Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik

Resolving Partially Ordered Traces Using Deep Learning

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B. Sc.)

eingereicht von:	Glenn Dittmann
geboren am:	13.06.1993
geboren in:	Berlin
Gutachter/innen:	Prof. Dr. Matthias Weidlich
	Prof. Dr. Han van der Aa

eingereicht am: .... verteidigt am: ....

Abstract

Process Mining and especially *Conformance Checking* are about relating a descriptive model with event data (*Event Log*) to give predictions, alter processes or check compliance. The sequentially ordered *Event Log* data however has time stamp issues, which yields in only *partially ordered traces*. Can *machine learning* models give an accurate solution to resolving these partially ordered *traces*? So far machine learning methods have not been used to solve the problem of partial order resolution for *Event Log* data. That is why in this research project state-of-the art models for sequence to sequence prediction have been utilised, to predict possible resolutions to the partially ordered data.
Different methods to present the event log data to the algorithms are proposed. Based on that the performance of two machine learning models was assessed empirically on real-life and synthetic data sets. It was found that one model is clearly desirable over the other, as it's predictions are more accurate and certain.

Zusammenfassung

Process Mining und insbesondere *Conformance Checking* wollen ein deskriptives Modell und aufgenommene Daten (*Event Log*) in Beziehung setzen, um Vorhersagen zu treffen, Prozesse zu verändern oder Compliance sicherzustellen. Die sequentiell geordneten Daten des *Event Logs* haben jedoch Fehler in den Zeitstempeln, wodurch nur *teil-geordnete traces* entstehen. Können *Machine Learning* Modelle die richtige Auflösung dieser nur teiwleise geordneten *traces* präzise vorhersagen? Bisher wurden Methoden des maschinellen Lernens noch nicht angewendet, um das Problem einer Teilordnung auf *traces* eines *Event Logs* zu lösen.
Es werden mehrere Methoden vorgeschlagen, um die Event Log Daten für die Algorithmen zu kodieren. Darauf aufbauend wurden zwei Modelle aus dem Bereich der Sequenz zu Sequenz Vorhersagen angewendet und ihre Performanz hinsichtlich synthetischer sowie echter Datensätze empirisch untersucht. Eines der Modelle löst das Problem deutlich besser, da die Vorhersagen präziser und sicherer sind.

Contents

1 Introduction

Process Mining, as an area of research in computer science, emerged in the recent years from bringing together *process science* and *data science.* The former "*combines knowledge from information technology and knowledge from management sciences to improve and run operational processes*" and the latter is a general term for multiple computer science and math disciplines (e.g. statistics, algorithms, databases) which extract, prepare and explore data to provide predictions, automate decisions and create models or insightful visualizations. [1, p.10ff]
The data used in *process mining* are event logs, consisting of *traces*, such that each trace resembles the execution of one instance of the corresponding process (model). A key assumption on *event logs* is that there is a total order imposed on the activities of a *trace*, i.e. the order in which each of the activities have been executed is unambiguously apparent from the data; most often by timestamps associated to all events. [1] [7]
However the quality criteria of data in general are not always met [23] and for *process mining* in particular a study on general data quality issues to appear in event logs showed that at least 80% of the examined data sets had timestamp imperfections. [6]
A way to overcome this data imperfection would be leaving out the quantity of the imperfect data, i.e. filtering the data and to only further process the part of the data that meets all the assumed criteria, including a total order over the events.
When it comes to generating information from data filtered by this strategy, e.g. computing fitness or precision metrics, the accuracy of the results would be (1) biased, since important structures of the data might be completely unconsidered and (2) less generalisable since the distribution of structures in the data, e.g. *trace* frequency, might not resemble the true distribution.
Therefore, it is of high interest to develop and maintain accurate strategies to resolve these timestamp issues. Thus, in the endeavours of this research project the performance of two state-of-the-art machine learning models has been examined resolving partial orders in real-life as well as synthetic event log data.
In the following paragraphs a brief introduction into the two research areas, *Process Mining* and *Machine Learning*, will be given. Furthermore the questions that drove this research are set into context. Ultimately, the general approach and methods of the research are explained.

1.1 The Context of Process Mining and Machine Learning

Process Mining and *Machine Learning* are two areas of research in computer science. The former naturally inherits methods from the latter to solve process related tasks. An overview over both of the areas will be given in the following two paragraphs.

Process Mining

As described above, *process mining* is a rich area of research, which has many dependencies into others areas of computer sciences and mathematics. A general description of process mining is given by van der Aalst:

Process mining [...] sits between machine learning and data mining on the one hand and process modeling and analysis on the other hand. The idea of process mining is to discover, monitor and improve real processes [...] by extracting knowledge from event logs [...].[1, p.32ff]

A broad overview of the typical process mining idea is given by Figure 1 (inspired by [1, p.32, Fig. 2.5]), which emphasises the definition of process mining given above.
In a typical scenario, activities are being executed in the "*real world*" by various resources (people, machines, organizations), which are directly supported and/or controlled by software systems and stored as *event logs* by *process-aware information systems* (PAIS). These event logs are then being used to (typically) conduct three types of process mining. The first one being *discovery*, where using algorithms (e.g. the α-algorithm familiy) the goal is to leverage the internal structure of the event log data to create descriptive process models from the corresponding event data. Typical languages that are used to describe process models are *Petri Nets* or the *Business Process Model and Notation* (BPMN).

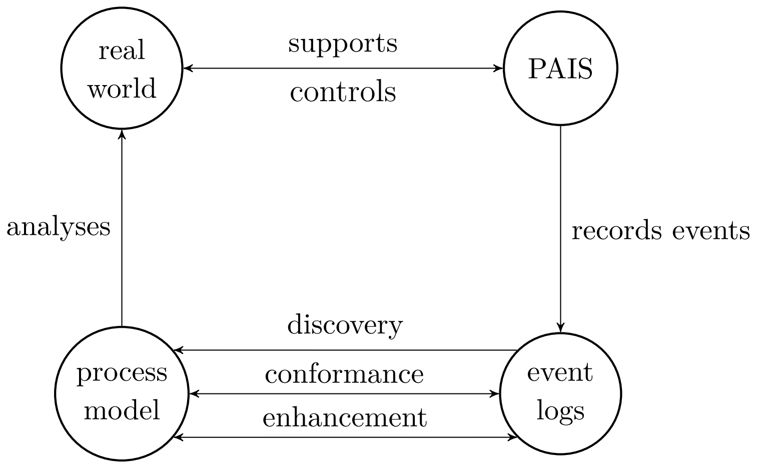


Figure 1: a general visualization of process mining

Furthermore, *process mining* thrives to *check the conformance* between some given event log and a formal process model, where the latter can be derived by *discovery* or created by hand of professionals.
Lastly, *Enhancement* takes an existing process model and the information contained in an event log for that exact process to alter the model. For instance such that it more accurately resembles the data in the event log or for repairing the model. [1]
The data as stored by the PAIS contains events of which each can be related to a distinct instance of process execution (case). All the events, in order of their execution, of one such case describe a *trace*, i.e. the sequential execution of an instance of the corresponding process (model). [1]
Recall that one of the key assumptions in *Process Mining* is, that there is a total order imposed on the events of a trace of a log. This criteria however is not always met, as timestamp related data quality issues are prominent in all domains [23] [30] and the following difficulties encountered are described in multiple research papers.

The quote "*garbage in - garbage out*" states that poor input data quality for data-to-information processes, such as process mining, implies poor quality information extracted. Data analysis applications such as stock market prediction or weather forecasts need precise timestamp information, whereas in process mining the information is needed to induce the total ordering over events within a case. This ordering is crucial to perform state-of-the-art process mining techniques for discovery, performance or conformance checking of the given data/ event logs.[13].
Van der Aa et. al [32] describe the three main reasons the lead to faulty time stamps in event log data as:

- ◇ *Lack of synchronization*: the problem that event log data is created from multiple information systems, that have to synchronize their clocks and the fact that the actual recording order of events and the order induced by the time stamps may not be consistent [18] [26]
- ◇ *Manual recording*: in some situations event logs are recorded manually by the corresponding staff, which may result into faulty time stamps, for instance when the data is recorded at the very end of the shift [22] [21]
- ◇ *Data sensing*: event logs derived from (sensed) data recorded by real-time locating systems makes event construction and time stamp assignment inaccurate due to the underlying probabilistic inference [31] [28] [24] [29]

Further reasons that lead to faulty timestamps and thus hinder a total ordering of traces have been described. Examples of timestamp quality issues related to their granularity, order anomaly or statistical anomaly, as described in [13], are: imprecise timestamps [1][23], event logs composed of events of different systems with distinct time recording strategies [33][16], timestamp formatting ("unanchored timestamp problem") [16][29], events being recorded after the process was finished [1], as well as manual entering events or timestamps [23][22] and problems occurring due to multiple, different time zones [16]. The inaccuracies have an effect on a broad range of process mining techniques, e.g. correct performance measure and process model discovery. [13] This means resolving the timestamp related issues that forbid to impose a total order on the event of trace will be of great usage in more accurately performing process mining tasks.

Machine Learning

In order to briefly introduce the context and topic of machine learning, a very general definition is given:

> *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E"* [25]

Some typical tasks T that are solved by machine learning methods are *regression, classification, transcription* and *machine translation.* Measures P that express the

performance of the model usually are the *error rate* or the *accuracy* respectively. The experience E can be divided into *supervised* and *unsupervised* learning.
When it comes to the latter method, an algorithm is presented some data, which internal structure may let the model learn useful properties. In *supervised learning* the algorithm is presented the data set as well as some label (target) provided for each example. [15]

1.2 Research Question

The problem of traces being only partially ordered and thus breaking one of the key assumptions in process mining, as well as rendering certain conformance checking tasks not applicable, has been worked on before. More detailed information about related research can be found in section 2.2. The problem of resolving partially ordered traces, in the sense of giving predictions for *possible resolutions* for traces with time stamp issues, has only initially been proposed by van der Aa et al. in 2020[32].
They created three stochastic models to resolve those partial orders and find the most probable ordering. They improved the average error by 60%, when applying those stochastic methods further to compute alignments and alignment based fitness.
This first approach however does not use machine learning models. In the light of *partial order resolution* for event logs being a sequence to sequence prediction task and the fact that multiple well performing machine learning models exist to generally solve the task of sequence to sequence prediction, e.g. *machine translation* [20] [9], it appears meaningful to make use of those models.
This report wants to empirically assess the performance P, as introduced in section 1.1, of two different *Recurrent Neural Network* structures on resolving partially ordered traces for three characteristically different event logs. Given these conditions and assumptions the two main research questions (RQ) that are covered in this research project are:

- ◇ *RQ1*: *How can Machine Learning models be utilised to solve sequence to sequence prediction tasks in the context of partial order resolution*?
- ◇ *RQ2*: *How do the chosen different models perform given some measure P?*

The former of the two research questions ($RQ1$) is necessary in order to find a reasonable and feasible way to translate the sequential event log data to a machine learning model, which perform algorithms on numerical presentations of the data.
The latter research question ($RQ2$) then leads the way to comparing different machine learning models on different solutions to the first research question, i.e. measure their predictive performance P. To make the models comparable to each other and obtain information in order to answer the research question 2 their respective performance P was quantified by three different performance measures, $P.1$ to $P.3$, introduced in Section 4.

1.3 Research Approach and Methods

In this research scenario the aim is to, given some supervised data set (experience E, event log), correctly classify uncertain event sets to their respective possible resolutions (task T) and ultimately doing this to obtain the maximum accuracy and certainty (performance P).
Thus to overcome the problem of partially ordered traces, models from machine learning have been leveraged to solve a sequence to sequence prediction task, which are (1) a *Recurrent Neural Network* using *LSTM cells* and (2) an *Encoder-Decoder* architecture.
In order to give an answer to the first research question, three different vector based representations of event log data are proposed in Section 3, which is necessary to perform the models algorithms' on the event log data.
As described in the previous paragraph in order to comprehensively answer the second research question for both of the machine learning models, three different measure of the performance P are used and introduced in detail in Section 4.
The general method to resolve the partial order in traces is to train a machine learning model on some supervised event log. Then the model can be presented an uncertain trace and produce the most likely sequence of events for that trace.
Given the aforementioned solutions to $RQ1$ and performance measures the models performances were then quantified and compared to each other for all data sets and representations of data, i.e. answers to $RQ1$.
The rest of this paper is structured as follows. First, the background (Section 2) is discussed giving insight on the technical and formal aspects on both *Process Mining* and *Machine Learning*, as well as covering related work. In section 3, the context of the research problem will be introduced in more detail by introducing a solution of how to translate the problem for the models to be used. The exact execution of the research in the context of the given problem as well as the results produced by the experiments will be presented and discussed in section 4. Ultimately the paper will be concluded in section 5.

2 Background

In the first part of this chapter the formal structures necessary to follow the research and to answer the proposed research questions are laid out in worthy detail. In the second part an overview of related work is given.

2.1 Preliminaries

The formal preliminaries that are necessary to be explained for this report can be divided into two categories. The concepts related to process mining define the data structures, the assumptions imposed by them and the context of how resolving a partial order is defined. Following that in the machine learning context, the models and their respective characteristics are explained.

Process Mining Related Preliminaries

The traditional concepts and notations in this paragraph are taken from the Process Mining book by Wil van der Aalst [1] and the Conformance Checking book by Matthias Weidlich et al. [7].
Informally an *Event Log* is a data structure holding sequentially executed *activities* of a corresponding *process (model)* as *events*.

Definition 1. (Event) Let $\mathcal{E}$ be the universe of events. An event $e \in \mathcal{E}$ describes that a most single unit of a process has been executed. Events hold certain attributes, from the universe of attributes $\mathcal{N}$. So a function $\#_n(e)$ for an $n \in \mathcal{N}$ is defined, which describes the value of attribute n for event e.

Each event is annotated with at least three attributes: (1) for clarifying which process instance the particular event belongs to (case id), (2) which step of the according process was executed (activity) and (3) at what specific time this event was executed (timestamp). For this research project these three attributes are sufficient, i.e. $\mathcal{N} = \{Activity, Case, Time\}$. Usually events in real-life logs contain more attributes, e.g. the associated resource or the current status of the process.

Definition 2. (Activity) Let $\mathcal{A}$ be the universe of activities. An event $e \in \mathcal{E}$ is mapped to a distinct activity, i.e. $\#_{Activity}(e) \in \mathcal{A}$, which describes the process step that was executed for that event.

The universe of activities $\mathcal{A}$ holds a finite number of possible actions that can be executed for a certain process. Note that all events in a log are uniquely identifiable, whereas multiple events can relate to the same activity execution. The definition of cases is needed for defining *traces* later on.

Definition 3. (Case) Let $\mathcal{C}$ be the universe of cases, then $\#_{Case}(e) \in \mathcal{C}$ is used for relating events to the same execution instance of a process.

For completeness the mapping of the last necessary attribute is defined such that $\#_{Time}(e)$ for an $e \in \mathcal{E}$ returns the exact time at which the happening of that event was recorded.
The events that have been recorded during the execution of activities within the context of the same case are considered to belong to the same *trace*. The traditional definition of traces in process mining is that a trace tr is a finite sequence of events $e_1, e_2, ..., e_n$, which is obtained by ordering each event by its timestamps, i.e. for all $1 \leq i < j \leq n$ it holds that $\#_{Time}(e_i) < \#_{Time}(e_j)$. All events of a trace are related to the same case, i.e. for all $e, e' \in tr, \#_{Case}(e) = \#_{Case}(e')$.
A *trace tr* can also be represented as a sequence of activities $\sigma_{tr} = \langle a_1, a_2, ..., a_{|tr|} \rangle \in \mathcal{A}^*$, where $a_i = \#_{Activity}(e_i)$ for each e_i in the sequence $e_1, e_2, ..., e_n$.
Since events with equal timestamps need to be considered, such ordering would not be achievable. Thus traces are defined as sets of events. For formally introducing the concept of partially ordered traces, events are gathered in event sets E_i, such that all events that share the same case id and timestamp are considered to belong to the same event set.

Thus one can order the traces of a log by their respective event sets.
The initial work of van der Aa et al. has introduced the necessary advanced concepts to cope with order uncertainty in traces and event logs, which are introduced in the rest of this paragraph and likewise guided by the initial work [32].

Definition 4. (Trace) A sequence of disjoint sets of events $E_1, E_2, ..., E_n \subseteq \mathcal{E}$ is a trace $\sigma = \langle E_1, ..., E_n \rangle$, if for all $e, e' \in E_\sigma$, $\#_{Case}(e) = \#_{Case}(e')$, where $E_\sigma = \bigcup_{1 \leq i \leq n} E_i$ is the set of all events of σ and $\forall E_i \in \sigma \forall e, e' \in E_i \#_{Time}(e) = \#_{Time}(e')$ and $\forall E_j, E_k \in \sigma \forall e_{jl} \in E_j$ and $e_{km} \in E_k \#_{Time}(e_{jl}) < \#_{Time}(e_{km}), 1 \leq j \leq k \leq n, l \in |E_j|, m \in |E_k|$

Similarly to the definition of traces an event log in the event set view is slightly deviant from the traditional definition, as for example used in [1] or [7]:

Definition 5. (Event Log) An Event Log $\mathcal{L}$ is a tuple $(\Sigma, \#_{Activity})$, where Σ denotes a set of traces and $\#_{Activity} : \bigcup_{\sigma \in \Sigma} E_\sigma \to \mathcal{A}$ assigns activities to all events.

For visualisation purposes, also for the encodings introduced in section 3, a running example is introduced. For the running example let the activity alphabet $\mathcal{A}_{re} = \{A, B, C\}$. Table 1 shows the event data, which can be referred to as the example log $\mathcal{L}_{re}$. Note, however, that this visualisation does not yet specify which view on traces is taken, i.e. one can interpret the traces in the traditional and in the *event set* way.
The log consists of three traces belonging to the cases with ID 1,2 and 3. As a shortcut notation, similar to the view on traces of the traditional definition, the activity mapping for a trace is implicitly assumed, such that for example the second trace according to Table 1 is written as $\sigma_2 = \langle \{A\}, \{B, C\}, \{D\} \rangle$ instead of $\sigma_2 = \langle \{e_4\}, \{e_6, e_9\}, \{e_{10}\} \rangle$.

Table 1: Running example event data

Event ID	**Case ID**	**Activity**	**Timestamp**
e_1	1	A	04:00
e_2	1	B	04:30
e_3	1	C	05:30
e_4	2	A	06:00
e_5	3	A	06:10
e_6	2	B	06:15
e_9	2	C	06:15
e_7	3	B	06:30
e_8	3	C	06:30
e_{11}	3	B	06:30
e_{10}	2	A	06:45

For any Event Set $E = \{e_1, ..., e_n\} \in \wp(\mathcal{E})$there exist $n!$ potential orderings of that Event Set, i.e. orderings in which way those events could have been executed in real life. One such sequential ordering of an *event set* is called a *possible resolution* and formally captured in the following definition.

Definition 6. (Possible Resolution of Event Set) For an event set $E \subseteq \mathcal{E}$ the possible resolutions of E are defined as all ordered sequences of the elements in E with $\phi(E) = \{\langle e_1, ..., e_{|E|}\rangle | \forall 1 \leq i, j \leq |E| : e_i, e_j \in E \wedge e_i = e_j \implies i = j\}$

According to this definition the possible resolution of a trace consisting of consisting of possible resolutions of event sets is given below.

Definition 7. (Possible Resolution of Trace) For a trace $\sigma = \langle E_1, ..., E_n\rangle$ the possible resolutions of σ are defined as all combinations of total orders over the event sets E_i in σ with $\Phi(\sigma) = \{\langle e_1^1, ..., e_1^{m_1}, ..., e_n^1, ..., e_n^{m_n}\rangle | \forall 1 \leq i \leq n : \langle e_i^1, ..., e_i^{m_i}\rangle \in \phi(E_i)\}$

For comparison, the traditional view and the event set view on the traces of the running example log are shown in Table 2. Here the ordering of the traditional traces is, as discussed later, done by the gold standard, i.e. the order in the log, as in Table 1.

Table 2: Running example traces

Trace	**Traditional View**	**Uncertain View**
σ_1	$\langle A, B, C\rangle$	$\langle \{A\}, \{B\}, \{C\}\rangle$
σ_2	$\langle A, B, C, A\rangle$	$\langle \{A\}, \{B, C\}, \{A\}\rangle$
σ_3	$\langle A, B, C, B\rangle$	$\langle \{A\}, \{B, C, B\}\rangle$

For the second trace of the running example log, $\sigma_2 = \langle \{A\}, \{B, C\}, \{A\}\rangle$, it follows, that the two possible resolutions $\Phi_1, \Phi_2 \in \Phi(\sigma_2)$ are $\Phi_1 = \langle A, B, C, D\rangle$ and $\Phi_2 = \langle A, C, B, D\rangle$. In order to categorise traces and logs according to their certainty, the notion of certain and uncertain traces and logs respectively is defined below.

Definition 8. (Certain, Uncertain Trace) A trace $\sigma = \langle E_1, ..., E_n\rangle$ is called certain if $\forall E_i \in \sigma : |E_i| = 1, i \in \{1, ..., n\}$. Otherwise σ is called uncertain.

Definition 9. (Certain, Uncertain Event Log) An Event Log L is called uncertain if there exists an uncertain trace $\sigma \in L$, otherwise L is called certain.

For example the log shown in Table 1 is considered uncertain, since the second and third trace of that log are uncertain as well. The first trace, however, is considered certain.
In the light of the given definitions the natural number K is used for describing the size of the largest *event set(s)* E found in an event log $\mathcal{L}$. So for example regarding the running example log $\mathcal{L}_{re}$ it follows that $K = 3$. Also for a given trace σ, the possible resolution $\Phi^\star \in \Phi(\sigma)$ denotes the true order of execution of the events of the given trace. Similarly $\phi^\star(E)$ denotes the true ordering of the events for an event set E.

Machine Learning Related Preliminaries

In this paragraph an overview over the machine learning models used is given, by first introducing *Recurrent Neural Networks* (RNN) and *Long-Short Term Memory* (LSTM)

cells followed by combining these structures to present the *Sequence-to-Sequence* (Seq2Seq) architecture. For this paragraph the concepts and notations are taken from [15] and [14]. The figures are inspired by [15], [30] and [19].
In this research project the artificial neural networks are supposed to solve a classification task, i.e. the model learns a mapping $f(\mathbf{x};\theta) = \mathbf{y}$. Regarding *partial order resolution* the inputs $\mathbf{x}$ are event sets, that are mapped to *possible resolutions* $\mathbf{y}$.
RNNs are a sophisticated type of neural networks, which have recurrent connections between hidden units (s. Figure 2), and therefore allow taking temporal dependencies of sequential data into account. Among other configurations, they can be modeled in a way that a prediction for each time step is produced and thus perform good on sequence to sequence prediction tasks. [15]

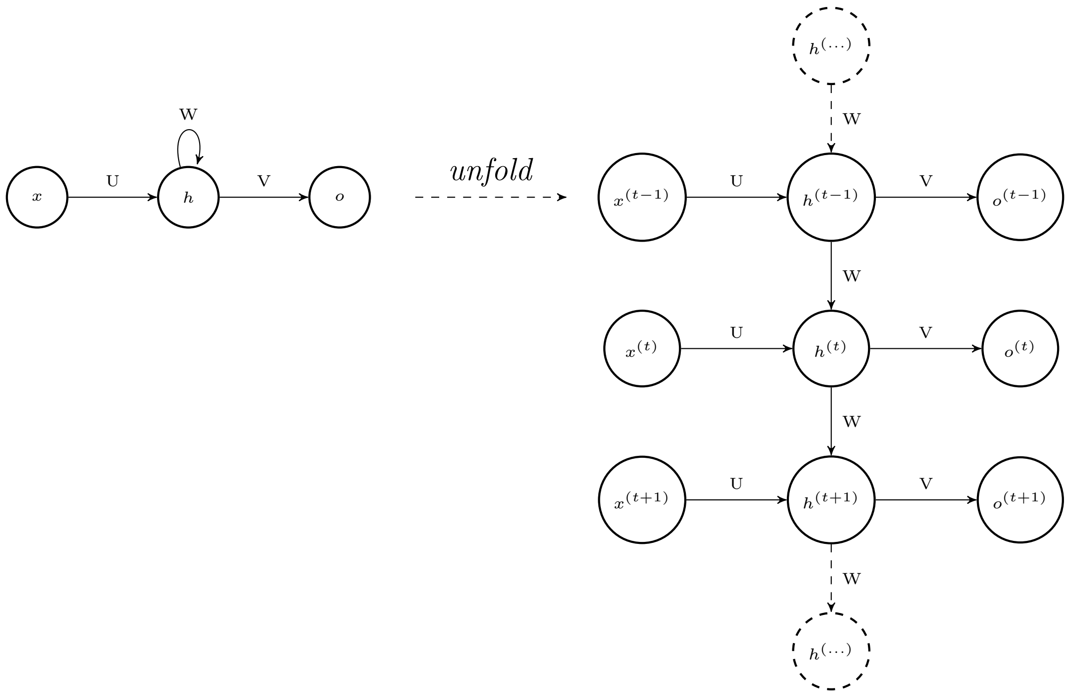
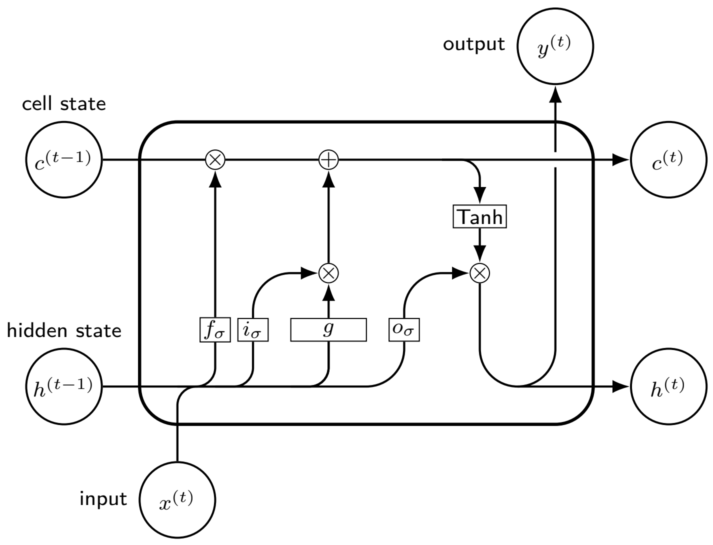


Figure 2: a simple recurrent neural network, compressed and unfolded

Figure 2 shows the two visualisations of a general RNN. The compressed representation can be unfolded, giving more insight into the mathematical dependencies of the model. At each time step $x^{(t)}$ of a given sequence $x^{(1)}$, ..., $x^{(t-1)}$, $x^{(t)}$, $x^{(t+1)}$, ..., $x^{(n)}$, the weight matrices U and V are used for computing the next layers' values, followed by some implicitly assumed activation function. This is done for all layers, starting at the input layer. The output layer is then computed by using the matrix V and an output function. The hidden state h computed while sequences are processed acts as the memory of the model, "remembering" the structure of input sequences.
Well-known candidates for the activation functions are the hyperbolic tangent or the sigmoid function. [30] In modern models the rectified linear unit (ReLU) is advised as well [15] and in classification tasks for the last layer the softmax function is usually used [14].

The loss for a given sequence of data points $\mathbf{x}$ and target values $\mathbf{y}$ is then defined to be the sum of the losses for all time steps and optimised by the back propagation through time algorithm (BPTT). [15]
With classical RNNs there still is the problem of the *vanishing gradient*, which states that over time, as sequences are processed through the model, the earlier inputs are becoming less influential to the learned parameters. Graphically speaking, those models have a short term memory, remembering only the latest inputs best. To resolve this issue, in [17] a dedicated RNN cell, the LSTM cell was introduced.
This LSTM cell replaces the hidden layer units, shown in Figure 2, so that next to the layers hidden states $\mathbf{h}$, also a new cell state $\mathbf{c}$ is kept and updated throughout the processing of sequences. Additionally, more elaborated computations resolve from that, which are visualised in Figure 3[1].

Figure 3: graphical representation of a LSTM cell



In a typical LSTM cell the weighted combination of the previous hidden state and the current input is directed by three gates (the forget gate f_σ, the input gate i_σ and the output gate o_σ), which is why RNNs using LSTM cells are also called *gated RNNs*. The σ in the index of the gates' names indicates that they have a sigmoid non-linearity, whereas g can have any squashing non-linearity. [15]
The main step is the one that gives g. It combines the current input $\mathbf{x}^{(t)}$ and the previous (short-term) hidden state $h^{(t-1)}$ (In a classical RNN this directly contributes to $h^{(t)}$ and $y^{(t)}$.)
The three gates control the information flow and since they all output values in the range of 0 and 1, which are distributed by element-wise multiplications, the gates can be considered closed (a value is 0) to open (a value is 1). Precisely, the forget gate f_σ determines, which parts of the long-term state should be forgotten. The input gate i_σ controls, which parts of the information of g should be added to the long term state.

[1]the initial code for this figure was taken from here

And the output gate o_σ is used for computing the information that should be the output to $y^{(t)}$ and $h^{(t)}$, in combination with a non-linear squash of the current cell state. [14]
Note that in a classical RNN model, as shown in Figure 2, the output $\mathbf{y}^{(t)}$ for a given element of a sequence to predict, is not the same as the hidden state $\mathbf{h}^{(t)}$, whereas when using LSTM cells $\mathbf{y}^{(t)} = \mathbf{h}^{(t)}$. [14]
There is one last model that needs to be introduced. It is the Sequence-to-Sequence model (Seq2Seq), also known as an Encoder-Decoder model. This model basically consists of two separate RNN models using LSTM cells.
(1) The *Encoder RNN*, gets a given sequence as input and processes it, producing the hidden state $h^{(t)}$ and the cell state $c^{(t)}$. In this model those states emitted by the *Encoder RNN* are regarded as the context C, a function of the states, that represents the structures in the input feature space.
(2) There is the *Decoder RNN*, which is provided the context C as input as well as the start of sequence symbol (SOS). The Decoder then predicts the symbol $t+1$, given the symbol t, e.g. predicting the first symbol when presented the context C and the SOS. Then the prediction $t+1$ and the updated decoder states are re-injected into the *Decoder RNN* to produce the next symbol (and updated states), until the end of sequence symbol (EOS) is predicted or some maximum sequence length is reached.
The Encoder and Decoder RNN are trained jointly, to maximise the probability of the output sequence, given the input sequence for all pairs in the training set. [15]
In the field of deep learning, for injecting non numerical data, such as text or event data, to the models inherent algorithms, a tractable representation of that data needs to be decided on. This is usually described as an *encoding* of the data. Loosely speaking an encoding of a data point x is a mapping to a sparse vector such that the original data structures are captured and the data can be reconstructed by the inverse of the mapping.

2.2 Related Work

In this section related work regarding data quality in general and for process mining especially, machine learning methods applied for process mining tasks and research that involved partially ordered traces or alignments is summarised.

Data Quality

In the Process Mining Manifesto [33], the importance of timestamps for process mining is summarised. They can be for instance used in the *enhancement* of a process model, so that it can be extended to make bottlenecks, service levels, throughput times, estimated waiting time for activities and frequencies visible.
In [33] van der Aalst et. al. introduce a quality rating, where the highest rank (*five star maturity level*) is argued to be the desired data quality and only obtained by *totally ordered* logs.
The research done by Accorsi et al [2] encourages the resolution of partially ordered traces to totally ordered ones. This case study shows, how process mining and conformance checking techniques can be used to verify security requirements for (business) processes and thus reveal violations of time constraints.

Adriansyah et al. show in [3], that a total order is necessary in order to perform state-of-the-art conformance checking on event logs.
In [13] Dixit et al. present a semi-automated method to repair timestamp imperfection in event logs with an alignment based approach.

Machine Learning Methods in Process Mining

In [30] Niek Tax et al. use RNNs with LSTM cells to answer predictive questions not yet visited in the field of process mining. However, they do not take into account the order certainty of traces and thus, while exploring an answer for the next activity, omit the information of certain parts of a trace already being ordered and ultimately do not resolve partially ordered traces in general.
Embeddings have been introduced to process mining successfully by de Koninck et al. in [10] with vector representations learned for activities (act2vec) and traces (trace2vec) among others, similar to *Word2Vec*. This work has been applied to a new conformance checking approach by Peeperkorn et al. with promising initial results [27].

Partially Ordered Event Data

Process Mining and Machine Learning are extensive areas of research. However, so far there has been little research done, addressing the problem of partially ordered event logs [22].
M. de Leoni et al. presented a technique to abstract the problem of aligning partially ordered traces into a PDDL-encoded planning problem. This approach will either report, that there exists no solution, i.e. optimal alignment, for an explicit trace and petri net. Otherwise in finite time an optimal alignment is found, whether or not the trace is sequential, i.e. totally ordered or containing concurrent events, i.e. partially ordered [12].
In [21], van der Aalst et al. took another approach in using partially ordered traces to compute partially ordered alignments with the aim to provide a model that can express concurrently running events. They investigated the usefulness of those partially-ordered alignments with case studies relying on real-world data from a Dutch hospital [22].
Another mention of partially ordered logs appears in [5], where Beschastnikh et al. propose three algorithms to extract temporal invariants from partially ordered logs to capture concurrency and apply process mining techniques to concurrently running systems.
None of the work above, however, solves the problem of *partially order traces* directly, as of finding the most probable ordering of a trace. This has initially been worked on by van der Aa et al. [32]
Van der Aa et al. have sought three stochastic models for resolving partially ordered traces in giving a probability distribution over all possible resolutions for an event set. From there they presented an algorithm to efficiently compute the conformance of partially ordered traces with a given process model.
This research project tries to find an approach to directly resolve partial orders that for instance can be applied to *conformance checking*, hopefully leading to more accurate results than [32].

3 Translating the Problem

In this section the first research question ($RQ1$) is addressed: *how can the problem of partial order resolution be solved using machine learning models?* The models that where examined in this research project are (1) a plain RNN using LSTM cells and (2) the Encoder-Decoder model (Seq2Seq2). Recall that it is necessary to find a useful representation of the sequential event log data, as for example depicted in Table 1, to be processed by the given machine learning models.
Therefore three different encodings of traces are proposed, which each cover different information on the event log data. In the following subsections, the term *input* refers to an encoded data point, i.e. an encoding vector for a given trace. Similar *output* denotes an encoding vector produced by a model in the prediction process and the encoding used to encode the target values for prediction comparison.

3.1 Activity Space Encoding

The idea for the *Activity Space Encoding* (Encoding 1) is to compress the log into one-hot vectors that are of the size of the activity universe $\mathcal{A}$, i.e. each position in the vector corresponds to an $a \in \mathcal{A}$. An encoding vector v for a given event set E then contains a 1 at every index that corresponds to the activity executed for each event $e \in E$. Traces shorter than the maximum trace length are padded with a zero vector.
To visualize the idea, the according encodings of the running example log $\mathcal{L}_{re}$ is given below.

Input vectors for the example log and the LSTM model:

$$\sigma_1 : \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{matrix} \{A\} \\ \{B\} \\ \{C\} \\ \text{pad} \end{matrix} \qquad \sigma_2 : \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{matrix} \{A\} \\ \{B,C\} \\ \{B,C\} \\ \{A\} \end{matrix} \qquad \sigma_3 : \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{matrix} \{A\} \\ \{B,C,B\} \\ \{B,C,B\} \\ \{B,C,B\} \end{matrix}$$

Note that one trace is visualized in matrix notation here. So one row of a matrix corresponds to the encoding of an event set or a possible resolution respectively. Also the input encodings slightly differ for the LSTM and the Seq2Seq model, due to the nature of the sequence generating process of the given models.
The *Activity Space Encoding* for the LSTM model will include repeated uncertain event sets. This means, that in order to make the input sequence match the length of the output sequence, for every uncertain event set in the trace the corresponding encoding vector is also added repeatedly, i.e. as often as indicated by the size of the uncertain event set.
For the Seq2Seq model the approach is similar. The one difference is that the one-hot encoded vectors of event sets of size ≥ 2 are not repeated as the input and output sequence length can vary for this model. Thus the encoding is slightly different, as can

be seen below, exemplary for the running example log.

Input vectors for example log and the Seq2Seq model:

$$\sigma_1 : \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\\0&0&0\end{bmatrix}\begin{matrix}\{A\}\\\{B\}\\\{C\}\\\text{pad}\end{matrix} \qquad \sigma_2 : \begin{bmatrix}1&0&0\\0&1&1\\1&0&0\\0&0&0\end{bmatrix}\begin{matrix}\{A\}\\\{B,C\}\\\{A\}\\\text{pad}\end{matrix} \qquad \sigma_3 : \begin{bmatrix}1&0&0\\0&1&1\\0&0&0\\0&0&0\end{bmatrix}\begin{matrix}\{A\}\\\{B,C,B\}\\\text{pad}\\\text{pad}\end{matrix}$$

The encoding vectors of the ground truth, i.e. the true possible resolution ($\Phi^\star(\sigma) \in \Phi(\sigma)$) of a trace as well as for the output vectors, are the same for the LSTM and the Seq2Seq model. They are defined as a basic one-hot vector over $\mathcal{A}$.

Output vectors for the example log:

$$\sigma_1 : \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\\0&0&0\end{bmatrix}\begin{matrix}A\\B\\C\\\text{pad}\end{matrix} \qquad \sigma_2 : \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\\1&0&0\end{bmatrix}\begin{matrix}A\\B\\C\\A\end{matrix} \qquad \sigma_3 : \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\\0&1&0\end{bmatrix}\begin{matrix}A\\B\\C\\B\end{matrix}$$

The advantage of this encoding is the small encoding vector size. Each event set is encoded in a way such that the length of the vector is equal to the size of the activity universe $\mathcal{A}$, so the length for both *input* and *output* vectors of the running example log $\mathcal{L}_{re}$ is 3.

3.2 Event Set Encoding

In order to encode more naturally the structure prominent in an event log, which contain traces as sequence of event sets, as described in definition 4, this and the following encoding take the space of event sets and possible resolutions as feature space for input and output respectively. The idea is to map each event set of a trace to a possible resolution for that event set.
Clearly the size of these spaces is not implicitly limited, since arbitrary sized event sets and according possible resolutions could potentially appear in logs. This size is limited by the number K of a given log, which is the size of the largest event set in the log.
For a given K and an activity set $\mathcal{A}$ the number of possible event sets to appear and thus the length of input vector size can be computed according to equation (1) below.

$$\sum_{k=1}^{K} \frac{(|\mathcal{A}|+k-1)!}{(k! \cdot (|\mathcal{A}|-1)!)} \quad (1) \qquad \sum_{k=1}^{K} |\mathcal{A}|^k \quad (2)$$

On the other hand the output vector size, i.e. all possible sequences from length 1 to K can be simply computed as given by equation (2). Thus the output vector size naturally is an upper bound to the input vector size.
An example of the *Event Set Encoding* (Encoding 2) for the running example log $\mathcal{L}_{re}$ is

given below.

Input vectors for the example log and both the LSTM and Seq2Seq model:

$$\sigma_1 : \begin{bmatrix} 1 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{ll} \{A\} & \\ \{B\} & \\ \{C\} & \\ \text{pad} & \end{array}$$

$$\sigma_2 : \begin{bmatrix} 1 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & ... & 1 & ... & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{ll} \{A\} & \\ \{B, C\} & \text{1 at index 7} \\ \{A\} & \\ \text{pad} & \end{array}$$

$$\sigma_3 : \begin{bmatrix} 1 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & ... & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{ll} \{A\} & \\ \{B, C, B\} & \\ \text{pad} & \\ \text{pad} & \end{array}$$

Consequently to the formulas above for the running example log the input vectors are of size 19 (all possible event sets up to length K) and the output vectors are of size 39 (all possible resolutions up to length K), with $|\mathcal{A}_{re}| = 3$ and $K_{re} = 3$.
Likewise the output vectors according to Encoding 2 are given by a one-hot encoding over all the possible resolutions for event sets of size up to K. Note that in this case the *input* and *output* encodings for both of the models are the same.

Output vectors for the example log for both the LSTM and Seq2Seq model:

$$\sigma_1 : \begin{bmatrix} 1 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{ll} A & \\ B & \\ C & \\ \text{pad} & \end{array}$$

$$\sigma_2 : \begin{bmatrix} 1 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & ... & 1 & ... & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{ll} A & \\ B, C & \text{1 at index 8} \\ A & \\ \text{pad} & \end{array}$$

$$\sigma_3 : \begin{bmatrix} 1 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & ... & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{ll} A & \\ B, C, B & \text{1 at index 28} \\ \text{pad} & \\ \text{pad} & \end{array}$$

This representation of the event log data ensures that the whole log, as viewed as traces of event sets, can be encoded.

3.3 Extended Event Set Encoding

The third encoding is a more sophisticated version of the *Event Set Encoding.* Here for a given event set, the input vector not only is assigned a 1 at the index corresponding to the event set. Additionally, it will carry a 1 at every index corresponding to a subset of the event set that is to be encoded.
An example of that encoding is given below, and thus for instance the encoding of the second event set in the second trace σ_2 of $\mathcal{L}_{re}$ carries a 1 not only at the index corresponding to the event set $\{B, C\}$ but also at the index corresponding to all subsets, i.e. $\{B\}$ and $\{C\}$.

Input vectors for both the LSTM and Seq2Seq model and Encoding 3:

$$\sigma_1 : \begin{bmatrix} 1 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{l} \{A\} \\ \{B\} \\ \{C\} \\ \text{pad} \end{array}$$

$$\sigma_2 : \begin{bmatrix} 1 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & ... & 1 & ... & 0 & 0 \\ 1 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{ll} \{A\} & \\ \{B, C\} & \text{1 at index 1,2,7} \\ \{A\} & \\ \text{pad} & \end{array}$$

$$\sigma_3 : \begin{bmatrix} 1 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & ... & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & ... & 0 & 0 & 0 & 0 \end{bmatrix} \begin{array}{ll} \{A\} & \\ \{B, C, B\} & \text{1 at index 1,2,6,7,16} \\ \text{pad} & \\ \text{pad} & \end{array}$$

The output vector for the *Extended Event Set Encoding* is encoded as a one-hot over all the possible resolutions up to length K, and therefore the same as for the *Event Set Encoding.* Also the size of the respective *input* and *output* vectors can as well be computed according to the formulas (1) and (2) of the previous section and will amount to the same quantities as for the *Event Set Encoding.*
To sum up, three different solutions are proposed for the first research question. With that it is possible to use the particular machine learning models for *partial order resolution.* The evaluation of exactly that, using the solutions proposed here, is presented in the next section.

4 Evaluation

This chapter will summarize the objective, that means stating what is to be evaluated and quantified in regard to some performance measures, in order to deduce answers to $RQ2$. Then the setup and execution of the experiments will be illuminated, giving further insight on the experimental data and the methods. Ultimately the results will be presented with a following discussion, that sets the results and the research questions in context, interpreting the outcome of the research.

4.1 Objective

The objective of this research project is to empirically answer research question 2: "*How do the chosen different models perform given some measure P?*". Measuring this performance P is done on multiple levels of abstractions and results in fanning out RQ2 into multiple sub-research questions:

- ◇ *RQ2.1*: *How accurate are a models predictions?*
- ◇ *RQ2.2*: *To what extent is a model capable of predicting possible resolutions only?*

The first two of those more fine granular research questions both assess some sort of accuracy of the predictions made by a given model.
RQ2.1 will give insight on the plain accuracy given by a model and basically asks if one of the models is superior of the other, when it comes to plain prediction accuracy.
RQ2.2 relates to the fact that by construction the models are capable of predicting non possible resolutions for a given event set, i.e predicting a sequence, that is not in the set of possible resolutions for a given event set. For example for the event set $\{A, B\}$ the prediction BC is not a possible resolution. The interest of research here is, how often a given model needs to be advised to re-consider it's prediction, as it did not predict a possible resolution.
Research question 2.3 results from examining the predictions probability distributions and is stated below:

- ◇ *RQ2.3*: *Which deductions can be made from the prediction probabilities in terms of learned behavior by the model and uncertainty of the predictions?*

By this the goal is to obtain data that gives insights into probability distributions of the predictions scores for certain types of event sets.
The fact, that there are multiple proposed solutions to *RQ1*, results in relating the three questions above with the encodings introduced in Section 3:

- ◇ *RQ2.4*: *How are the results above influenced by different solutions to RQ1?*

When comparing different models regarding the outcomes of research questions 2.1 to 2.4, a pooled answer to $RQ2$ can be found.

4.2 Experimental Data, Setup and Execution

In this section insight on the studied data is included, since event log data is highly variable. Thus a good overview on the general implications made by the structure of the data is necessary. After that in order to quantify results, giving answers to the research questions *RQ2.1 - RQ2.3* three performance measures (*P.1 - P.3*) are introduced. Then a short explanation of the general research pipeline and technical details is given.

Data Understanding

Event Logs are gathered in various fields of applications, which range from the medical to the transport sector up to typical business applications. In this research project two real life logs were examined. The BPIC-14 log [34], provided by Rabobank Group ITC, contains events on their service desk process. The second one is the Road Traffic Fine Management Process (traffic fines) log [11], which holds data stored by a PAIS managing road traffic fines. Additionally a third, synthetically generated, log was examined, which originates from the same pipeline that was used in [32].

As the logs come from different applications or are synthetic, the characteristics of the data in these event logs are highly diverse. The most important characteristics of the event logs examined in this research project are summarised in Table 3. It shows, that the event logs vary significantly in most of their traits.

The size of the activity universe $\mathcal{A}$ varies from nine for the BPIC-14 log, over eleven for the traffic fines log to seventeen for the artificial log. Real-life Event Logs however, can have a much bigger activity spaces, e.g. the log of the BPIC-15 [35], which contains almost 400 possible activities to be executed. This is an important trait as this defines the size of the encoding vectors and thus space complexity and implicitly practical applicability.

The number of traces is usually in the multiple of thousands for real-life logs. Accordingly

Table 3: Characteristics of the examined event log data

Characteristic	**BPIC-14 [34]**	**Traffic fines [11]**	**Artificial [32]**
$\lvert\mathcal{A}\rvert$	9	11	17
# Traces	41353	150370	1000
# Events	369485	561470	8502
# Variants	14948	231	505
# Variants (unc.sets.)	24	25	102
Trace length (avg.)	8.9	3.7	8.5
Trace Uncertainty	93%	6%	49%
Size of event set (avg.)	2.6	2.0	2.1
Largest event set (K)	4	3	4

the respective real life logs contain about 40000 traces (BPIC-14) and 150000 respectively (traffic fines). The artificial log however is rather small and consists of only 1000 traces. Inherently depending on the number of traces the number of events vary from about 370000 (BPIC14), to 561000 (traffic fines) and 8502 for the artificial log.

The amount of uncertainty, i.e. the number of traces that do at least contain one event set of size ≥ 1, also span a broad spectrum. The traffic fines log contains 94% certain traces, whereas the BPIC-14 log only has a level of certainty of about 7%. The artificial log has an uncertain event set in about half of the traces.

It is remarkable, that for the real life logs from possible 714 (363) event sets, that could appear in the BPIC-14 (traffic fines) log, only 33 (36) are present in the recorded data, i.e. only about 5% (10%). For the artificial log 2%, 119 of 5984, of the possible event

sets were found in the created data.
Interestingly the size of the uncertain event sets, present in the three logs, are less deviant from each other. The biggest uncertain event sets are found in the artificial and BPIC-14 logs and are of size four. The largest event set for the traffic fines log is of size 3. Related to that, the average size of event sets in the examined logs behave similarly. The values are as low as two and never exceed three (2.6 for the BPIC14 log, 2.0 for the traffic fines log and 2.1 for the artificial log).

Performance Measures for Evaluation

In order to asses the performance of the models on different data sets on different levels, three measures of performance were used:

- *P1*: general prediciton accuracy
- *P2*: re-prediction count
- *P3*: evaluation of prediction distributions

The first measure, $P.1$, purely describes the accuracy of a model. For a given set of traces, previously unseen to the model and commonly referred to as the *test set*, the correctness of the models predictions is captured.
The models generally can predict any sequence of events (up to length K) for a given input event set E_i. For instance the sequence $\langle B, C, D\rangle$ could be considered most probable for the event set $\{A, B, C\}$. This is not a desired solution, as clearly the predicted sequence is not a possible resolution of the event set, i.e. $\langle B, C, D\rangle \notin \phi(\{A, B, C\})$. This domain knowledge is leveraged and a model will be asked for the next most probable prediction p, until p is a possible resolution for the given event set E_i, i.e. $p \in \phi(E_i)$.
A prediction is then considered correct, if for each event set in the trace the true possible resolution $\phi^\star$ was predicted. Then the accuracy measure *P.1* is given by the quotient of correct predictions and the size of the training set.
$P.2$ draws onto that by measuring the amount of first predictions, that were not a possible resolution. If the first prediction p needs to be reconsidered, this is counted, at most once per event set, and the *re-prediction count* is then given by this number. The *re-prediction percentage* is then given by the quotient of the *re-prediction count* and the number of event sets in the training set.
The last measure, $P.3$, is less straight-forward. It generally breaks down into comparing the probability distributions imposed by the output vectors for a prediction p. Probability distributions are then visualized in boxplots and will be compared by usual stochastic measures, such as mean, variance and inter quartile range (IQR).
For this, the prediction probabilities for two scenarios are compared. (1) After the prediction for an event set is done, the probability associated to this predictions is gathered as well as (2) the probability that is associated to the ground truth by the current prediction vector. Thereby, multiple distributions of probabilities can be compared.

Execution and Technical Details

With performance measures clearly described the research in general was conducted as follows and consist of three steps: (1) data preprocessing, (2) training the model and (3) evaluating the model.

(1) In the pre-processing of the data, the time zone information in the timestamps for all events was equalised, since some event logs contain multiple time zones across the whole log. Furthermore, only the relevant information of the logs was extracted (omitting all non-relevant attributes) and they were transformed into the necessary and descriptive data structures.

Also a log was split into a *certain log*, a sub-log containing only the *certain traces* and an *uncertain log*, holding only the *uncertain traces* respectively. The data used for training and testing the machine learning model was then the *uncertain log.* The procedure also includes the encoding of the data according to one of the proposed methods and splitting the data randomly into a training and test set by the ratio of 80 / 20. Note that for the ground truth, as in [32], a gold standard value was used, determining the correct order as the order of the trace in the log. Also, for the BPIC-14 log the size of the data set was reduced to be comparable to the traffic fines log.

(2) With the pre-processing done, the chosen machine learning model was trained on the training set. Exact specification to hyperparameters such as the number of epochs, optimisers and precise network architecture can be found in the git repository [2].

(3) Finally the experiments were evaluated according to the performance measures *P.1* to *P.3* using the test set. Note that for $Encoding1$ for event sets of size ≥ 1 multiple singular activities are predicted. Thus, for an event set E to find the most probable possible resolution the length l of the current event set E was memorised. Then the for the next l predictions for all possible combinations the product of the probabilities were computed, which was then used to find the most probable possible resolution. For Encoding 2 and 3 though the prediction probabilities for the most probable possible resolution can directly be extracted.

To implement the experiments *python* was used alongside the state-of-the-art *process mining* library *pm4py* [8] for working with event log data. For the machine learning models the deep learning library *keras* [4] was leveraged.

Note that to encode the data with Encoding 2 or 3 a machine with the required RAM capacities is necessary. These experiments were conducted on the computer servers (gruenau8) of the institute of computer science at the Humboldt-Universität zu Berlin [3].

The experiments with Encoding 1 have been executed on Google Colab [4] machines.

All of the code and the data sets can be found in this git repository.[2]

To answer the research question, multiple combinations of the two models along with the encodings are possible. In the remainder of this thesis a tuple of a particular model (LSTM, Seq2Seq) and encoding (Encoding 1, Encoding 2, Encoding 3) is referred to as a *configuration.* With the given specification there are six of those *configurations* to be

[2]the git repository with the code and data sets for the conducted experiments

[3]the exact specifications of the HU server can be found here

[4]the exact specifications of the Google Colab server can be found in this notebook

evaluated.
Furthermore, a three letter code of *configuration* plus data set is used to describe the parameters of a given experiment in the evaluation. The code consist of a letter for the model ($\in \{\mathrm{L}, \mathrm{S}\}$), followed by a letter that indicates the encoding ($\in \{1, 2, 3\}$). The last letter ($\in \{\mathrm{B}, \mathrm{T}, \mathrm{A}\}$) states which data set was examined. So for example the three letter code L1B means that the *configuration* (LSTM, Encoding 1) was evaluated on the BPIC-14 log and the code S2T stands for the traffic fines log evaluated on the Seq2Seq model using Encoding 2. The full description of shortcut notation can be found in the appendix.

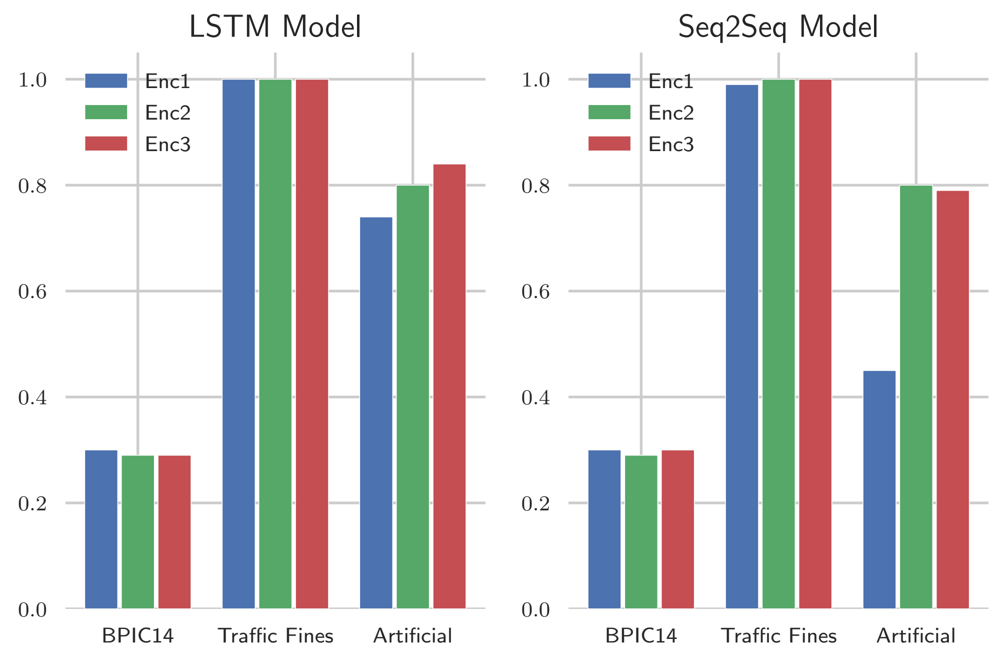
4.3 Results

The results, obtained conducting the research as described in the previous paragraph and quantified according to the performance measures $P1$, $P2$ and $P3$ are described in this section.

Accuracy results

In this paragraph acquired results according to the performance measure $P1$ are presented and ultimately allow for deriving an answer to $RQ2.1$. The accuracy results for all configurations an experimental data sets can be seen in Figure 4.
The left hand part shows the accuracy results for the LSTM model for all three experimental data sets and encodings. The first thing to remark is that the accuracy for a given real-life log do not significantly change with the different encodings. The correct predictions where made in about 30% of the case for the BPIC-14 log (30% for Encoding 1, 29% for Encoding 2 and Encoding 3). For the traffic fines log all predictions where correct, i.e. the accuracy is 100%, for all three of the encodings. For the artificial log the accuracy value are not as equal between the three encodings. They range from only 74% for Encoding 1 up to 85% for Encoding 3.
It is also noticeable that the accuracy values between the data sets are highly diverse with the average (over all three encodings) being as low as 0.29% for the BPIC-14 log or as high as 100% for the traffic fines log. The accuracy for the artificial log is settled in between with an average accuracy of 79%.
On the right hand side the same accuracy results are shown for the Seq2Seq model. For the two real-life logs, the results are similar to the LSTM model. The accuracy for the BPIC-14 log averages at around 30%, with a maximum difference of 1% accuracy between the encodings. The same holds when comparing for the traffic fines log. The average accuracy for all encodings is 100%, with only the first encoding yielding a slightly lower accuracy of 99%.
For the artificial log it can also be observed, that the results here are equally more broadly distributed. For the first encoding less than half of the predictions were correct (45%). The values for the second and third encoding are again similar to the ones for the LSTM model with 80% (Encoding 2) and 79% (Encoding 3) of the predictions being correct.

Figure 4: Accuracy results for the examined configurations and data sets



Re-prediction percentage

In this paragraph the results for the performance $P2$ are presented, which are utilised to deduce answers to $RQ2.2$.

In Table 4 the percentage a second prediction had to be taken into account is visualised. (Note that due to the dynamic behavior, time issues and ambiguity of *Encoding 1* the respective data was not collected for the Seq2Seq model.)

For the LSTM Model in general one can observe that for all the encodings and real-life logs the number of re-predictions is zero or nearly zero. Precisely when evaluating the test set on the BPIC-14 log every prediction for an event set E was already in the set of possible resolutions $\Phi(E)$ for all three encodings. The number of re-predictions for the traffic fines log is almost as lying between 0.0% and 0.004% for the three possible *configurations*.

When it comes to the artificial log, noticeably, the number of re-predictions is higher and differently distributed. For *Encoding 1* every fifth prediction made, needed to be reconsidered. Regarding *Encoding 2* and *3* 95% of the initial predictions were not a possible resolution for the given event set. Summarised for the LSTM model the re-prediction mean in total amounts to 0.24%. When only considering the two real-life logs it is as low as 0.001%.

For the *Seq2Seq* model the numbers look differently. For the BPIC-14 log almost half the predictions where initially not a possible resolutions for both *Encoding 2* and *3* (44%, 46%). In case of the traffic fines log the number of re-predictions with 3% for both encodings is ten times higher then the respective value for the LSTM model. Yet for the artificial log the Seq2Seq yields best results with 67% of re-predictions for *Encoding 2*

Table 4: Reprediction percentage

LSTM

Enc1			Enc2			Enc3		
BPIC14	Traffic	Artificial	BPIC14	Traffic	Artificial	BPIC14	Traffic	Artificial
0.0	0.0	0.21	**0.0**	**0.004**	0.95	**0.0**	**0.002**	0.95

Seq2Seq

Enc1			Enc2			Enc3		
BPIC14	Traffic	Artificial	BPIC14	Traffic	Artificial	BPIC14	Traffic	Artificial
-	-	-	0.44	0.03	**0.67**	0.46	0.03	**0.70**

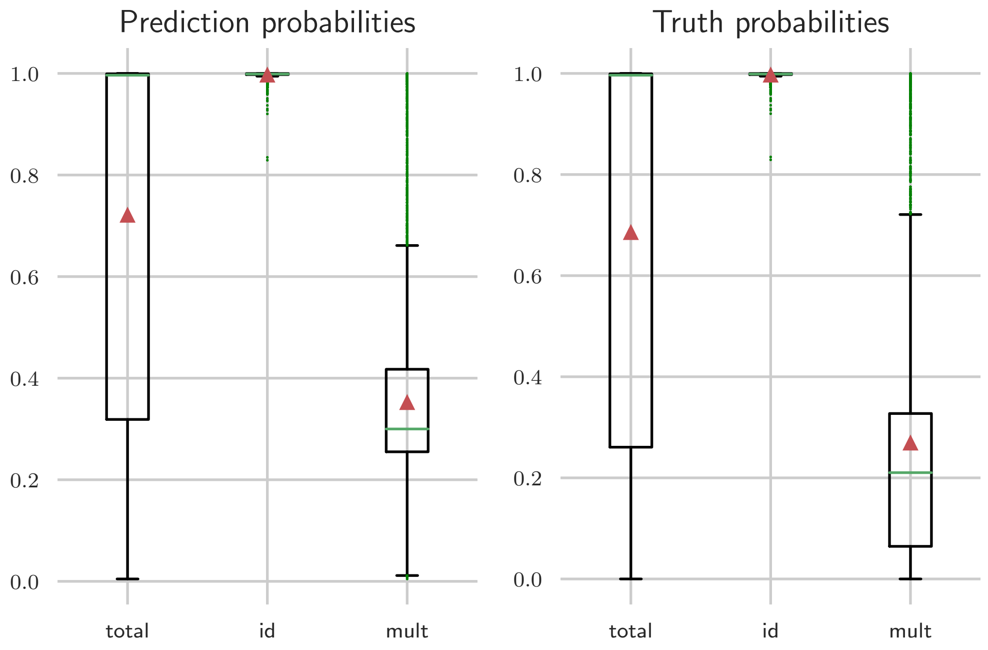
and 70% for *Encoding 3*.
With that, for the Seq2Seq model the average number of re-predictions for all encodings and data sets with 0.39% is 15% higher than for the LSTM model. When not taking into account the artificial log, the average re-prediction percentage, with 24%, for the Seq2Seq model is 240 times higher than for the LSTM model.
It also appears that for the Seq2Seq model there are no significant differences in the number of re-predictions throughout the encodings and data sets. For the LSTM model significant differences only appear for the artificial log.

Analyzing the probability densities in the prediction space

To gain more insight in prediction accuracy and stability and to obtain answers to $RQ2.3$ the probability distributions produced by the *configurations* are investigated in terms of $P3$ in this paragraph.
To begin with for a given *configuration* and data set the distribution of the probability scores assigned to the predicted event set and to the ground truth are compared.
In Figure 5 the probability scores for the *configuration* (LSTM, Encoding 1) evaluated on the BPIC-14 log are shown exemplary. In the left half of the figure the probability scores for the predicted resolutions are shown.
Here *total* states that all probability scores were considered. The term *id* means that the probability scores only for predictions for event sets with length 1 are considered and likewise *mult* means that probability scores of predictions made for all event sets with length > 1 have been considered.
In the right sub-figure the same subsets of probabilities are shown for the probabilities the model assigned to the true possible resolution $\Phi^\star$.
In general it appears that both the prediction and the truth probabilities are distributed similarly, especially for the total predictions and the identity predictions made. The total prediction probabilities show a wide range of values and are equal between the prediction and the truth probabilities.

Only minor differences in truth and prediction probabilities in general for the rest of

Figure 5: Probability densities of predicted and ground truth (L1B)

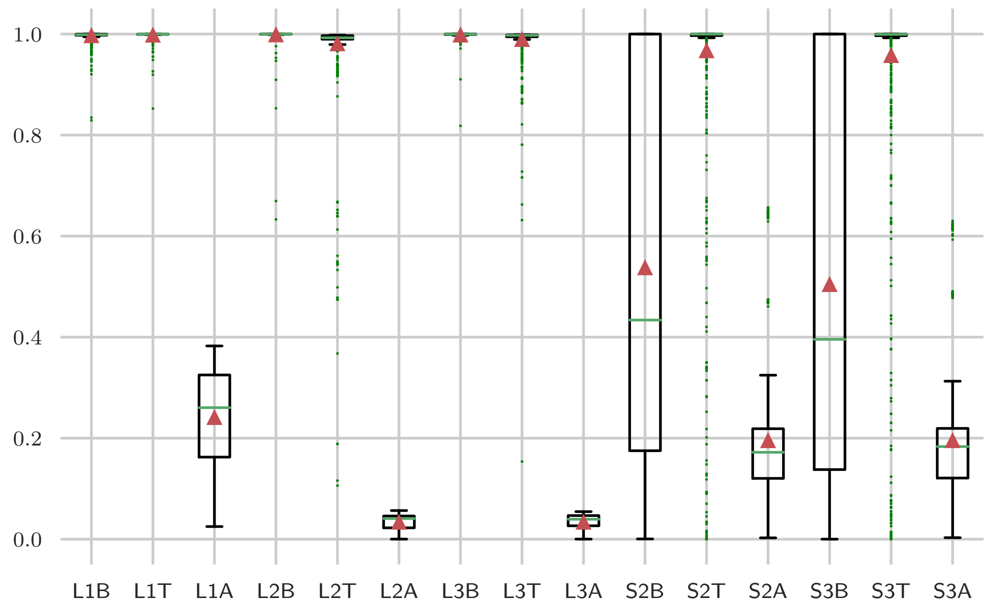


the figures (s. Appendix 5). In detail for all the comparisons of prediction and truth probabilities the distributions are mostly even and only slightly differing in values.
For instance for all the *configurations* and data sets the difference between the respective means of the probabilities of the predictions and those associated to the truth is at maximum 0.04, when considering the total probabilities. For the mult probabilities this maximum is doubled with 0.08. For the id probabilities the the mean differences between prediction and truth probabilities are exactly 0.00 for all the 15 experiments.
In fact when only considering the id probabilities the distributions are the same for all the plots, i.e. the differences between the boxplot characteristics (min, lower quartile, median, mean, upper quartile, max) for the predicted and truth probability are exactly 0.00.
When looking at the total probabilities the largest differences are found for the L2B and L3B experiments, where the difference of the lower whiskers is 12%. And for the L2B experiment the lower quartile difference lies at 10%. For the rest of the boxplot values the differences never exceed 0.04).
The biggest variance in predicted to truth probability is present when only comparing the probability values for the mult (length ≥ 2) event sets. For all experiments done on the BPIC-14 log the difference here are rather high, for example the difference between the lower whisker for ground and truth of L3B amounts to 34% and the difference between lower quartile for L1B and L2B is about 20 %. For all *configurations* involving other data sets then the BPIC-14 log the difference between any box plot characteristic is non present, i.e. 0.0.

In total the only difference between truth and prediction probabilities throughout all the conducted experiments is 0.0 for all data sets except for the BPIC-14 log, where the variance between the boxplot measures is as described above.
The remaining 14 figures can be found in the appendix 5.

From comparing the probabilities of the predicted possible resolutions with the ones assigned to the true resolution, the prediction probabilities in the following will be compared for event sets of different size.
In Figure 6 the probability scores of the identity predictions (predictions for event

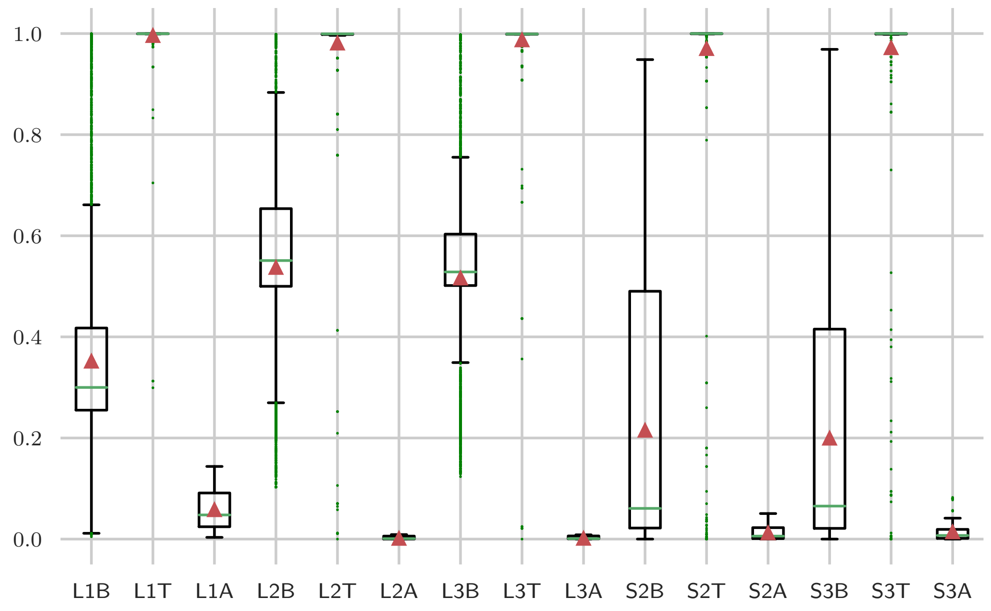
Figure 6: Probability densities for all identity mappings



sets of length 1) for the test set are presented for all experiments conducted, i.e. all configurations and event logs. For an event set $\{a\}$ where $a \in \mathcal{A}$ the identity mapping, i.e the mapping $\{a\} \implies a$, as a resolution is trivial, as in this case there exists only exactly one possible resolution for the given event set.
It stands out that for the LSTM model and the real-life logs the identity mapping was learned (almost) perfectly for all three encodings, meaning the probability mass for predictions accumulates around 1.0 with the lowest mean at 0.98 for the L2T experiments.
For the artificial log the LSTM model produced less certain predictions for identity mappings. The first encoding achieved best results (L1A) with a mean of 0.25 and the inter quartile range (IQR) ranging between 0.16 and 0.33. For the encodings 2 and 3 (L2A, L3A) here the values are lowest among all *configurations* with a mean of 0.03 for both and the IQR ranging only from about 0.02 to 0.05.
For the sequence to sequence model the identity mappings have not been learned as certain. For the the traffic fines log the probability masses are distributed similar to the LSTM model; the lowest mean just lying a little lower with 0.96 for the S3T experiment.

For the BPIC-14 log, however, the probabilities are distributed significantly more versatile. Here for both encodings (S2B, S3B) the means lie at around 50% (0.54, 0.51) and the IQR taking more than 80% of the possible probability space, ranging from 0.18, 0.14 respectively to 1.00.
When looking at the values for the artificial log (S2A, S3A), one can see that the probabilities are more certain than for the LSTM. In both cases the mean is 0.20 and the IQR ranges from 0.12 to 0.20.
Likewise, in Figure 7 the probability scores for the predictions for event sets of length ≥ 2, i.e. for uncertain event sets, are shown.
The first thing to notice is that for the traffic fines log the probability score is close to 100% for all *configurations*, with the means lying between 0.97 and 1.00.
The probability scores are on the other side of the spectrum for the artificial log. For encodings 2 and 3 on both models (L2A, L3A, S2A, S3A) the means lie between 0.00 and 0.03 and the IQR taking at most 3% of the probability space. There is one exception though. L1A has significantly higher probabilities with an at lest two times larger mean of 0.06 and an IQR between 0.02 and 0.09.
Lastly there is the BPIC-14 log, which will be described separately for each model.

Figure 7: Probability densities for all multi mappings



For the LSTM model the distributions of probability scores for Encoding 2 and 3 (L2B, L3B) are somewhat similar with a mean of 0.53 and 0.55. The Encoding 1 (L1B) is less certain with predictions on average with a mean of only 0.35.
For the Seq2Seq model Encoding 2 and 3 behave similar and are assigning the chosen resolution a probability of 0.22 / 0.20 respectively on average.
Note that here for the Seq2Seq model (S2B, S3B) the IQR range is more than three times as large (from 0.02 to 0.49/0.42) as for the respective LSTM probabilities (L1B,

L2B, L3B) which are ranging between 0.26 to 0.42, 0.50 to 0.65 and 0.50 to 0.60 and thus take between 10% and 16% of the probability space.

Answers to $RQ2.4$ will be deduced from the results presented in each of the paragraphs above and taken into account in the next section, the *interpretation* of the results.

4.4 Discussion

The answers to the research questions $RQ2.1$ to 2.4 that can be deduced from the experimental results of the research presented in the previous section are discussed and set into context with the assumptions.

RQ2.1

In order to give an answer to *RQ2.1*, reconsider Figure 4. For the real-life logs there is no significant difference in terms of prediction accuracy measured by $P1$ for the two models. For the artificial log there is a higher variance in the accuracies and it seems like the LSTM model performing better on average for all the three encodings.
This and the outlier of the prediction accuracy for the Seq2Seq model and *Encoding 1* can be explained due to the small size of the log. With only thousand traces, of which about half where uncertain the training and test set sizes, when splitting 80 / 20, where rather small compared to the real-life event logs. Thus the structures on the data that can be learned from are limited, e.g. in terms of variance of trace length and types of uncertain event sets. Also there is a higher discrepancy, when shuffling the data and randomly assigning test and train set, i.e. it is more likely to produce a train / test set that is non-representative of the actual data structure present in the whole event log.
Summarised for the real-life logs no significant difference as of $P1$ was measured. The results for the artificial log generally imply the same, as for the real-life logs. Due to the small size of the data set, however the these results need to be interpreted with caution, hence a comprehensive study with larger artificial data sets should be considered.

RQ2.2

Regarding the research question 2.2, it can be clearly seen that the re-prediction scores are significantly higher for the Seq2Seq model than for the LSTM model. Especially for the real-life logs the LSTM model has close to none re-predictions, whereas the Seq2Seq model needs to re-predict in at least 3% and at most almost 50% of the cases.
In general the *RNN with LSTM cells* is designed to predict sequences of the same length as the respective input sequence. In contrary the *Seq2Seq* model is capable of predicting an arbitrary length sequence for any kind of length input sequence.
One can think for example of the the Seq2Seq model predicting for the sequence $\sigma = \langle A, B, C, D \rangle$ and being as far as having already predicted the sequence A, B. Assuming that the model has learned that under these conditions, it is very probable that the sequence will end with the activity D and thus predicting A, B, D.
In the LSTM model however this “early stopping” of sequences is not possible as the

input and predicted sequences will always be of equal length. This is likely to be one reason, why the "plain" LSTM model performs superior here. This more powerful design however gives the model additional degrees of freedom, which lead to potentially more errors.
For the artificial log, as in the previous paragraph, the results are less consistent between the different encodings. This again is likely to be due to the aforementioned reasons. Mere numbers would suggest that the *configuration* (LSTM, Encoding 1) performs best, with 21%.
In total, due to the consistent results for the real-life event logs the LSTM log can be regarded as performing superior and being having learned more accurately to map to possible resolutions only.

RQ2.3

The fact that the artificial log contains little data could also be an explanation for probabilities scores being less certain throughout all models and encodings compared to the other data sets for the identity mappings as well as for the multi-length mappings. The model may not have seen enough data points to make an educated prediction.
The fact that the LSTM learns the identity mapping (nearly) perfectly (except for the artificial log; see reasoning above), which is the desired behavior, since for any event set of length 1 there is only exactly one possible resolution.
Given that the Seq2Seq performed worse learning this necessary behavior, is likely to be conditioned on the reasoning made in the previous passage. As a consequence it is likely that with less interfering validation methods, i.e. not taking predictions until an actual possible resolution is predicted, may lead to lower accuracy scores for the Seq2Seq based configurations.
Thus for research question 2.3, it follows that the *LSTM* performs better, as it appears to learn trivial data structures more accurately (identity mapping) and in general for the rest of the probability scores there are no significant difference from the ones for the *Seq2Seq* model.

RQ2.4

When concluding the above in the aspect on evaluating it for the three different encodings, it appears that throughout all measures no significant differences between the different encodings was found, e.g. the prediction accuracy, the re-prediction percentage and the learning of identity mappings as well as the distribution of probability scores is consistent throughout all the encodings.
Only for the artificial log the results appear to be dependent on the chosen encoding and would either suggest the *Extended Event Set Encoding* (measure $P1$) or the *Activity Space Encoding* (measure $P2$).
Since these results are, however, to be taken cautiously the encoding is superior. Due to the significantly smaller encoding space, for input as well as for output vectors, the space and time complexity is greatly reduced, the *Activity Space Encoding* tractable in practical applications.

Summarising the discussion above, a comprehensive answer for research question 2 in general can be given. The models do perform equally given mere prediction accuracy. However, when trying to get some insight into the actual behavior learned by the model ($RQ2.2, RQ2.3$), the LSTM clearly strikes as superior, probably due to more context related regulations (e.g. fixed sequence length) and thus should be favored. The encodings do not appear to give a scientifically valid, measurable difference as of $P1$ to $P3$. Due to the lower space and time complexity, however, the *Activity Space Encoding* should be preferred.

5 Conclusion

In this research project it was, shown that artificial neural networks can be used to solve the task of partial order resolution in process mining by translating the problem to a sequence to sequence prediction task for machine learning methods.
It also came clear that the model leveraging the most of the a priori knowledge in general, yields more certain results, when however both models solved the problem equally accurate. In total this leads to the conclusion to prefer the LSTM model over the Seq2Seq model and the *Activity Space Encoding* over the other encodings for the task of *partial order resolution* in process mining.
Though the results were clear, the research of this project can be improved in multiple ways.
First of all, the performance measures $P.1$ to $P.3$ can be improved. $P.1$ could be advanced such that a more detailed accuracy measure of the prediction of possible resolutions is achieved. For instance not only measuring a correct or wrong prediction, i.e. *0-1-loss*, but to somehow include the extent to which the prediction is inaccurate, e.g. by relating it to the length of the trace and / or the uncertain sequence. This could be done, for instance by incorporating some sort of string distance measure, e.g. the levenshtein distance. Performance measure $P.2$ could be improved by counting not only *if* a re-prediction was done for each predicted event set, but to count the total number of re-predictions. Both of these changes would make future results more comparable.
Furthermore the experiments conducted in this research lack completeness, i.e. a broader range of event logs could be evaluated, in order to more strictly assure the obtained results. Since the experimental pipeline is now built, still under the constraint of being improved, experiments with multiple different real life logs grouped into sets with equal characteristics, e.g. same size or average trace length, so to say a survey, could answer those questions.
Furthermore in [32] van der Aa et. al. already implemented a ready to use pipeline for synthesizing artificial logs with the hyperparameters adjustable to determine for instance the rate of uncertainty or the log size. With that one could similar to the idea above undertake a further study on artificial logs themselves or enrich the sets of logs from the idea above with more data.
Additionally the accuracy results as measured by P.1 and asked for by RQ1 have not been

compared to any other previous results. There could have been at least the comparison with the accuracy of the stochastic models proposed in [32].
What also still is a big limitation to the given encodings is the natural combinatorial space explosion for *Encoding 2* and *3*. For the given logs and their respective activity universe size, the largest size of uncertain event sets, that was still admissible was four. This greatly limited the idea of 1) introducing further uncertainty into the logs and 2) generating more and richer training data by abstracting certain temporal units from the time stamps.
Also, the problem of trace probability is addressed by van der Aalst et al. in [33], stating that considering all possible traces made from the activity set $\mathcal{A}$ is not feasible and also improbable. This undermines the idea of shrinking the feature space, by only considering event sets and their possible resolutions that are occurring in the event log.
Another solution to this could be to compress the data by making use of the embeddings introduced in [10] for *partial order resolution* with artificial neural networks.
In general also in the recent years more sophisticated neural network models have emerged. There are for *Bidirectional RNNs* to further enhance LSTM models. For sequence to sequence learning for instance *Beam Search* and the *Attention Mechanisms* have been proposed. Here especially the transformer architectures could yield good results. [14] [9]
Lastly an important part of the work that van der Aa et. al. did in [32] was, not only to resolve partial orders for uncertain traces, but to also additionally relate this to a conformance checking task and reducing the state-of-the-art error by 60%. This was missed out during the work of this thesis and should thus be considered to work on in the future.

References

[1] Wil van der Aalst. Process mining : data science in action / by wil van der aalst, 2016.

[2] Rafael Accorsi and Thomas Stocker. On the exploitation of process mining for security audits: the conformance checking case. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 1709–1716, 2012.

[3] Arya Adriansyah, Boudewijn F van Dongen, and Wil MP van der Aalst. Conformance checking using cost-based fitness analysis. In *2011 IEEE 15th International Enterprise Distributed Object Computing Conference*, pages 55–64. IEEE, 2011.

[4] Alessandro Berti et al. Keras. `https://github.com/fchollet/keras`, 2015.

[5] Ivan Beschastnikh, Yuriy Brun, Michael D Ernst, Arvind Krishnamurthy, and Thomas E Anderson. Mining temporal invariants from partially ordered logs. In *Managing Large-scale Systems via the Analysis of System Logs and the Application of Machine Learning Techniques*, pages 1–10. Association for Computing Machinery, 2011.

[6] RP Jagadeesh Chandra Bose, Ronny S Mans, and Wil MP van der Aalst. Wanna improve process mining results? In *2013 IEEE symposium on computational intelligence and data mining (CIDM)*, pages 127–134. IEEE, 2013.

[7] Josep Carmona, Boudewijn van Dongen, Andreas Solti, and Matthias Weidlich. *Conformance Checking*. Springer, 2018.

[8] François Chollet et al. pm4py. `https://github.com/pm4py/pm4py-core`, 2019.

[9] Community. Papers with code: Machine translation. `https://paperswithcode.com/task/machine-translation`. Accessed: 09.06.2021.

[10] Pieter De Koninck, Seppe vanden Broucke, and Jochen De Weerdt. act2vec, trace2vec, log2vec, and model2vec: Representation learning for business processes. In *International Conference on Business Process Management*, pages 305–321. Springer, 2018.

[11] M. (Massimiliano) de Leoni and Felix Mannhardt. Road traffic fine management process, Feb 2015.

[12] Massimiliano de Leoni, Giacomo Lanciano, and Andrea Marrella. Aligning partially-ordered process-execution traces and models using automated planning. In *Twenty-Eighth International Conference on Automated Planning and Scheduling*, 2018.

[13] Prabhakar M Dixit, Suriadi Suriadi, Robert Andrews, Moe T Wynn, Arthur HM ter Hofstede, Joos CAM Buijs, and Wil MP van der Aalst. Detection and interactive repair of event ordering imperfection in process logs. In *International Conference on Advanced Information Systems Engineering*, pages 274–290. Springer, 2018.

[14] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, 2019.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[16] Theresia Gschwandtner, Johannes Gärtner, Wolfgang Aigner, and Silvia Miksch. A taxonomy of dirty time-oriented data. In *International Conference on Availability, Reliability, and Security*, pages 58–72. Springer, 2012.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[18] Eric Koskinen and John Jannotti. Borderpatrol: isolating events for black-box tracing. *ACM SIGOPS Operating Systems Review*, 42(4):191–203, 2008.

[19] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[20] Xiaodong Liu, Kevin Duh, Liyuan Liu, and Jianfeng Gao. Very deep transformers for neural machine translation. *arXiv preprint arXiv:2008.07772*, 2020.

[21] Xixi Lu, Dirk Fahland, and Wil MP van der Aalst. Conformance checking based on partially ordered event data. In *International conference on business process management*, pages 75–88. Springer, 2014.

[22] Xixi Lu, Ronny S Mans, Dirk Fahland, and Wil MP van der Aalst. Conformance checking in healthcare based on partially ordered event data. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–8. IEEE, 2014.

[23] Ronny S Mans, Wil MP van der Aalst, and Rob JB Vanwersch. Data quality issues. In *Process Mining in Healthcare*, pages 79–88. Springer, 2015.

[24] Ronny S Mans, Wil MP van der Aalst, Rob JB Vanwersch, and Arnold J Moleman. Process mining in healthcare: Data challenges when answering frequently posed questions. In *Process Support and Knowledge Representation in Health Care*, pages 140–153. Springer, 2012.

[25] Tom M Mitchell et al. Machine learning. *None*, 1997.

[26] Christopher Mutschler and Michael Philippsen. Reliable speculative processing of out-of-order event streams in generic publish/subscribe middlewares. In *Proceedings of the 7th ACM international conference on Distributed event-based systems*, pages 147–158, 2013.

[27] Jari Peeperkorn, Seppe vanden Broucke, and Jochen De Weerdt. Conformance checking using activity and trace embeddings. In *International Conference on Business Process Management*, pages 105–121. Springer, 2020.

[28] Arik Senderovich, Andreas Rogge-Solti, Avigdor Gal, Jan Mendling, and Avishai Mandelbaum. The road from sensor data to process instances via interaction mining. In *International Conference on Advanced Information Systems Engineering*, pages 257–273. Springer, 2016.

[29] Suriadi Suriadi, Robert Andrews, Arthur HM ter Hofstede, and Moe Thandar Wynn. Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. *Information Systems*, 64:132–150, 2017.

[30] Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. Predictive business process monitoring with lstm neural networks. In *International Conference on Advanced Information Systems Engineering*, pages 477–492. Springer, 2017.

[31] Thanh Tran, Charles Sutton, Richard Cocci, Yanming Nie, Yanlei Diao, and Prashant Shenoy. Probabilistic inference over rfid streams in mobile environments. In *2009 IEEE 25th International Conference on Data Engineering*, pages 1096–1107. IEEE, 2009.

[32] Han van der Aa, Henrik Leopold, and Matthias Weidlich. Partial order resolution of event logs for process conformance checking. *Decision Support Systems*, 136:113347, 2020.

[33] Wil Van Der Aalst, Arya Adriansyah, Ana Karla Alves De Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose, Peter Van Den Brand, Ronald Brandtjen, Joos Buijs, et al. Process mining manifesto. In *International Conference on Business Process Management*, pages 169–194. Springer, 2011.

[34] B.F. (Boudewijn) van Dongen. Bpi challenge 2014, Apr 2014.

[35] B.F. (Boudewijn) van Dongen. Bpi challenge 2015, May 2015.

Appendix: Additional Models, Formulas and Graphs

The shortcut notation for the models is used as seen below:

- ◇ LSTM $\implies$ L
- ◇ Seq2Seq $\implies$ S

The three encodings are also referred to as:

- ◇ *Activity Space Encoding* $\implies$ Encoding 1
- ◇ *Event Set Encoding* $\implies$ Encoding 2
- ◇ *Extended Event Set Encoding* $\implies$ Encoding 3

For the data sets their respective first letter is used. So B stand for the BPIC-14, T for the traffic fines and A for the artificial log.
The concepts and notations shown below are taken from or inspired by [15], [14] and [30].

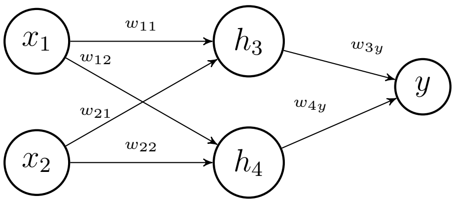


Figure 8: a simple feedforward neural network

The forward pass, for the general model depicted in Figure 2, is given by:

$$\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} \qquad \mathbf{h}^{(t)} = g_1(\mathbf{a}^{(t)})$$
$$\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)} \qquad \hat{\mathbf{y}}^{(t)} = g_2(\mathbf{o}^{(t)})$$

Here g_1, g_2 can be arbitrary activation functions, e.g. $tanh(x)$ and $softmax(x)$ and $\mathbf{b}, \mathbf{c}$ are bias vectors.
The forward pass, for an RNN using LSTM cells as in Figure 3, are given by:

$$f_\sigma^{(t)} = sigmoid(W_f \cdot [h^{(t-1)}, x^{(t)}] + b_f) \qquad i_\sigma^{(t)} = sigmoid(W_i \cdot [h^{(t-1)}, x^{(t)}] + b_i)$$
$$o_\sigma^{(t)} = sigmoid(W_o \cdot [h^{(t-1)}, x^{(t)}] + b_o) \qquad g^{(t)} = activation(W_g \cdot [h^{(t-1)}, x^{(t)}] + b_g)$$
$$C^{(t)} = f^{(t)} \otimes C^{(t-1)} \oplus i^{(t)} \otimes g^{(t)} \qquad h^{(t)} = o^{(t)} \otimes tanh(C^{(t)})$$

Here W_a, b_a are the associated weight matrices and bias vectors; $a \in \{f, i, o, C\}$ and *activation* can be any activation function, usually chosen to be *tanh*. The symbols $\oplus$ and $\otimes$ denote the element-wise addition and multiplication respectively.

Appendix: Remaining figures

Figure 9: Probability densities of predicted and ground truth (L1T)

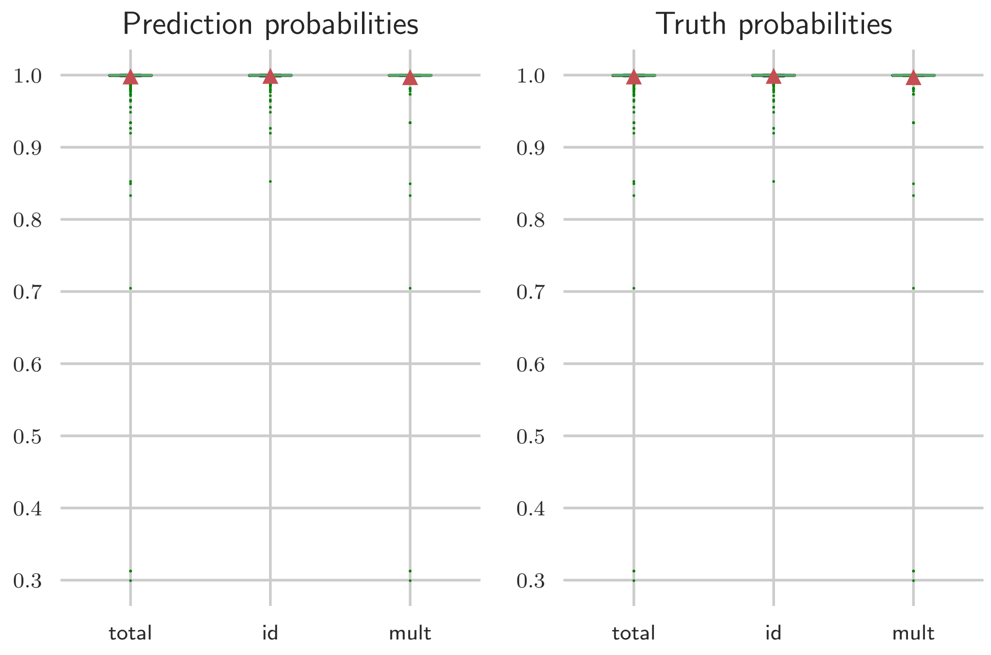


Figure 10: Probability densities of predicted and ground truth (L1A)

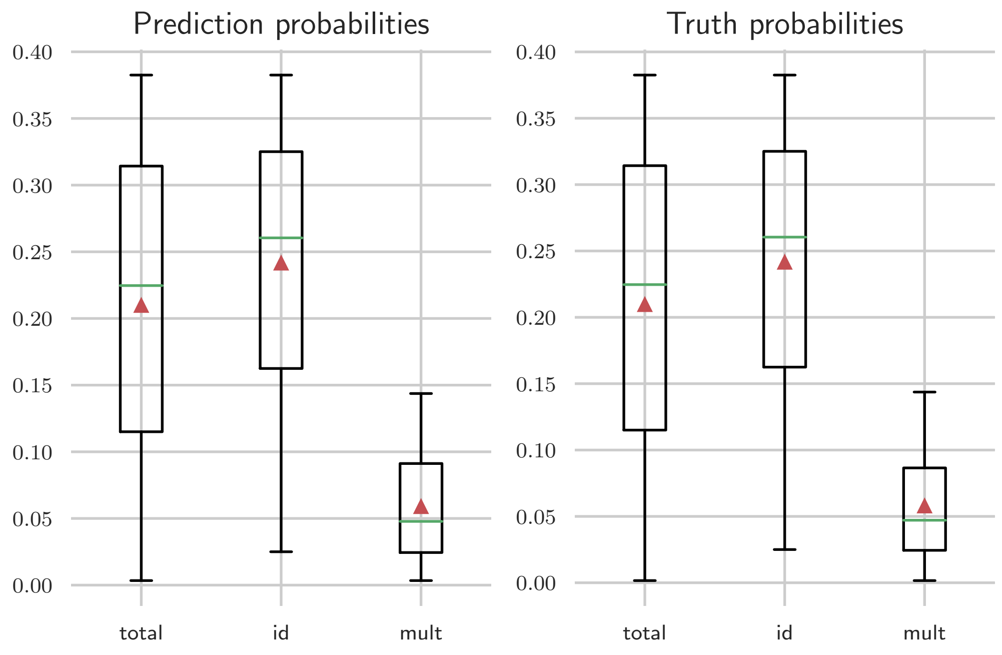


Figure 11: Probability densities of predicted and ground truth (L2B)

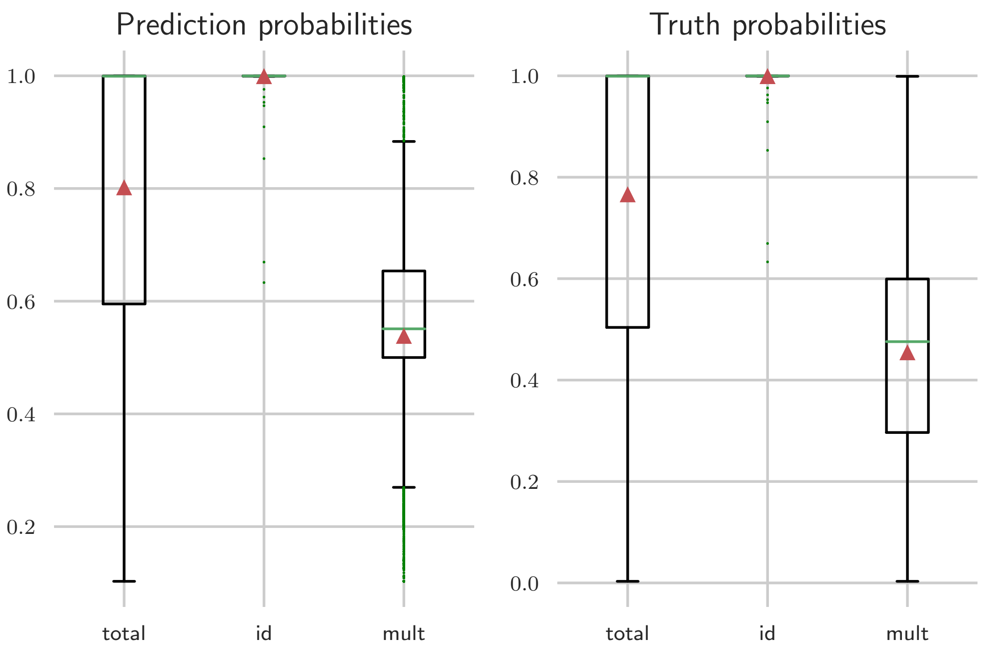


Figure 12: Probability densities of predicted and ground truth (L2T)

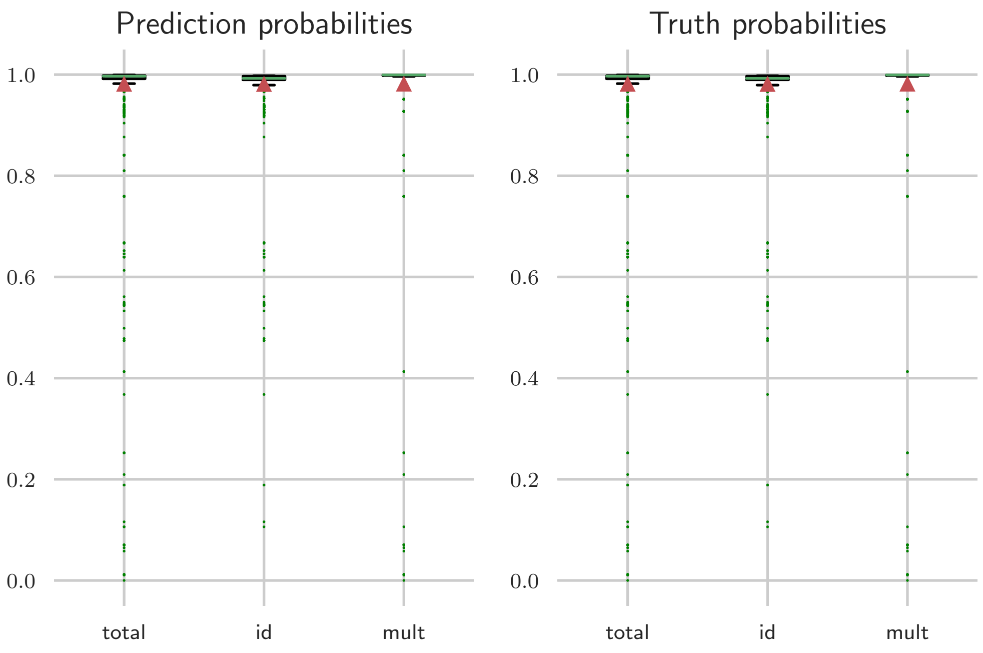


Figure 13: Probability densities of predicted and ground truth (L2A)

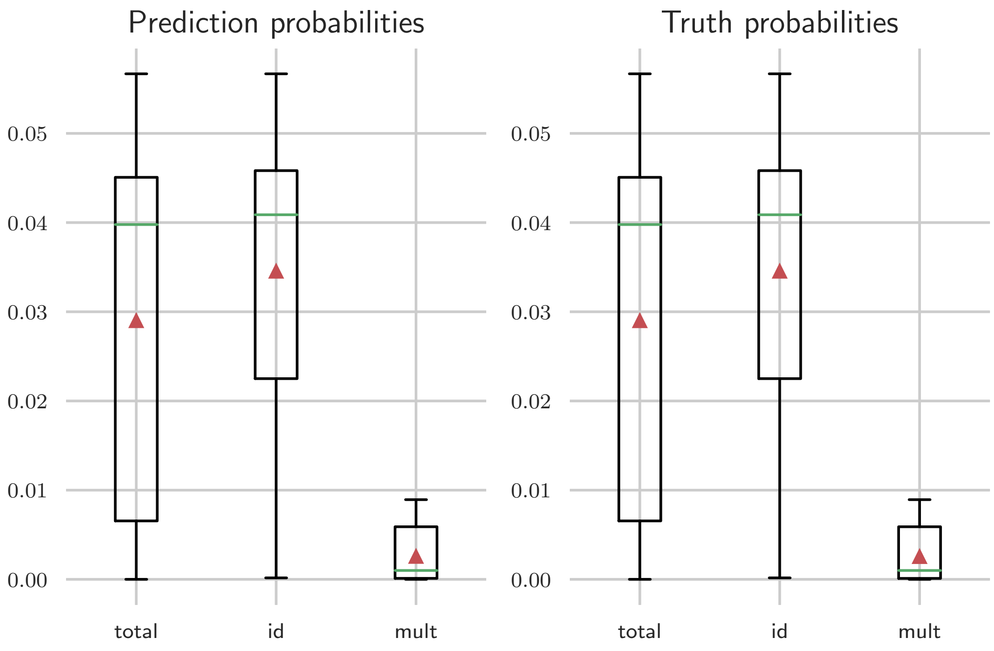


Figure 14: Probability densities of predicted and ground truth (L3B)

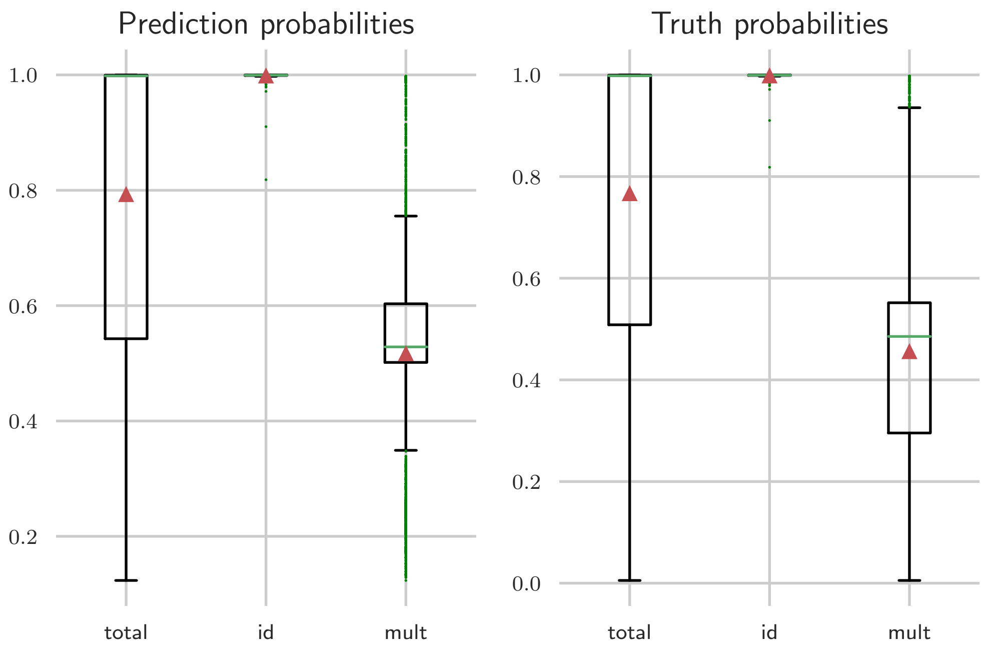


Figure 15: Probability densities of predicted and ground truth (L3T)

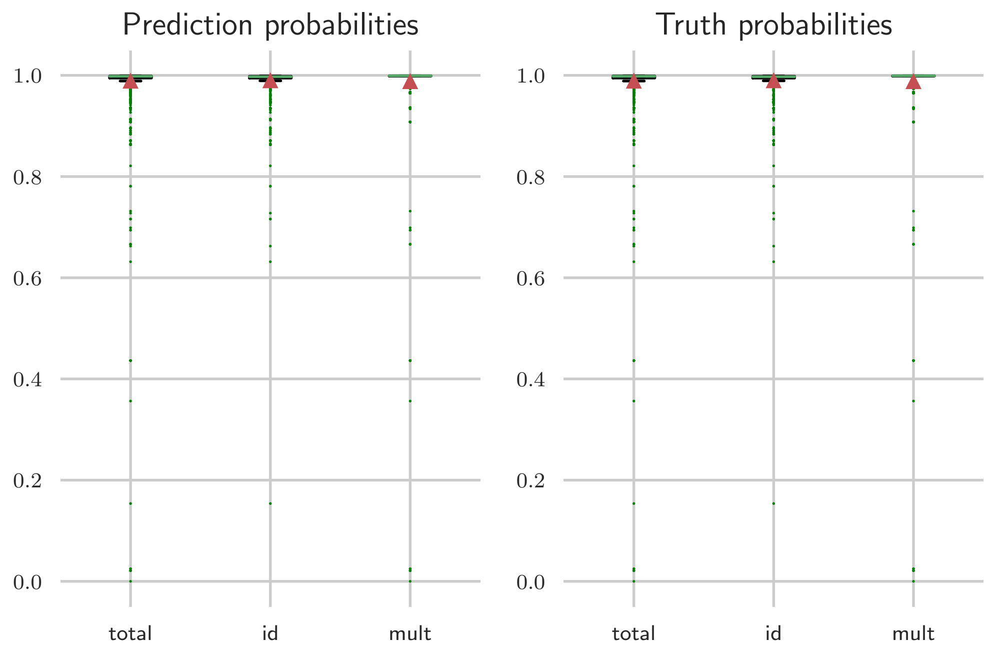


Figure 16: Probability densities of predicted and ground truth (L3A)

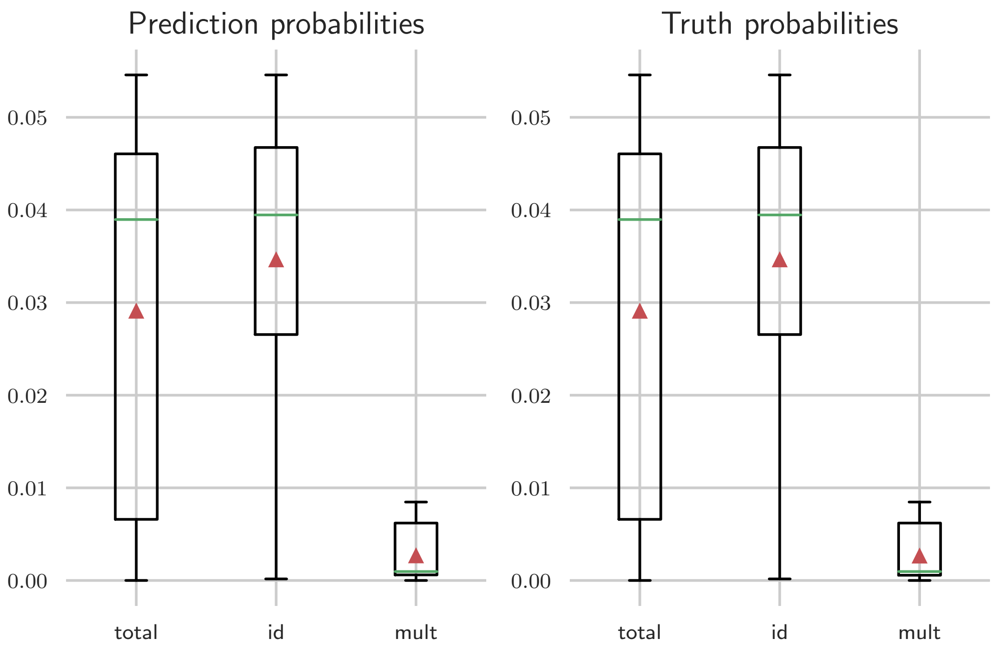


Figure 17: Probability densities of predicted and ground truth (S2B)

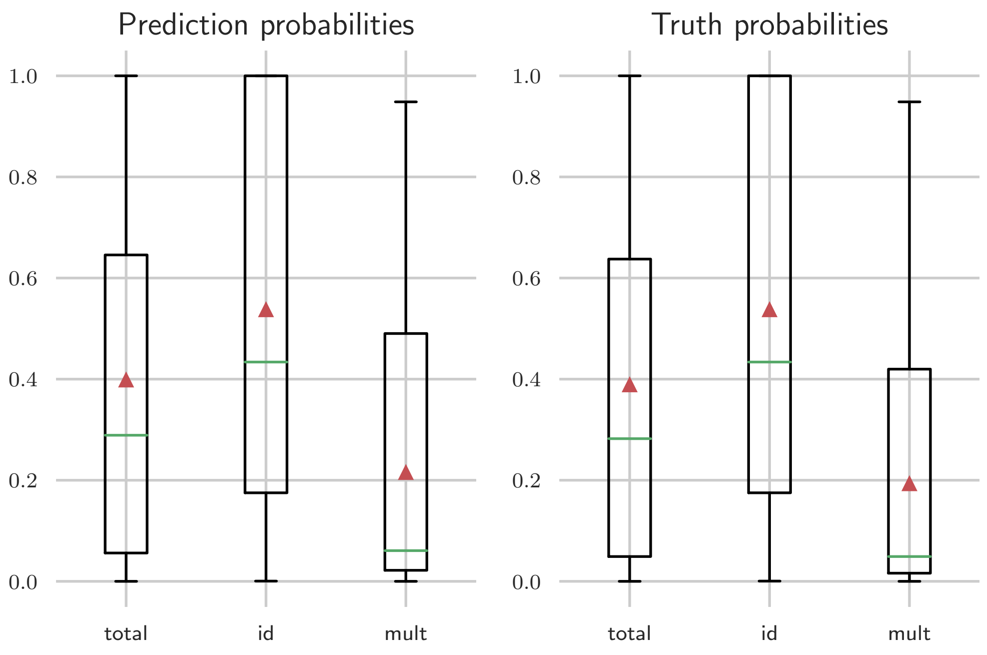


Figure 18: Probability densities of predicted and ground truth (S2T)

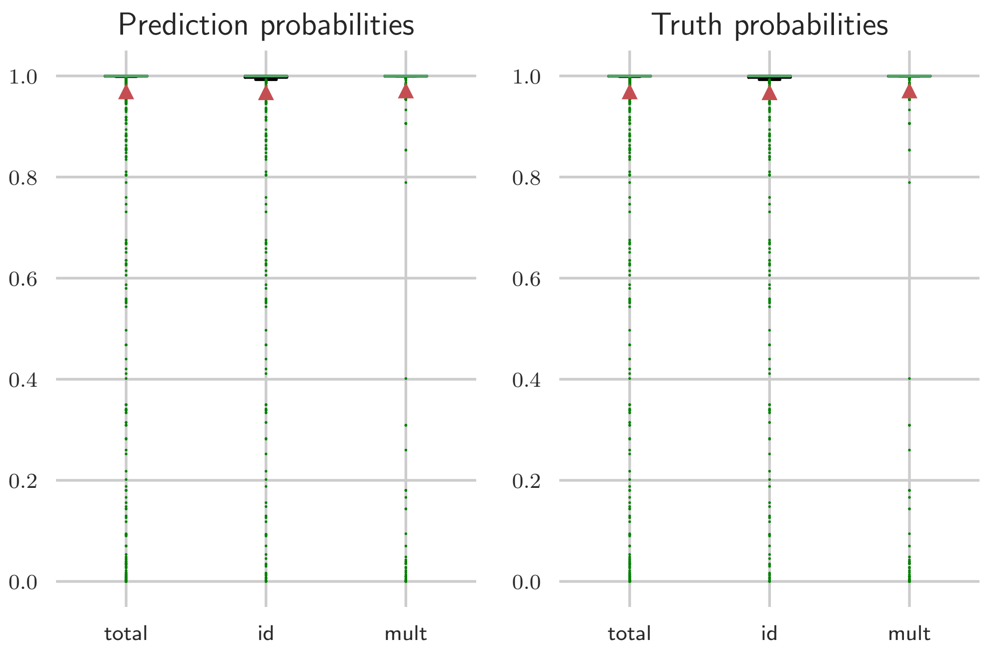


Figure 19: Probability densities of predicted and ground truth (S2A)

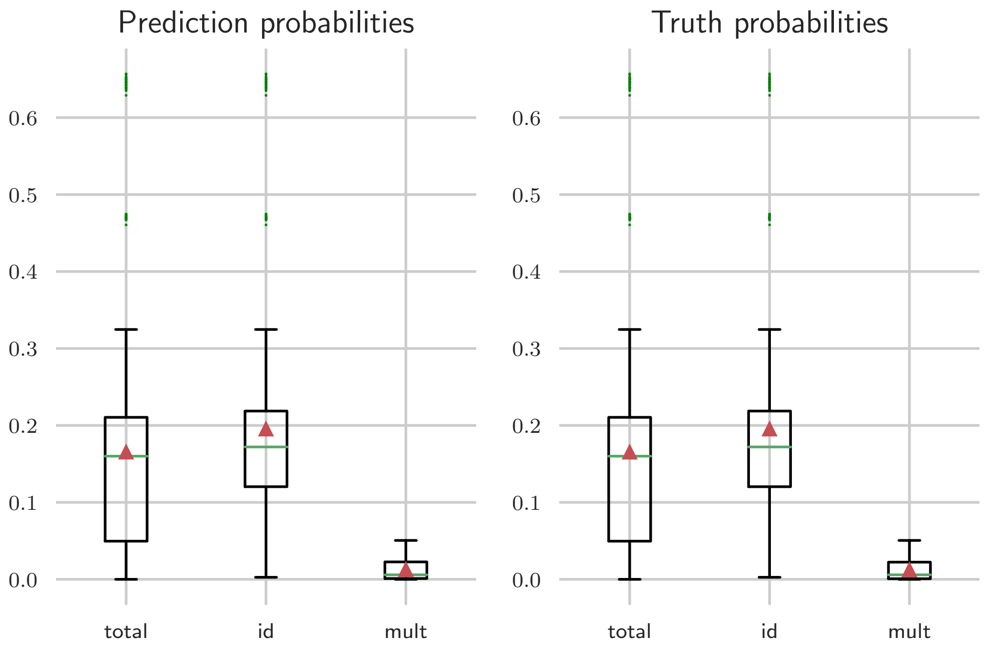


Figure 20: Probability densities of predicted and ground truth (S3B)

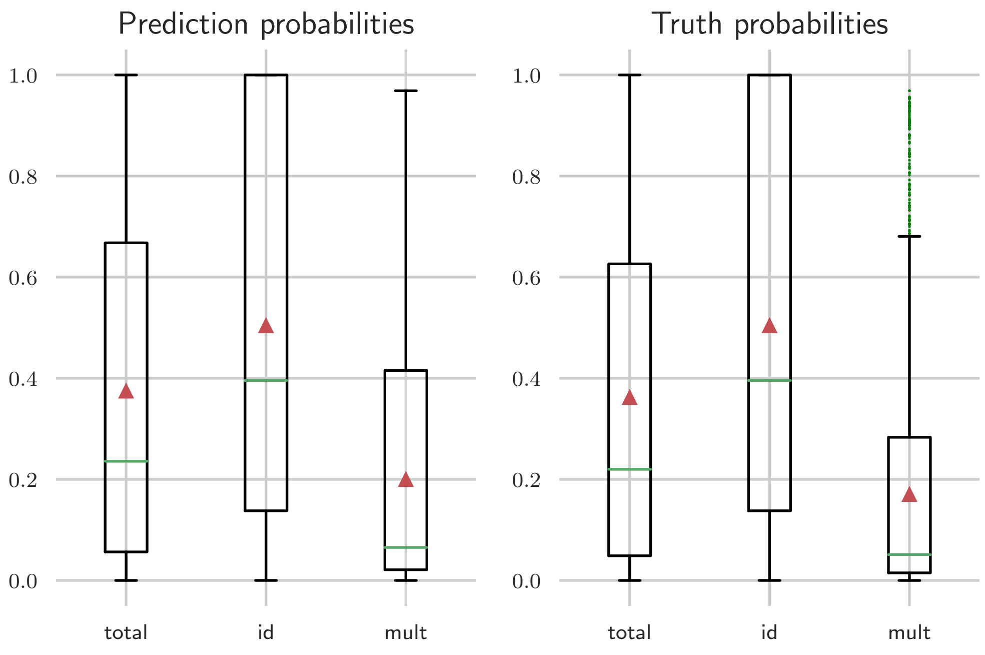


Figure 21: Probability densities of predicted and ground truth (S3T)

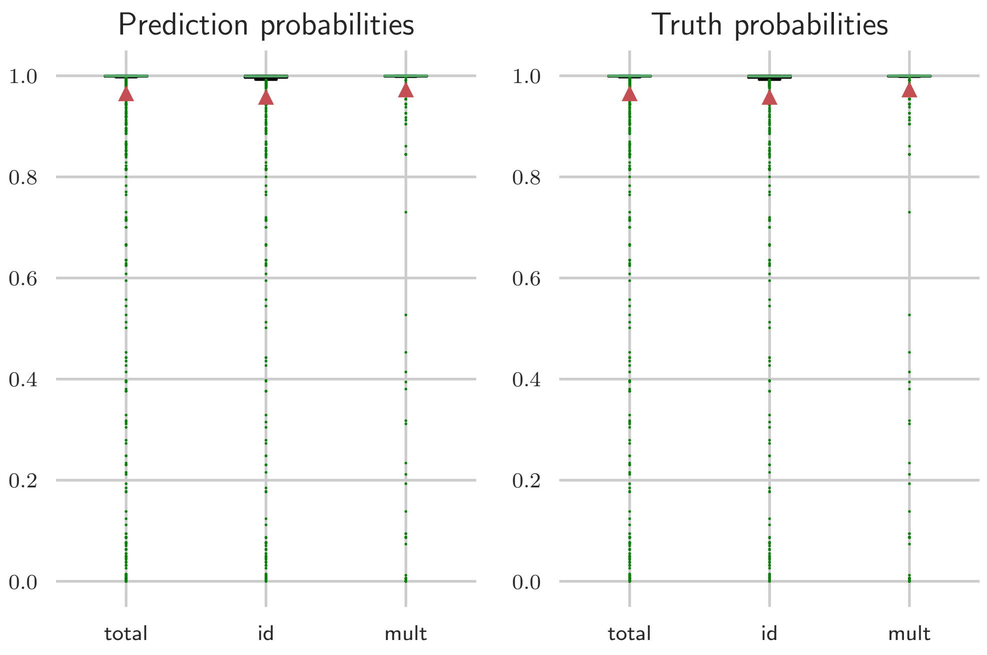
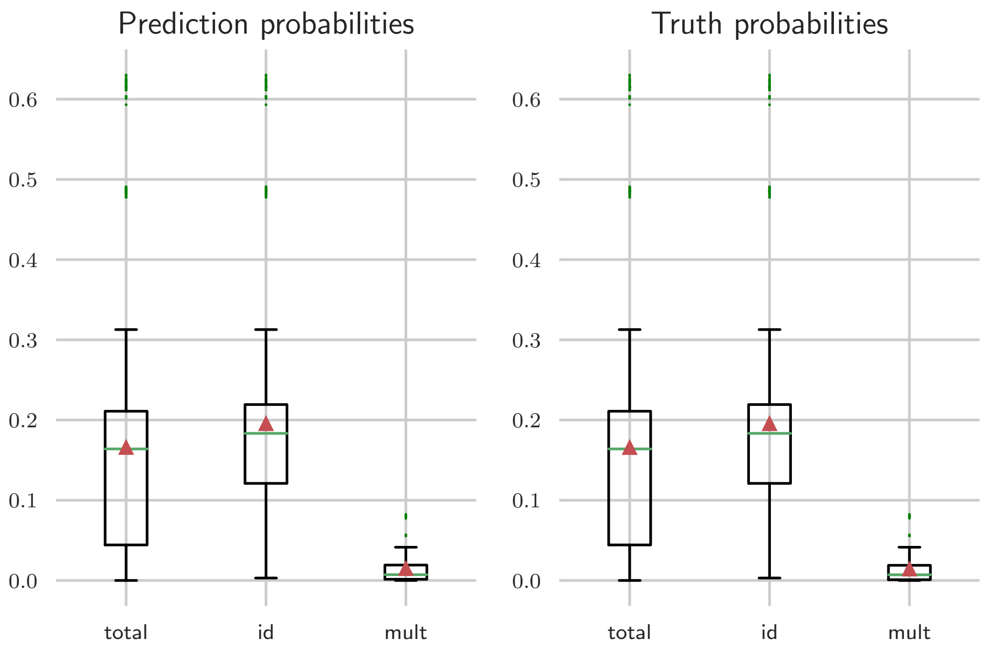


Figure 22: Probability densities of predicted and ground truth (S3A)



Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und noch nicht für andere Prüfungen eingereicht habe. Sämtliche Quellen einschließlich Internetquellen, die unverändert oder abgewandelt wiedergegeben werden, insbesondere Quellen für Texte, Grafiken, Tabellen und Bilder, sind als solche kenntlich gemacht. Mir ist bekannt, dass bei Verstößen gegen diese Grundsätze ein Verfahren wegen Täuschungsversuchs bzw. Täuschung eingeleitet wird.

Berlin, den 21/06/2021 ....