks such as text generation and classification.

The vast majority of models focus on the English language (Vanroy, 2023). Since both the described processes and the dataset columns are in Dutch, there is a need for a LLM that is fine-tuned for the Dutch language. In this way, vector representations are more in line with the semantics of the Dutch language. LLMs trained for the Dutch language include BERTje from de Vries et al. (2019), RobBERT from Delobelle et al. (2020) and GEITje from Rijgersberg and Lucassen (2023).

Vanroy (2023) conducted a review of several state-of-the-art (SOTA) LLMs to compare their performance on various Dutch benchmarks for NLP tasks. The review included the aforementioned LLaMA and GPT models, as well as others. The test data used for the benchmarks was however machine-translated to Dutch as no Dutch benchmarks were available. Models not fine-tuned for Dutch performed surprisingly well on the benchmarks, with Zephyr by Tunstall et al. (2023) performing best overall. However, interactions of the author with the best-performing model fine-tuned for Dutch, GEITje, showed a more natural conversation than interactions with models not fine-tuned on Dutch text. Non-fine-tuned models "read more like a poor translation from English to Dutch" (Vanroy, 2023, p. 10). It is noteworthy that Zephyr and GEITje are both based on the same Mistral 7B model architecture by Jiang et al. (2023).

6.14.1 Proposed Methodology

Part of designing the rules for the RBS is finding and investigating the most relevant columns for each activity. The rules are then designed based on these selected columns. This can be a

time-consuming task for the domain experts designing the rules, especially when the data consists of a large number of columns. To facilitate this process, an AI system is proposed to suggest the most relevant columns for each activity. This is done based on the semantic similarity between the description of an activity and the descriptions of all columns in the dataset. The hypothesis is that the most relevant column has a description most similar to the description of the activity. This approach could not only save time for the domain experts but also reduce the chance of overlooking or missing potentially relevant columns. As mentioned above, semantic similarity algorithms consist of two parts:

1. Converting the input data to a numerical representation.
   - Often in the form of a vector or matrix.
   - This includes the pre-processing of the input data.
2. Calculating the similarity or distance between two numerical representations using a similarity function.

The description of a column c_k can be retrieved via $d(c_k)$, which is elaborated on in subsection 3.1.1. After converting the descriptions of a target activity a and all columns into numerical representations and calculating their similarity, the most similar columns to an activity can be retrieved.

This process can be formalised as follows: given an activity description $d(a)$ and a sequence of column descriptions $D(\mathcal{M})$, the algorithm converts $d(a)$ to a numerical representation $v(d(a))$ and each column description $d(c_k) \in D(\mathcal{M})$ into a numerical representation $v(d(c_k))$. The similarity between $v(d(a))$ and each $v(d(c_k))$ is then computed using a similarity function $sim(v(d(a)), v(d(c_k)))$. The column descriptions with the highest similarity to $d(a)$ are considered the most relevant columns for a. The columns are presented to the domain expert in descending order of similarity.

The activity description $d(a)$ is any ordered sequence of characters with a length greater than zero: a non-empty string. Each column description $d(c_k)$ is likewise defined as an ordered sequence of characters with a length greater than zero. A popular similarity function is the cosine similarity metric, as presented in Eq. (6.1). It has a range between -1 and 1. A value of 1 indicates that the two vectors are identical, while a value of -1 indicates that the two vectors are completely dissimilar. A value of 0 indicates that the two vectors are orthogonal. The above process is captured in the pseudocode of Algorithm 2.

Algorithm 2 Semantic Similarity Algorithm

$d(a) \leftarrow$ target activity description ▷ Activity for which to find the most similar columns
$D(\mathcal{M}) \leftarrow [d(c_1), d(c_2), \ldots, d(c_k)]$ ▷ Sequence of column descriptions
$v \leftarrow$ vectorisation function ▷ Convert input characters to a numerical representation
$sim \leftarrow$ similarity function
$S \leftarrow [\ldots]$ ▷ Sequence to store calculated similarity scores

$v(d(a)) \leftarrow$ convert $d(a)$ to a numerical representation ▷ Only done once for $d(a)$
for all column descriptions $d(c_k) \in D(\mathcal{M})$ **do**
    $v(d(c_k)) \leftarrow$ convert $d(c_k)$ to a numerical representation
    $s_i \leftarrow sim(v(d(a)), v(d(c_k)))$ ▷ Calculate similarity between $d(a)$ and $d(c_k)$
    Append s_i to S
end for

$D(\mathcal{M})_{\text{sorted}} \leftarrow$ sort $D(\mathcal{M})$ by S in descending order ▷ Sort column descr. by sim. score
return $D(\mathcal{M})_{\text{sorted}}$

The algorithms discussed above to convert data into a numerical representation focus on textual data. However, the methodology described can be applied to any type of data by using the appropriate algorithm to convert the input into a numerical representation suitable for the chosen similarity function.

6.14.2 Quantifying Semantic Similarity Results

Suppose that the set of columns used to design the rules for an activity a_i is represented as $C(a_i) = \{c_1, c_2, \ldots, c_n\}$. Their corresponding descriptions are represented as $D(C(a_i)) = \{d(c_1), d(c_2), \ldots, d(c_n)\}$. This set thus consists of all relevant column descriptions for the activity a_i.

$D(C(a_i))$ can then be compared to the top-n most similar elements of the ordered sequence of column descriptions $D(\mathcal{M})_{ordered}$ determined for a_i. Here, n is the number of columns used by a_i and is equal to the number of descriptions in $D(C(a_i))$. As each activity can use a different number of columns, n can differ for each activity. For instance, if a_i uses three columns, $n = 3$ and we compare $D(C(a_i))$ to the top three most similar elements of $D(\mathcal{M})_{ordered}$.

The semantic similarity algorithm is quantified using the harmonic mean of precision and recall: the F_1-score. It is defined as follows:[1]

$$P = \frac{TP}{TP + FP} \qquad (6.2) \qquad R = \frac{TP}{TP + FN} \qquad (6.3) \qquad F_1 = 2 \cdot \frac{P \cdot R}{P + R} \qquad (6.4)$$

Here, TP is the number of true positives, FP is the number of false positives and FN is the number of false negatives. A true positive is a column present in both $D(C(a_i))$ and $D(\mathcal{M})_{ordered}$. A false positive is a column present in $D(\mathcal{M})_{ordered}$ but not in $D(C(a_i))$. A false negative is a column present in $D(C(a_i))$ but not in $D(\mathcal{M})_{ordered}$. As with Eq. (3.7), the F_1-score has a range of $[0, 1]$, where a value of 1 indicates a perfect match. Ideally, the F_1-score should be as close to 1 as possible indicating that the semantic similarity algorithm can retrieve all relevant columns for activity a_i.

As the column descriptions in $D(C(a_i))$ have no order, we do not need to take this into account when comparing $D(C(a_i))$ to the top-n of $D(\mathcal{M})_{ordered}$.

[1] Note the difference between this formula and the harmonic mean of fitness and precision in Eq. (3.7), which is written subtly differently as $\mathcal{F}_1$ instead of F_1 to avoid confusion.

Acronyms

Ac Abcoude. 20, 43
Aco Abcoude Overloopwissels. 7, 20, 43
AI Artificial Intelligence. v, 16, 25, 57
Asa Amsterdam Amstel. 43
Asb Amsterdam Bijlmer. 43
Asd Amsterdam. 1, 2, 4–7, 18, 27, 36, 40, 43
Asdar Amsterdam ArenA. 43
Asdz Amsterdam Zuid WTC. 43
Ashd Amsterdam Holendrecht. 43
ATB automatische treinbeïnvloeding. 4–6, 27, 42, 45

BERT Bidirectional Encoder Representations from Transformers. 56
Bkl Breukelen. 7, 20, 43
Bkla Breukelen Aansluiting. 7, 20, 43
BLOOM BigScience Large Open-science Open-access Multilingual. 56
BPMN Business Process Model and Notation. 8, 9, 18, 28, 34, 36, 47–49

Ddm De Diemen. 43
Dmnz Diemen Zuid. 7, 43
DRP dienstregelpunt. 7, 17, 18, 20, 27, 43, 44
Dvaw Duivendrecht Aansluiting West. 43
Dvaz Duivendrecht Aansluiting Zuid. 43
Dvd Duivendrecht. 43

EG eerste generatie. 6
EI European Instruction. 45
EoA End of Authority. 42
ERTMS European Rail Traffic Management System. 4, 5
ETCS European Train Control System. 5, 6, 17, 18, 21, 22, 27, 28, 42
EU European Union. 5

FS Full Supervision. 6, 45, 46
FTE Full-time equivalent. 1

GloVe Global Vectors. 56
GP gebruikersproces. 1, 2, 5, 6, 8, 18, 19, 22, 23, 28, 31, 32, 34–36, 41–49, 54, 55
GPT Generative Pre-trained Transformer. 56

Hmlba Harmelen-Breukelen Aansluiting. 43

IAA inter-annotator agreement. 25, 32, 33, 35, 37
ICNG Intercity Nieuwe Generatie. 5

L2 Level 2. 6, 19, 41, 42, 45
LCS longest common subsequence. ii, 14, 22–26, 31–35, 53–55
LLaMA Large Language Model Meta AI. 56
LLM large language model. 56
Lls Lelystad. 2, 40
LSTM Long Short-Term Memory. 14

MA Movement Authority. 18, 41, 46

Mas Maarssen. 20, 43
Mdsa Muiderstraatweg Aansluiting. 43

N/A not applicable. 31, 32
NLP natural language processing. 13, 25, 56
NS Nederlandse Spoorwegen. 4, 5
NTC National Train Control. 5, 6, 40–42, 45

OCR optical character recognition. 14
OS On Sight. 21, 22, 45, 46

PCA Principal Component Analysis. 14
pdf portable document format. 9
PoC proof of concept. ii, 2–4, 15–18, 21, 23, 34–37

Rai Amsterdam RAI. 43
RBC Radio Block Centre. 42, 45, 46
RBS rule-based system. ii, 13, 20, 21, 29, 34, 35, 37, 51, 52, 56
ROZ rijden op zicht. 41
RQ research question. 3
RQ1 What data sources provide predictive information about the user processes?. 3, 34
RQ2 How can the description of a user process and its activities be encoded into a formal definition?. 3, 34
RQ3 How can the formal definition be used to extract a sequence of activities from data?. 3, 34
RQ4 Which algorithm is best in identifying user processes in the retrieved sequences of activities using the formal definitions?. 3, 31, 34
RQ5 What evaluation metric is best to determine the performance of the system?. 3, 35

SB Stand By. 46
SD Standard Deviation. 27, 31
SN National System. 45
SNG Sprinter Nieuwe Generatie. 5
SoM Start of Mission. 29, 37
SOTA state-of-the-art. 56
SR Staff Responsible. 45
STM Specific Transmission Module. 45
STS stoptonend sein. 29, 42, 45, 46

TB Terabyte. 7
TBR token-based replay. ii, 12, 23–26, 31–36, 54, 55
Trdl treindienstleider. 1, 6, 18, 28, 29, 34, 36

Ut Utrecht. 1, 2, 4–7, 18, 20, 27, 36, 40, 43
Utma Utrecht-Maarsen Aansluiting. 43
Utzl Utrecht Zuilen. 20, 43
UU Utrecht University. v, 16

Vspa Venserpolder Aansluiting. 7, 43

Zl Zwolle. 2, 40

References

Alberga, C. N. (1967). String similarity and misspellings. *Communications of the ACM*, *10*(5), 302–313. https://dl.acm.org/doi/pdf/10.1145/363282.363326

Artstein, R., & Poesio, M. (2008). Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics*, *34*(4), 555–596. https://doi.org/10.1162/coli.07-034-R2

Berti, A., & van der Aalst, W. M. (2019). Reviving token-based replay: Increasing speed while improving diagnostics. *ATAED@ Petri Nets/ACSD*, *2371*, 87–103. https://ceur-ws.org/Vol-2371/ATAED2019-87-103.pdf

Bonnyman, A. M., Webber, C. E., Stratford, P. W., & MacIntyre, N. J. (2012). Intrarater reliability of dual-energy x-ray absorptiometry-based measures of vertebral height in postmenopausal women. *J Clin Densitom*, *15*(4), 405–412. https://pubmed.ncbi.nlm.nih.gov/22578772/

Buijs, J. C. A. M., et al. (2010). Mapping data sources to xes in a generic way. *Department of Mathematics and Computer Science, Eindhoven University of Technology*. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=7fb657506c5de81de8d722b857bb0c3adbc9a6ae

Carmona, J., van Dongen, B., Solti, A., & Weidlich, M. (2018, November). *Conformance checking: Relating processes and models, relating processes and models*. Springer. https://doi.org/10.1007/978-3-319-99414-7

Chaabi, Y., & Ataa Allah, F. (2022). Amazigh spell checker using Damerau-Levenshtein algorithm and N-gram. *Journal of King Saud University - Computer and Information Sciences*, *34*(8, Part B), 6116–6124. https://doi.org/10.1016/j.jksuci.2021.07.015

Cheatham, M., & Hitzler, P. (2013). String similarity metrics for ontology alignment. *The Semantic Web–ISWC 2013: 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II 12*, 294–309. https://corescholar.libraries.wright.edu/cgi/viewcontent.cgi?article=1174&context=cse

Chinchor, N., & Sundheim, B. M. (1993). MUC-5 evaluation metrics. *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993*. https://aclanthology.org/M93-1007.pdf

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, *20*(1), 37–46. https://doi.org/10.1177/001316446002000104

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2019). Unsupervised Cross-lingual Representation Learning at Scale. *CoRR*, *abs/1911.02116*. http://arxiv.org/abs/1911.02116

Cormode, G., & Muthukrishnan, S. (2007). The string edit distance matching problem with moves. *ACM Transactions on Algorithms (TALG)*, *3*(1), 1–19. https://dl.acm.org/doi/pdf/10.1145/1186810.1186812

Dahlmeier, D., Ng, H. T., & Wu, S. M. (2013, June). Building a large annotated corpus of learner English: The NUS corpus of learner English. In J. Tetreault, J. Burstein & C. Leacock (Eds.), *Proceedings of the eighth workshop on innovative use of NLP for building educational applications* (pp. 22–31). Association for Computational Linguistics. https://aclanthology.org/W13-1703

Dakic, D., Stefanovic, D., Lolic, T., Narandzic, D., & Simeunovic, N. (2020). Event Log Extraction for the Purpose of Process Mining: A Systematic Literature Review. In G. Prostean, J. J. Lavios Villahoz, L. Brancu & G. Bakacsi (Eds.), *Innovation in sustainable management and entrepreneurship* (pp. 299–312). Springer International Publishing. https://rdcu.be/dyeGT

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, *7*(3), 171–176. https://dl.acm.org/doi/10.1145/363958.363994

Delobelle, P., Winters, T., & Berendt, B. (2020). RobBERT: a Dutch RoBERTa-based Language Model. *CoRR*, *abs/2001.06286*. https://arxiv.org/abs/2001.06286

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, *abs/1810.04805*. http://arxiv.org/abs/1810.04805

de Vries, W., van Cranenburgh, A., Bisazza, A., Caselli, T., van Noord, G., & Nissim, M. (2019). BERTje: A Dutch BERT Model. *CoRR*, *abs/1912.09582*. http://arxiv.org/abs/1912.09582

Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 1285–1298. https://doi.org/10.1145/3133956.3134015

Duignan, B. (2024, January 2). Occam's Razor. Retrieved February 8, 2024, from https://www.britannica.com/topic/Occams-razor

Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2018). *Fundamentals of business process management*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-56509-4

ERTMS NL. (2022). *Planning ERTMS*. Retrieved January 11, 2024, from https://www.ertms.nl/over-ertms/planning/planning-groot/default.aspx

European Union Agency for Railways. (2023, October). *Control Command and Signalling TSI*. Retrieved January 2, 2024, from https://www.era.europa.eu/domains/technical-specifications-interoperability/control-command-and-signalling-tsi_en

Goddard, K., Roudsari, A., & Wyatt, J. C. (2011). Automation bias: a systematic review of frequency, effect mediators, and mitigators. *Journal of the American Medical Informatics Association*, *19*(1), 121–127. https://doi.org/10.1136/amiajnl-2011-000089

González López de Murillas, E., Reijers, H. A., & van der Aalst, W. M. P. (2019). Connecting databases with process mining: A meta model and toolset. *Software & Systems Modeling*, *18*(2), 1209–1247. https://doi.org/10.1007/s10270-018-0664-7

Günther, C. W., & Rozinat, A. (2012). Disco: Discover your processes. *BPM (Demos)*, *940*(1), 40–44. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=e75b8e646f8bc77302bd17801e8d15aa9df86734#page=46

Haldar, R., & Mukhopadhyay, D. (2011). Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach. *CoRR*, *abs/1101.1232*. http://arxiv.org/abs/1101.1232

Hall, P. A., & Dowling, G. R. (1980). Approximate string matching. *ACM computing surveys (CSUR)*, *12*(4), 381–402. https://dl.acm.org/doi/pdf/10.1145/356827.356830

Hamming, R. W. (1950). Error detecting and error correcting codes. *The Bell System Technical Journal*, *29*(2), 147–160. https://doi.org/10.1002/j.1538-7305.1950.tb00463.x

Hunt, J. W., & Szymanski, T. G. (1977). A fast algorithm for computing longest common subsequences. *Communications of the ACM*, *20*(5), 350–353. https://dl.acm.org/doi/pdf/10.1145/359581.359603

Infrasite. (2021, March 17). *Dienstregelpunt — Infrasite*. Retrieved January 20, 2024, from https://www.infrasite.nl/glossary/dienstregelpunt/

Jaccard, P. (1912). THE DISTRIBUTION OF THE FLORA IN THE ALPINE ZONE. *New Phytologist*, *11*(2), 37–50. https://doi.org/10.1111/j.1469-8137.1912.tb05611.x

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., & Sayed, W. E. (2023). Mistral 7B. https://arxiv.org/abs/2310.06825

Lacoste, A., Luccioni, A., Schmidt, V., & Dandres, T. (2019). Quantifying the Carbon Emissions of Machine Learning. *CoRR*, *abs/1910.09700*. http://arxiv.org/abs/1910.09700

Landauer, M., Onder, S., Skopik, F., & Wurzenberger, M. (2023). Deep learning for anomaly detection in log data: A survey. *Machine Learning with Applications*, *12*, 100470. https://doi.org/10.1016/j.mlwa.2023.100470

Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, *33*(1), 159–174. Retrieved June 23, 2024, from http://www.jstor.org/stable/2529310

Levenshtein, V. I., et al. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, *10*(8), 707–710. https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf

Li, B., & Han, L. (2013). Distance Weighted Cosine Similarity Measure for Text Classification. In H. Yin, K. Tang, Y. Gao, F. Klawonn, M. Lee, T. Weise, B. Li & X. Yao (Eds.), *Intelligent data engineering and automated learning – ideal 2013* (pp. 611–618). Springer Berlin Heidelberg. https://rdcu.be/dAgAl

Lin, Q., Zhang, H., Lou, J.-G., Zhang, Y., & Chen, X. (2016). Log clustering based problem identification for online service systems. *Proceedings of the 38th International Conference on Software Engineering Companion*, 102–111. https://doi.org/10.1145/2889160.2889232

Lu, X., Tabatabaei, S. A., Hoogendoorn, M., & Reijers, H. A. (2019). Trace Clustering on Very Large Event Data in Healthcare Using Frequent Sequence Patterns. In T. Hildebrandt, M. van Dongen Boudewijn F.and Röglinger & J. Mendling (Eds.), *Business process management* (pp. 198–215). Springer International Publishing. https://link.springer.com/chapter/10.1007/978-3-030-26619-6_14

Măruşter, L., Weijters, A. J. M. M., Van Der Aalst, W. M. P., & Van Den Bosch, A. (2006). A Rule-Based Approach for Process Discovery: Dealing with Noise and Imbalance in Process Logs. *Data Mining and Knowledge Discovery*, *13*(1), 67–87. https://doi.org/10.1007/s10618-005-0029-z

Masri, N., Sultan, Y. A., Akkila, A. N., Almasri, A., Ahmed, A., Mahmoud, A. Y., Zaqout, I., & Abu-Naser, S. S. (2019). Survey of rule-based systems. *International Journal of Academic Information Systems Research (IJAISR)*, *3*(7), 1–23. https://www.academia.edu/download/91489715/10-08-2019-05.pdf

McHugh, M. L. (2012). Interrater reliability: The kappa statistic. *Biochem Med (Zagreb)*, *22*(3), 276–282. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3900052/

Middelraad, P. (2000). Voorgeschiedenis, Ontstaan en Evolutie van het NS-Lichtseinstelsel. Retrieved January 20, 2024, from https://www.irse.nl/boeken/page35/files/NS%20Lichtseinstelsel.pdf

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. https://arxiv.org/abs/1301.3781

Min, K., Yoon, J., Kang, M., Lee, D., Park, E., & Han, J. (2023). Detecting depression on video logs using audiovisual features. *Humanities and Social Sciences Communications*, *10*(1), 788. https://www.nature.com/articles/s41599-023-02313-6

Naidu, G., Zuva, T., & Sibanda, E. M. (2023). A Review of Evaluation Metrics in Machine Learning Algorithms. In R. Silhavy & P. Silhavy (Eds.), *Artificial intelligence application in networks and systems* (pp. 15–25). Springer International Publishing. https://rdcu.be/dx3m3

Nederlandse Spoorwegen. (2022, September). *NS Start Met Implementatie Nieuw Beveiligingssysteem Treinen*. Retrieved January 20, 2024, from https://nieuws.ns.nl/ns-start-met-implementatie-nieuw-beveiligingssysteem-treinen/

Nederlandse Spoorwegen. (2023a, April). *Nieuwe NS-Intercity debuteert tussen Amsterdam en Rotterdam*. Retrieved January 20, 2024, from https://nieuws.ns.nl/nieuwe-ns-intercity-debuteert-tussen-amsterdam-en-rotterdam/

Nederlandse Spoorwegen. (2023b, May). *Eerste reizigers stappen in laatste nieuwe Sprinter*. Retrieved January 20, 2024, from https://nieuws.ns.nl/eerste-reizigers-stappen-in-laatste-nieuwe-sprinter/

Object Management Group. (2011, January). Business Process Model and Notation (BPMN), Version 2.0. *Object Management Group*. http://www.omg.org/spec/BPMN/2.0

Operationeel Kenniscentrum ERTMS. (2023). *Gebruikersprocessen - ERTMS NL*. Retrieved January 11, 2024, from https://www.ertms.nl/oke/wat-we-leveren/gebruikersprocessen/default.aspx

Opitz, J., & Burst, S. (2019). Macro F1 and macro F1. *CoRR*, *abs/1911.03347*. http://arxiv.org/abs/1911.03347

Papers With Code. (2024). *Papers With Code - Anomaly detection.* Retrieved January 21, 2024, from https://paperswithcode.com/task/anomaly-detection

Pennington, J., Socher, R., & Manning, C. (2014, October). GloVe: Global Vectors for Word Representation. In A. Moschitti, B. Pang & W. Daelemans (Eds.), *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532–1543). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162

Peterson, J. L. (1977). Petri nets. *ACM Comput. Surv.*, *9*(3), 223–252. https://doi.org/10.1145/356698.356702

Petri, C. A. (1962). Kommunikation mit automaten. *Schriften des Rheinisch-Westfälischen Instituts für Instrumentelle Mathematik, Bonn, Germany.* https://edoc.sub.uni-hamburg.de/informatik/volltexte/2011/160/

Preece, A. D., Shinghal, R., & Batarekh, A. (1992). Principles and practice in verifying rule-based systems. *The Knowledge Engineering Review*, *7*(2), 115–141. https://doi.org/10.1017/S026988890000624X

ProRail. (2019). *ERTMS: het digitale spoorplatform.* Retrieved January 20, 2024, from https://www.prorail.nl/programmas/ertms

ProRail. (2022, August). *Oefenen met ERTMS: van papieren plan naar werk aan het spoor.* https://www.prorail.nl/nieuws/oefenen-met-ertms-van-papieren-plan-naar-werk-aan-het-spoor

ProRail. (2023, November). *Netverklaring 2024* (tech. rep. No. T20180019-117460140-6530). Utrecht, the Netherlands. https://www.prorail.nl/siteassets/homepage/samenwerken/vervoerders/documenten/netwerkverklaring-2024/netverklaring-2024-versie-1.3.pdf

ProRail. (2024, April 4). *Jaarverslag 2023.* Retrieved May 10, 2024, from https://www.jaarverslagprorail.nl/FbContent.ashx/pub_1002/downloads/v240418111216/ProRail_Jaarverslag_2023.pdf

ProRail Assetmanagement. (2023, September). *Gebruikersprocessen 'rijden met treinen'- op dual signalling baanvakken Amsterdam - Utrecht en de Hanzelijn* (tech. rep. No. RLN60561-6). https://www.ertms.nl/bibliotheek-ertms/gebruikersprocessen_doc/handlerdownloadfiles.ashx?idnv=2424459

ProRail ERTMS Integratie Lab. (2020, August). *The most complete ERTMS integration test facility.* Retrieved January 20, 2024, from https://www.prorail.nl/siteassets/homepage/samenwerken/vervoerders/documenten/folder-ertms-integratie-lab_web.pdf

Rattanasawad, T., Buranarach, M., Saikaew, K. R., & Supnithi, T. (2018). A comparative study of rule-based inference engines for the semantic web. *IEICE TRANSACTIONS on Information and Systems*, *101*(1), 82–89. https://www.jstage.jst.go.jp/article/transinf/E101.D/1/E101.D_2017SWP0004/_pdf

Reimers, N., & Gurevych, I. (2019, November). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In K. Inui, J. Jiang, V. Ng & X. Wan (Eds.), *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)* (pp. 3982–3992). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1410

Rijgersberg, E., & Lucassen, B. (2023, December). GEITje: een groot open Nederlands taalmodel. https://github.com/Rijgersberg/GEITje

Rijksoverheid. (2023, October). *Wetten.nl - regeling - Regeling Spoorverkeer - BWBR0017707* (tech. rep. No. BWBR0017707). https://wetten.overheid.nl/jci1.3:c:BWBR0017707&bijlage=4&z=2023-10-04&g=2023-10-04

Rodrıguez, C., Engel, R., Kostoska, G., Daniel, F., Casati, F., Aimar, M., et al. (2012). Eventifier: Extracting process execution logs from operational databases. *Proceedings of the demonstration track of BPM*, *940*, 17–22. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=e75b8e646f8bc77302bd17801e8d15aa9df86734#page=23

Rozinat, A., & van der Aalst, W. M. P. (2006). Conformance Testing: Measuring the Fit and Appropriateness of Event Logs and Process Models. In C. J. Bussler & A. Haller (Eds.),

Business process management workshops (pp. 163–176). Springer Berlin Heidelberg. https://research.tue.nl/files/1837465/362106957468163.pdf

Rozinat, A., & van der Aalst, W. M. P. (2008). Conformance checking of processes based on monitoring real behavior. *Information Systems*, *33*(1), 64–95. https://doi.org/10.1016/j.is.2007.07.001

Santoso, P., Yuliawati, P., Shalahuddin, R., & Wibawa, A. P. (2019). Damerau levenshtein distance for Indonesian spelling correction. *J. Inform*, *13*(2), 11. https://pdfs.semanticscholar.org/920f/50778988a9d76d56e6b20c17103ad3a00ebf.pdf

Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., Tow, J., Rush, A. M., Biderman, S., Webson, A., Ammanamanchi, P. S., Wang, T., Sagot, B., Muennighoff, N., del Moral, A. V., ... Wolf, T. (2023). BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. https://arxiv.org/abs/2211.05100

Stein Dani, V., Leopold, H., van der Werf, J. M. E. M., Lu, X., Beerepoot, I., Koorn, J. J., & Reijers, H. A. (2022). Towards Understanding the Role of the Human in Event Log Extraction. In A. Marrella & B. Weber (Eds.), *Business process management workshops* (pp. 86–98). Springer International Publishing. https://link.springer.com/chapter/10.1007/978-3-030-94343-1_7

Stein Dani, V., Leopold, H., van der Werf, J. M. E. M., & Reijers, H. A. (2023). Supporting Event Log Extraction Based on Matching. In C. Cabanillas, N. F. Garmann-Johnsen & A. Koschmider (Eds.), *Business process management workshops* (pp. 322–333). Springer International Publishing. https://drive.google.com/file/d/12FtmLB5AOn2SpT8NzjLcWKr3f41RhOqi/view

Tunstall, L., Beeching, E., Lambert, N., Rajani, N., Rasul, K., Belkada, Y., Huang, S., von Werra, L., Fourrier, C., Habib, N., Sarrazin, N., Sanseviero, O., Rush, A. M., & Wolf, T. (2023). Zephyr: Direct Distillation of LM Alignment. https://arxiv.org/abs/2310.16944

Utrecht University. (2023, July 31). *Guidance for Research Master Thesis Students*. Retrieved January 25, 2024, from https://www.uu.nl/en/research/institute-of-information-and-computing-sciences/ethics-and-privacy/guidance-for-research-master-thesis-students

Van der Aa, H., Leopold, H., & Reijers, H. A. (2017). Comparing textual descriptions to process models–the automatic detection of inconsistencies. *Information Systems*, *64*, 447–460. https://www.sciencedirect.com/science/article/pii/S0306437915301666

Van der Aalst, W. M. P. (2010). Process Discovery: Capturing the Invisible. *IEEE Computational Intelligence Magazine*, *5*(1), 28–41. https://doi.org/10.1109/MCI.2009.935307

Van der Aalst, W. M. P. (2012). Process Mining: Overview and Opportunities. *ACM Trans. Manage. Inf. Syst.*, *3*(2). https://doi.org/10.1145/2229156.2229157

Van der Aalst, W. M. P., Adriansyah, A., de Medeiros, A. K. A., Arcieri, F., Baier, T., Blickle, T., Bose, J. C., van den Brand, P., Brandtjen, R., Buijs, J., Burattin, A., Carmona, J., Castellanos, M., Claes, J., Cook, J., Costantini, N., Curbera, F., Damiani, E., de Leoni, M., ... Wynn, M. (2012). Process Mining Manifesto. In F. Daniel, K. Barkaoui & S. Dustdar (Eds.), *Business process management workshops* (pp. 169–194). Springer Berlin Heidelberg. https://link.springer.com/chapter/10.1007/978-3-642-28108-2_19

Van der Aalst, W. M. P., Weijters, T., & Maruster, L. (2004). Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, *16*(9), 1128–1142. https://doi.org/10.1109/TKDE.2004.47

Van der Loo, M. P., et al. (2014). The stringdist package for approximate string matching. *R J.*, *6*(1), 111. https://journal.r-project.org/archive/2014-1/loo.pdf

Van Dongen, B. F., de Medeiros, A. K. A., Verbeek, H. M. W., Weijters, A. J. M. M., & van der Aalst, W. M. P. (2005). The ProM Framework: A New Era in Process Mining Tool Support. In G. Ciardo & P. Darondeau (Eds.), *Applications and theory of petri nets 2005* (pp. 444–454). Springer Berlin Heidelberg. https://link.springer.com/chapter/10.1007/11494744_25

Vanroy, B. (2023). Language Resources for Dutch Large Language Modelling. https://arxiv.org/abs/2312.12852

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Weijters, A., van der Aalst, W. M. P., & Medeiros, A. (2006, January). *Process Mining with the Heuristics Miner-algorithm* (Vol. 166). https://research.tue.nl/files/2388011/615595.pdf

Xu, W., Huang, L., Fox, A., Patterson, D., & Jordan, M. (2009). Largescale system problem detection by mining console logs. *Proceedings of SOSP'09*. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-103.pdf

Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., Xie, C., Yang, X., Cheng, Q., Li, Z., Chen, J., He, X., Yao, R., Lou, J.-G., Chintalapati, M., Shen, F., & Zhang, D. (2019). Robust log-based anomaly detection on unstable log data. *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 807–817. https://doi.org/10.1145/3338906.3338931