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: .....

Contents

Purpose and scope of your entire report. The purpose of your entire report is to make a **scientific argument using the scientific method**.A scientific argument always has the following steps that all must come in this order.

SM1 **Explicate the assumptions and state of the art** on which you are going to conduct your research to investigate your research problem / test the hypothesis.

SM2 Clearly and precisely **formulate a research problem or hypothesis**.

SM3 **Describe the (research) method** that you followed to investigate the problem / to test the hypothesis in a way that **allows someone else to reproduce your steps**. The method must include steps and criteria for evaluating whether you answered your question successfully or not.

SM4 **Provide execution details** on how you followed the method in the given, specific situation.

SM5 **Report your results** by describing and summarizing your measurements. You must not interpret your results.

SM6 **Now interpret your results** by contextualizing your measurements and drawing conclusion that lead to answering your research problem or defining further follow-up research problems.

1 Introduction

Purpose and scope of Section 1. The introduction is a summary of your work and your scientific argument that shall be understandable to anyone in your scientific field, e.g., anyone in Data Science. A reader must be able to comprehend the problem, method, relevant execution details, results, and their interpretation by reading the introduction and the introduction alone. Section 1.1 introduces the general topic of your research Section 1.2 discusses the state of the art and identifies a research. Section 1.3 then states the research problem to investigate. Section 1.4 explains the research method that was followed, possibly with execution details. Section 1.5 then presents the results and their interpretation. Only if a reader thinks they are not convinced or they need more details to reproduce your study, they shall have to read further. The individual chapters and sections provide the details for each of the steps in your scientific argument.

You usually write the introduction chapter *after* you wrote all other chapters, but you should keep on making notes for each of the subsections as you write the later chapters.

1.1 Context and Topic (SM1)

Process Mining is a subarea of data science and involves research to more accurately and efficiently perform the automated techniques of analyzing the formal description of processes, i.e. process models, and their corresponding executions, i.e. event logs. The

techniques make it possible to reconstruct process models from real-life process executions (process discovery), enhancing the discovered models based on quality metrics (process enhancements) and checking the conformance of a given process model, discovered or hand-made, against a the traces of a log, i.e. execution sequences of the log. [1] [2]
Process Mining as a form of data mining takes recorded instances of process executions (event logs) to derive insights in the performance and behavior of the same. Generally an event log is a collection of events (steps of a process) which belong to execution instances of that process (model) (cases). Each event is annotated with at least three attributes for clarifying which process instance an event belongs to (case id), which process step was executed in that event (activity) and at what time this event was executed (timestamp). The latter allows for a total ordering over the events within a case. [9]
"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.[9].

In the Process Mining Manifesto [26] the importance of timestamps for process mining is summarized. They can be used in *enhancement* the model can be extended to make bottlenecks, service levels, throughput times, estimated waiting time for activities and frequencies visible. Only fully ordered logs can be treated as five star maturity level. Analyze temporal behavior during replay, add expected waiting times from differences in times per event. The problem of missing or inadequate timestamps is already addressed and interpolation of temporal data is suggested. Also the problem of trace probability is addressed, stating that considering all possible traces made from the activity set is not feasible and also very unlikely. So obviously not all possibilities occur and even for some of the occurring traces it is complicated that there frequency may vary a lot. (considered "noise")[26]
Timestamp related data quality issues and the following difficulties encountered are described in:
In [1] data quality issues in general and specifically related to missing / imprecise timestamps are discussed as the timestamp, as the position, is one of the two mandatory event attributes to determine the order of the events in a trace, i.e. to establish the total order. "In most cases" this comes down to timestamps rather than positions. (*missing, imprecise or incorrect attributes*). Timestamps can give detailed insight through process mining techniques (*waiting and service time, bottleneck detection and anlysis, flow time and SLA analysis, analysis of frequencies and utilization*). Challenges with timestamps are, although not necessarily needed in order to order a trace, for example that when it comes to combining events from different sources timestamps are needed to sort events. Problem: multiple clocks and delayed recording.
[22]
[18]

Literature that describes data quality issues in general is: [1], [26], [5], [22], [17] (only read here) and timestamp specific data in [12]...
Examples of timestamp quality issues relate to their granularity, order anomaly or statistical anomaly as described in [9] are imprecise timestamps [1][17][...], event logs made from events of different systems with distinct time recording strategy [26][12] or timestamp formatting ("unanchored timestamp problem") [12][22], events being recorded after the process was finished, i.e. post mortem [1], or manual entering events or timestamps [17][16] or problems occurring due to multiple, different timezones [12]. The inaccuracies have a effect on a broad range of process mining techniques [9] this means for conformance checking that...

1.2 State of the Art (SM1)

1.3 Research Question (SM2)

Do Deep Learning techniques, LSTM, exceed precision in resolving partially ordered traces compared to the baseline (??) and the procedure recently proposed by van der Aa et. al. in [25]?

1.4 Method or Approach (SM3, SM4)

We will use state-of-the-art Deep Learning techniques to resolve the partial orderings, namely LSTMs. Therefore we incorporate two/three special encodings to generate input and output sequences that hold the "new" view on events/traces/logs.

1.5 Findings (SM5, SM6)

2 Background

2.1 Preliminaries

Process Mining is a field in the area of data science, next to other famous domains such as statistics, datamining or machine learning. [1, p.3ff] It's goal is to evaluate event data against some formal model, e.g. Petri Nets or BPMN models, describing the execution of processes. Event data is gathered manually or automatically likewise are process models. They can either be created according to given ideas and regularities describing the possible executions of an instance of the process. On the other hand there exist algorithmic techniques to discover process models from event data. With the given data and models it is then possible to evaluate certain properties of this data-model-complex, the three main types being discovery, enhancement and conformance. In the field of process mining, a subfield of (see process mining book) we deliberately investigate the happenings of processes of the "real world". It is about modeling, checking and improving such processes. Furthermore we can apply the area of conformance, checking on the area

of process mining, which is basically a more in depth investigation how well a model of a process and the actual execution of such processes are conforming. This chapter is therefore used to gently introduce typical concepts that are used in process mining and more precisely defining the attributes we need in order to answer the research question. Additionally our (main) method to solve the problem of finding resolutions for partially ordered traces will be using neural networks, i.e. a special method from the wide field of machine learning. Thus we will give a short overview over typical machine learning ideas and more concisely introduce artificial neural networks, all of their properties and capabilities as well as the different network architectures we will be have used.
Most of the plain mathematical concepts should be of knowledge to the reader, and therefore instead of defined mostly explained in text. There might be some exceptions relating to the fields of either conformance checking or neural network such as conformance checking measures or algorithms, gradient descent functions for error minimizing in ANN's or the way of encoding input data in an ANN.
Informally an *Event Log* is a data structure holding all sequentially executed *activities* as *events* of a given *process model*. Thus for any *Event Log* $\mathcal{L}$ the universe of *events*, *activities* or *cases* are denoted as $\mathcal{E}$, $\mathcal{A}$ and $\mathcal{C}$ respectively.
For a running example we will refer to the b12-Log, which we also examined for solving the proposed research questions. This log is accessible at ??.

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 we define a function $\#_n(e)$ for an $n \in \mathcal{N}$, which describes the value of attribute n for event e.

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. $\#_{Acitivity}(e) \in \mathcal{A}$, which describes the unique activity that was executed for that event.

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.

The definition of cases is needed for defining *traces* later on.
In our case, i.e. in the logs we investigated, events hold various attributes, but only the attributes *ID*, *Acitivity* and *Time* are relevant. For instance $\#_{Time}(e)$ for an $e \in \mathcal{E}$ returns the exact time at which the happening of that was record. Note that this obviously must not be the time at which the event had happened really.

The classical definition of traces in process mining is that a trace T is a set of events $e_1, e_2, ..., e_n$ all related to the same case, i.e. for all $e, e' \in T, \#Case(e) = \#_{Case}(e')$. A *trace* E can also be represented as a sequence of activities $\sigma = \langle a_1, a_2, ..., a_{|E|} \rangle \in \mathcal{A}^*$, where $a_i = \#_{Activity}(e_i)$ of the sequence $\langle e_1, e_2, ..., e_{|E|} \rangle \in E^*$, which is obtained by ordering each event by its timestamps, i.e. for all $1 \leq i \leq j \leq |E|$ it holds that $\#_{Time}(e_i) < \#_{Time}(e_j)$. Since we consider events with equal timestamps and thus such ordering would be ambiguous we will define traces as sets of events. Each event set contains events with the

same timestamp and thus we can order the log clearly.

Definition 4. (Trace) A sequence of disjoint set 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 the definition of an Event Log is slightly deviant than from the usual ones, as for example used in [1] or [6], where Logs are built from the powerset of the event universe $\mathcal{E}$

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

Based on the timestamps associated with events in a given log, we define a partial ordering of events for a certain trace. Thus an event e1 < e2, if the timestamp of e1, i.e. time(e1), is smaller than/happens before, the timestamp of e2, i.e time(e2). Otherwise the timestamps of e1 and e2 are the same, leading to e1 = e2, i.e. time(e1) = time(e2). So for any Event Set $E = \{e_1, ..., e_n\} \in \wp(\mathcal{E})$there exist n! possible resolutions of that Event Set, i.e. orderings in which way those events could have been executed in real life. This is formally captured in the following definition.

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

For speaking of traces and logs according to their traits in respect to their certainty we define the notion of certain and uncertain traces or logs respectively, which resemble the appearance of uncertain event sets in either of the two.

Definition 7. (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 8. (Certain, Uncertain Event Log) Similarly an Event Log L is called uncertain if there exists an uncertain trace $\sigma \in L$, otherwise L is called certain.

Definition 9. (Artifical Neural Network) Artifial Neural Network (ANN)

Definition 10. (feedforward-ANN) feedforward-ANN

Definition 11. (Recurrent Neural Network) Recurrent Neural Network (RNN)

In the field of deep learning it is typically needed to encode your data for feeding it into your deep learning models such that the underlying algorithms can process and calculate the given input data. In our case the data is in a text context as events are mostly described in names.

Definition 12. (Encoding An encoding is a mapping of given data to some numerical value.

We will use classic encoding strategies for our first encoding approach (One-Hot- and Embed-Encoding) and use a second approach use an advanced version of the classic one-hot-encoding.

Partial Order, Total Order

2.2 Related Work

The field of conformance checking and process mining is very broad, so a lot of research has been done there up to today. Furthermore the field of machine learning has reached another peak of high interest in business applications, as well as media, teaching and research.

The particular problem of solving partially ordered event logs from real-life process applications though, has been investigated in a fairly small amount compared to the above. Different techniques have been used to solve the problem differently and machine learning was only used once for predicting information of event logs.

To my best knowledge and also stated in previous work [16] so far there has been little research done addressing the problem of only partially ordered event logs.

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. Or it will, in finite time, find an optimal alignment for a trace and petri net, whether or not the trace is sequential, i.e. totally ordered or a trace containing concurrent events, i.e. partially ordered. (note that by definition every totally ordered trace it also a partially ordered trace).[8]

Van der Aalst et al. took another approach in defining partially ordered traces and from those compute partially ordered alignments, with the aim to provide a model that can express concurrently running events and from there getting insight in the meaning of those. [15] They researched the usefulness of those partially-ordered alignments with case studies relying on real-world data from a Dutch hospital. [16]

In [9] Dixit et. al. show a semi-automated method to repair timestamp imperfection in event logs as they...

In [23] Niek Tax et. al. introduce an approach to tackle three question not yet visited in the field of process mining. They use RNN's, LSTM's specifically, to answer three questions 1),2),3). However they let the order certainity of traces unvisited and thus, while exploring an answer for the next activity, omit the information of certain parts of a trace already being order / they only give answer on how future traces could end most possibly.

Weidlich et. al. have sought three algorithmic approaches for resolving partially ordered traces in giving a probability distribution over all possible resolutions and from there on efficiently compute the conformance of partially ordered traces with a given process

model.[25]
As of my best knowledge no research has yet been done, to resolve partial ordering of traces in conformance checking / process mining. We expect to find valuable solutions for resolving the partial ordering of events happening at the same time, exploiting the field of deep learning, neural networks respectively.

In [5] Bose et al. address the problem of imprecise event data in real-life logs, going into detail about three timestamp related problems among nine other categories, such as heterogeneous cases or the increasing amount of general data volume for logs. Real-life event logs are fine-granular, heterogeneous, voluminous, incomplete and noisy. They describe common data quality issues for event logs and state that coarse or mixed granular timestamps render process mining algorithms to deliver "wrong" control-flows, i.e. sequential activities are modeled in parallel, or even completely impossible, e.g. performance analysis. It may be good practice to look in uncertain logs for typical outliers in the appearing uncertain traces as proposed in this paper to further increase the validity of the calculated probabilities for the possible resolutions. They also show the BPI-Challenge 2012 would not yield any timestamp related issues !? Mismatch between event triggered and recorded, because it is first queued in an internal buffer or not synchronized locked. Manual logging of multiple events at the same time makes it appear the events happened in the same millisecond whereas in reality the did not. They found timestamp issues in 80% in other words 4 of the 5 examined real-life logs.
Another mention of partially ordered logs appears in [4] 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.
[2] encourages the resolution of partially ordered traces to totally ordered ones, as 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 for instance.
A. 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. Conformance Checking important for process management, process improvement and compliance. Skipping or inserting events are typical deviations. Conformance comprises of fitness, precision, generalization, structural.
In [27] van der Aalst et. al. investigate the quality metrics for alignments. Fitness: Can the Model generate the observed behavior? Precision: Avoid Underfitting. Generalization: Avoid Overfitting. Simplicity: Ocamms Razor. The three types of process mining are Discovery, Enhancement, and Conformance Checking. Process Mining sits between Computational Intelligence and data mining on the one hand and on the other between process modeling and analysis. Starting point for all process mining techniques are event logs, which are built by recording events sequentially (assumption). Then each unique event is related to a certain activity (i.e. a defined step of the process) and a particular case / trace (i.e. an execution instance of the process). For future work not only resolving the order but also resolving an acutal timestampt for the events from the rest of the log might be of interest to answer questions like: the observed average waiting time, compute

flow time, service time or synchronization times for models and log. A process model annotated with timestamps can be uses to diagnose performance problems. Furthermore learned time depending behavior can be used to make predictions: "Remaining time for this case?" or recommendations: "Which actions minimize the overall cost?". Process Models are Petri Nets, UML activity diagrams, BPMN, EPCs, YAWL etc. Basis are transition systems and all models should be interchangeable, i.e. their languages are the same. Models with timestamps for performance analysis and replaying event logs with timestamps allows for bottleneck analysis and prediction as demonstrated in [see this paper].
The lack of synchronization leading to faulty timestamps is described in [19] and states that events timestamps do not always go hand in hand with the time of their recording. A novel approach is presented to outperform existing buffering techniques, but still out-of order timestamps remain. Also in [13] the problem of following and comparing timestamps in distributed systems is addressed, as not synchronized clocks do make timestamps partially incomparable.
That manual recording of execution time for events can lead to inaccurate traces has been shown in [16].
The problem of sensing the data with real-time locating systems and generating event logs from there is addressed in [24] and [21]. The construction of discrete events from raw signals is uncertain and generated using probabilistic / data learning approaches [24]. Whereas [21] shows that going from raw sensor data to event logs needs a fair amount of process knowledge and the derived logs still have quality issues. .

Machine Learning is a wide field of methods, also emerging from data sciences, trying to learn some behavior. There is wide range of tasks to learn such as regression, clustering, dimensionality reduction and classification. Among an equally wide spectrum of methods to incorporate these tasks artificial neural networks (ANN) are popular among amateurs and scientists as they provide an intuitive model of the underlying (biological) concept as well as delivering state of the art results in many scientific research programs, e.g. classification of handwritten digits or pictures of clothing.
Generally there are two types of neural networks, feedforward and recurrent ANNs. The latter inherit a hidden state that evolves over time and thus can handle sequential data especially well. The initial recurrent neural networks however have turned out to suffer from the *vanishing gradient* problem. This means that over time newer inputs of a given input sequence will have a larger effect on the hidden state than older inputs, for instance the very first one. Over a sufficient large sequence of inputs the respective gradients of very early inputs would basically become 0. The *long short-term memory* ANN (LSTM) are model originally introduced in ???? which control information flow of memory more accurately / flexible by using a more sophisticated hidden unit.

3 Problem Exposition (optional)

3.1 Context / Business Understanding (SM1)

As stated in the the Introduction 1 when logs are acquired in real life not all data will always be completely clean. For instance roles performing a certain activity might be missing or trace, i.e. instances of a process, might not have been recorded. Furthermore there is are multiple reasons for timestamps in the log to be vague. This can be due to the lack of synchronization, manual recording or data sensing. [25] For conformance checking techniques to be used on the even data to be executed properly and thus give meaningful insights into the correlation of model (Petri Net, ...) and data (Event Logs) there are different approaches one could undertake. For instance one could just drop out the uncertain traces before applying state of the art conformance checking techniques, which especially makes sense when there is only a small fraction of traces that are uncertain for the event log.
However we would rather try to resolve this uncertainty to get the most realistic image of what has happened when the activities in the logs had been executed.

3.2 Data Understanding (SM1)

To understand the data we are working with better we examined x Event Logs available from the open research website from the 4TU website. (link is now down unfortunately). Event Logs that are generated in real world processes vary largely in a lot of different aspects. For example for the x logs we examined a priori, to choose the right ones for evaluating the research question upon, the number of traces range widely from about 1000 to over 150000 per Event Log and furthermore the size of the Activity Universe $\mathcal{A}$ can be quite small with 9 activities or relatively large with up to 51 activities executable. See Table 1 for a concise summary.
The uncertainty of the logs, i.e. the fraction of traces, which are uncertain, go from 6% up to almost the whole log being uncertain with 96%. The number of uncertain sets we observed appearing in the given logs go from 15 up to 39. The basic statistical key figures however are fairly evenly distributed over all the logs. For simplicity we denote the set of uncertain sequences of a log as *unc_seq*, i.e. it contains all the sets of activities for which had the same timestamp in at least one trace of the log. Here so far we only consider the longest sequence, e.g. for a trace $\langle A, B, C, D, E\rangle$ in which B, C, D are uncertain we would only add $\{B, C, D\}$ to *unc_seq* and not $\{B, C\}$ and $\{C, D\}$ although one could argue they appear as uncertain in the log as well. However it is hard to provide evidence that the shorter sequences would appear uncertain "alone" as well and for example B and C would also be uncertain if D had a different timestamp than the two for that trace. So the average length of the uncertain sequences is between 2 and 3 and the median of the maximum length of the uncertain Sequences in each log is 4.

Example of a table

Log:	BPI 21	BPI 14	Traffic Fines
$\|\mathcal{A}\|$	24	9	11
# Traces	13087	41353	150370
# Events	262200	369485	561470
Trace Uncertainty	38%	93%	6%
Event Uncertainty	5%	40%	2%
$\|unc\_seq\|$	14	24	25
longest unc_seq	4	4	3
avg length of unc_seq	2.4	2.6	2.0

Table 1: Summary of empirically obtained data for the different logs

3.3 Detailed Research Questions (SM2)

The research question in detail tries to answer whether a dynamic deep learning neural network architecture can yield more precise and consistent results for partial order resolution in event logs than the statical algorithms approached in [25].
Therefore we need to ensure that we can define a model that suits our needs the best and the training data we will use for evaluating the deep learning model. The final goal of the research will be to acquire a model that we can feed any uncertain event set, for certain event set the problem is trivial, and it will present us the most probable resolution for that uncertain event set.

3.4 Detailed Method (SM3)

The method is to take real world event logs form open research repositories (link is down..) and process them as follows.
We extract all important data from the logs, mainly the traces and events. Additionally for each event we keep the needed attributes, which are the attendant Case-ID, Event-ID, timestamps and therefore their ordering. Also for comparism with the prior approach shown in [25] we implemented the three given algorithms in python and evaluated the results against what we could find with the deep learning approach of this thesis.

4 First Real Chapter addressing first Research Problem

4.1 Encoding the Event Data

Finding good encodings for events, is difficult, since all possible resolutions for an uncertain set leads to too many possible resolutions, i.e. the input or output vectors would get too large, when using *one-hot-encoding*. With *embeddings* the length of each input would be shorter or the chosen embedding dimension more precisely. But still the set of all possible resolution would be needed to calculate the embeddings for any occuring event set(maybe change name in definition) of the input log.

To address this problem only the occuring resoltions for any event set have been taken into account. The reduction of vector size for the five logs examined is represented in TABLE.
For any uncertain event set $e = e_1, e_2, e_3, ..., e_n$of length n, the size of the input vector results of following formula: FORMULA. The first part is the size of the activity universe, since all possible singular event sets must be considered.
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 as $\sum_{l=1}^{k}(|\mathcal{A}| + l - 1)!/(l! \cdot (|\mathcal{A}| - 1)!)$.
On the other hand the output vector size, i.e. all possible sequences from length 1 to k can be simply computed as $\sum_{l=1}^{k} |\mathcal{A}|^l$, and naturally is an upper bound to the input vector size.
This clearly increases exponentially by increasing the size of the activity universe or the length of the traces.
Encoding all possibilities vs encoding only the occurring possibilities
If we were to encode all possible up-to-k subsets for a given event set the number of encodings, i.e. the number of possibly appearing events is computed as follows. Thereby the number k is set to four at the moment, as the inspection of our three data sets / logs have shown the longest uncertain sequence is actually four.
$\#encodings(k = 4) = \sum_{j=1}^{k} |\mathcal{E}|^j$
For the three observed logs with a size of the event universe of 24 (BPI12), 9 (BPI14) and 11 (Traffic) respectively this would lead to a total number of 346200 (BPI12), 7380 (BPI14) and 16104 (Traffic) of possible events to encode.
However not all of these combinations of events do appear in the log. Actually we could observe that only 14 (BPI12), 24 (BPI14) and 25 (Traffic) of all possible event sets where present in the corresponding logs. (not counting the trivial event sets?)
Summing up the possible resolutions for each of these event sets as of: let m be the length of an uncertain set, then there are m! possible resolutions for that set. E.g. for the uncertain set {a,b,c,} of length 3 there exist 3! = 6 possible resolutions, {a,b,c}, {a,c,b}, {b,a,c}, {b,c,a}, {c,a,b}, {c,b,a}. Counting up from there we have 86 (BPI12), 134 (BPI14) and 63 (Traffic) encodings for the output vectors. Leveraging this idea would on the one hand ease encoding in making the feature vectors a lot smaller. In the best case scenario for the BPI12 log this would mean a reduction of feature vector length by roughly the factor of 25000.
Real-life Event Logs however can have a much bigger activity spaces, e.g. the log of the BPI-Challenge 2015 which contains almost 400 possible activities to be executed.
Furthermore one could reduce the size of the input vectors only(!), since ordering for any event set here does not matter / does not exist. So for instance the uncertain event set {a,b,c} and {c,a,b} resemble the same input structure and therefore could be treated as the same input, namely {a,b,c} in this case.
Additionally we only considered a k of 4. One however can easily see that for larger k this also explodes exponentially.
That is why embeddings where used for encoding, as literature suggests using this type of encoding over one-hot-encoding for vectors that would contain more than 50 different features.

Finally we consider two relevant encodings.

4.2 What is the training data

4.3 Trying out a feedforward-ANN

When considering the research problem, curiosity made the way for also trying out the learning behaviour of a feedforward-ANN. The hypothesis was that the feedforward-ANN would learn the current fraction of already correct ordered uncertain traces, when assuming the order in the log was the correct order for any given trace. By that way for a uncertain event say $\{a, b, c\}$ the net would possibly learn which ordering, $\{a, b, c\}$, $\{a, c, b\}$,$\{b, a, c\}$, $\{b, c, a\}$, $\{ca, b\}$, $\{c, b, a\}$ appears most frequently in the given log, and assign the each of the possible resolutions their following appearance probability.
´

5 Acquiring Data

The data was acquired from these links referring to standard event logs for research, annual challenges etc. in the field of process mining.

5.1 Abstracting timestamps

For resolving the partially ordered traces in the proposed manner it was sufficient to abstract from timestamps such that only the order remained but no exact timestamps, e.g. for events e_1, e_2 with timestamps t_1, t_2 we would keep the information that e_2 occurred before e_1 and omit the information about how much time exactly lies between those events.

5.2 Fraction of uncertain traces

5.3 Training and Test Data

5.4 Finding out the solutions and making predicitons

6 Evaluation

6.1 Objective (SM2)

Regarding the initial hypothesis that deep learning, artificial neural networks especially can resolve partial orders should be assessed here. Therefore we compare the neural network approach with the previous attempt [25] (and the ground truth). Following from that the precision of the provided resolution by each attempt are compared. Firstly we took all the uncertain traces from the experimental data (Sepsis, Permit, BPI 12, BPI 14, Traffic Fine Log) and extract from them all the uncertain sequences. We then let every

model predict the possible resolution for each of those. Note that the output layer of the artificial neural network is pretty large as is the probability space for the stochastic attempt. So as a first attempt we implement a 0/1-loss over all answers provided by the two models. This means we abstract from the models producing a probability distribution over the classes and rather transform the class prediction vector such that it contains a 1 where it had the highest probability value and 0 everywhere else. Than the total precision of the model is computed as the sum of the number of correct predictions normalized by the number of all predictions.

6.2 Setup (SM3)

Since, as described before, Event Logs can in practical use can take very different shapes, e.g. in terms of average trace length, size of the activity set, the average size of uncertain sequences and probably most importantly the level of uncertainty found in the log, it is important to compare both models individually on each event. Then from there on one could make deduct further insights.

6.3 Execution (SM4)

For LSTM with encoding 1 (set and sequences) epoches used: 100 for bpi14 log, 30 used for traffic fines log, the process for the bpi12 log was killed (probably due to high ram usage).
For accuracy we used the *catergorial_accuracy*, specified when compiling a model in keras, which creates *count* and *total* as two local variables and thereby computes the ratio of correct to total predictions.
Also we defined our own accuracy metrics, as *categorical_accuracy* did not seem to yield results matching the prediction accuracy of the trained models. We therefore manually introduced metrics that would make predictions for the hold out test set and compare the results to the actual / ground truth (what should be predicted) contained in the test set targets.

6.4 Results (SM5)

For the Sepsis Log [see table 1] the results are as expected. Since the log is rather small and only contains a small amount of certain traces the deep learning approach could only

For the BPI-12 Log training with k=4 lead converging at around 0.68 accuracy as fast as shortly before 10 epochs, k=3 also converged fast, after the third epoch with a training accuracy of about 0.82.

Log:	BPI 12	BPI 14	Traffic Fines
k = 2	x.xx, x.xx	x.xx, x.xx	x.xx, x.xx
k = 3	x.xx, x.xx	x.xx, x.xx	x.xx, x.xx
k = 4	x.xx, x.xx	x.xx, x.xx	x.xx, x.xx

Table 2: training accuracy for different values of k with ANN approach in each cell it is training accuracy, test accuracy

6.5 Discussion (SM6)

7 Conclusion

Your conclusions are not just a factual summary of your work, but they position, interpret and defend your findings against the state of the art that you discussed in Sect. 1.2. You specifically outline which concrete findings or methodological contributions advance our knowledge towards the general objective you introduced in Sect. 1.1. Objectively discuss which parts you solved and in which parts you failed.
You should explicitly discuss limitations and shortcomings of your work and detail what kind of future studies are needed to overcome these limitations. Be specific in the sense that your arguments for future work should be based on concrete findings and insights you obtained in your report.

Maybe state that different models, regarding to parameters, complexity, algorithmic details, need to be considered, for example pruning or application of Occam's Razors could have positive effects on computation time or error-rate. [10] Furthermore recently the decision making of machine learning models has been questioned as for instance in image classification horse images have been classified also by an text tag in the bottom left rather than the picture itself. [14] Thus further investigation of the learning scheme of models which resolve partial orders for event logs should be considered.

Some different encoding of the activity space could yield different results, as for example using word embeddings, such as *Word2Vec* or *GloVe*, in order to create a meaningful geometric space of related activities. Embeddings have been introduced to process mining successfully in [7] with vector representations learned for activities (act2vec) and traces(trace2vec) among others, similar to *Word2Vec*, and further already have been applied to a new conformance checking approach with promising initial results in [20].
Other more sophisticated approaches, that emerged in the recent years, could improve the prediction accuracy as well. Namely *Bidirectional RNNs*, *Beam Search*, *Attention Mechanisms*, here especially the transformer architectures. [11]

For the examined logs the average length of uncertain sequences never exceeds the value of 3, as table 1 shows, which encourages the assumption to invent strategies that omit the possibly few longer uncertain sequences in order to gain advantages in terms of

execution time and space complexity.

8 brainstorming

Table 3: Used models and specifications

model	log	encoding	ordering	Done ?	Results			
					train	val	test	accuracy
LSTM	BPIC12	basic	log order	✓	2.28e-08	1.41e-08	1.30e-07	1.00
LSTM	BPIC14	basic	log order	✓	6.36e-09	6.45e-09	6.34e-09	1.00
LSTM	traffic	basic	log order	✓	2.24e-08	2.24e-08	2.24e-08	1.00
LSTM	BPIC12	...	...	...	x.xx	x.xx	x.xx	
LSTM	BPIC14	sets	...	...	0.024	0.020	0.024	
LSTM	traffic	sets	...	...	x.xx	x.xx	x.xx	
seq2seq	BPIC12	basic	log order	✓	0.032	0.064	-	0.45
seq2seq	BPIC14	basic	log order	✓	0.062	0.50	-	0.66
seq2seq	traffic	basic	log order	✓	8.66e-04	0.002	-	1.00 (0.998)
seq2seq	BPIC12	...	...	...	x.xx	x.xx	-	
seq2seq	BPIC14	sets	log order	...	0.073	0.255	-	
seq2seq	traffic	sets	log order	...	0.098	0.108	-	

The LSTM, basic was done with 0.2 test size, 0.1 validation size and 100 epochs, one hidden lstm layer
The Seq2seq basic was done with 0.2 test size and 50 epochs, "standard" encoder-decoder architecture of 2 LSTMs in total

The LSTM, basic was done with 0.2 test size, 0.1 validation size and 30 epochs, one hidden lstm layer
Options for large vocabulary:

- ◇ mark rare tokens / event sets with "RARE"
- ◇ https://stackoverflow.com/questions/51057123/keras-one-hot-encoding-memory-management-best-possible-way-out
- ◇ https://stackoverflow.com/questions/41002722/how-to-handle-very-sparse-vectors-in-tensorflow

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] 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. 2011.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] Pedro Domingos. Occam's two razors: the sharp and the blunt. In *KDD*, pages 37–43, 1998.

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

[12] 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.

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

[14] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, 10(1):1–8, 2019.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] Han van der Aa, Henrik Leopold, and Matthias Weidlich. Partial order resolution of event logs for process conformance checking. 2019.

[26] 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.

[27] Wil Van der Aalst, Arya Adriansyah, and Boudewijn van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.

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 22/03/2021 .....