Department of Mathematics and Computer Science
Process Analytics

The Generation of interpretable counterfactual examples by finding minimal edit sequences using event data in complex processes

Master Thesis

Olusanmi A. Hundogan

Supervisors:
Xixi Lu
Yupei Du
March 8, 2022

Contents

Chapter 1

Introduction

1.1 Motivation

Many processes, often medical, economical, or administrative in nature, are governed by sequential events and their contextual environment. Many of these events and their order of appearance play a crucial part in the determination of every possible outcome. With the rise of AI and the increased abundance of data in recent years several techqniques emerged that help to predict the outcomes of complex processes in the real world.**[Expand the domain application.]**

For instance, research in the Process Mining discipline has shown that is possible to predict the outcome of a particular process fairly well CITE . **[However, while many prediction models can easily certain outcomes, it remains a difficult challenge to understand what led to a particular outcome. This obstacle is undesirable, as knowing the main factors to an outcome can help understand how to steer a process to a desired outcome with minimal effort.]** In other words, we want to change the outcome of a particular event, by making it maximally likely, with as little interventions as possible CITE TEST .

One-way to better understand the Machine Learning (ML) models lies within the eXplanable AI (XAI) discipline. XAI dedicates its research to the **research and** development of so-called *black-box models* that are difficult to interpret. Most of the discipline's techniques produce explanations that guide our understanding.

A prominent and human-friendly approach uses the generation of counterfactuals as primary explanation tool. Counterfactuals within the AI framework help us to answer hypothetical "what-if" questions. In this thesis, we will raise the question, how we can use counterfactuals to change the trajec-

tory of a models' prediction towards a desired outcome. Knowing the answers will help us further understand what to do to avoid or enforce the outcome of a process.[WHY]

1.2 Problem Space

In this paper, we will approach the problem of generating counterfactuals for processes. The literature has provided a multitude of techniques to generate counterfactuals for AI models, that are derived from static data[1]. However, little research has focussed on counterfactuals for dynamic data[2]. A major reason, emerges from a **[multitude – better #]** of challenges, when dealing with counterfactuals and sequences. First, counterfactuals within AI attempt to explain outcomes, that did not happen. Therefore, there is no evidence data, from which one can infer predictions. Subsequently, this lack of evidence further complicates the evaluation of generated counterfactuals. In other words, you cannot validate the correctness of a theoretical outcome that has never occured. Second, sequential data is not only has a highly variable form, too **CITE** . The sequential nature of the data impedes the tractability of many problems due to the combinatorial explosion of possible sequences which depends on the length of the sequence. Third, process data of requires knowledge of the underlying and often hidden causal structures that produce the data in the first place. However, these structures are often hidden and it is a NP-hard problem to elicit them **CITE Check process discovery literature** . Furthermore, the data generated is seldomly one-dimensional or discrete. Henceforth, each dimension's contribution can vary in dependance of its context, the time and magnitude. Hence, the field in which we can contribute to this open challenge is vast.

1.3 Approach

Due to the challenges imposed by process data, we have to restrict the solution space by imposing limitations and assumptions. We will explore these restrictions while describing the most important concepts in chapter 2. To solve the problems and explore a number of models this paper. We will first train a predictive model and compare two counterfactual generation methods. We evaluate the generated counterfactuals based on their *validity*. The

[1]With static data, we refer to data that does not change over a time dimension.

[2]With dynamic data, we refer to data that has time as a major component, which is also inherently sequential

validity of a counterfactual is explained in section 2.3. For the generation process, we will focus on exploring an evolutionary computing approach for its ability to optimise non-differential loss functions and a deep generative model as it can sample instances from similar sequences from its data distribution.

Chapter 2

Background

This chapter will explore the most important concepts for this work. Most of the concepts can have several meanings depending on the varying context in which they are applied. For this purpose, we will provide an intuitive understanding, the ensuing challenges, a concrete definition for this work and lastly and a mathematically formal description. The concepts we will cover encompase **sequence modelling**, process mining and counterfactual explanations.

2.1 Process Mining

2.1.1 A definition for Business Processes

Before elaborating on Process Mining, we have to establsih the meaning of the term *process* in the context of this paper. The term is broadly used in many contexts and therefore has a rich semantic volume. A process generally refers to something that advances and changes over time[**˙DefinitionPROCESS˙**]. Although, legal or biological processes may be valid understandings, we focus on processes *business processes*.

An example is a loan application process in which an applicant may request a loan at a specific point in time. The case is then assessed and reviewed by multiple approvers and ends in a final decision. The loan may be granted or denied. The *business* part may be misleading as these processes are not confined to commercial settings. For instance, a medical business process may cover a patients admission to a hospital, followed by a series of diagnostics and treatments and ending with the recovery or death of a patient. Another example from a human-computer-interaction**[Add to glossary]** perspective would be an order process for an online retail service like Amazon. The buyer

might start the process by adding articles to the shopping cart and proceeding with specifying their bank account details. This order process would end with the submission or receival of the order.

All of these examples have a number of common characteristics. They have a clear starting point which is followed by numerous intermediary steps and end in one of the possible sets of outcomes.For this paper we will refer to each step, including start and end points, as Process Event. Each Process Event may contain additional information in the form of event attributes. A collection of these Process Events refer to a Process Instance, if they all relate to a single run of a process. In line with the aforementioned examples, these Process Instances could be understood as a single loan application, a medical case or a buy order. We can also attach Process Instance related information to each instance. Examples would be the applicants race, a patients age or the buyers budget. In its entirety, a business process can be summarised as a *graph* or *flowchart*, in which every node represents an event and each arc the path to another event. This graphical representation is referred to as *process map*. Figure 2.1 shows an example of such a representation.

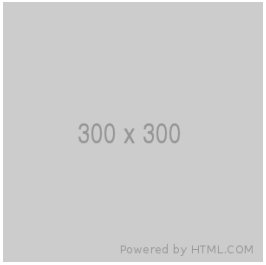

Figure 2.1: This graph shows an example of various process maps.

In conclusion, in this thesis a *business process* refers to

> *A finite series of discrete events with one or more starting points, intermediary steps and end points.*

However, we have to address a number of issues with this definition. First, this definition excludes infinite processes like **[XXX]** or continuous processes such as **[XXX]**. There may be valid arguments to include processes with these characteristics, but they are not relevant for this thesis. Second, in each example we deliberately used words that accentuate modality such as *may*, *can* or *would*. It is important to understand that each process anchors its definition in its application context. Hence, what defines a business process is indisputably subjective. For instance, while an online marketplace like Amazon might be interested in the process from the customers first click

to the successful shipment, an Amazon vendor might be interested in the delivery process of a product only. Third, the example provided in Figure 2.1 may not relate to the reality of a data generating process. In line with the second point, these examples subjective models of a process. They may or may not be accurate. The *true* process is often unknown to every actor. Therefore, we will distinguish between the *true process model* and a *process model*. The *true process model* is a hypothetical concept whose *true* structure remains unknown.

2.1.2 What is Process Mining

Having established our understanding of a process, we can turn towards *Process Mining*. This young discipline has many connections to other fields that focus on the modeling and analysis of processes such as Continuous Process Improvement (CPI) or Business Process Management (BPM). However, its data-centric approaches originate in Data Mining. The authors **vanderaalst˙ProcessMiningManifesto˙2012** describe this field as a discipline "to discover, monitor and improve real processes (i.e., not assumed processes) by extracting knowledge from event logs readily available in today's (information) systems"[**vanderaalst˙ProcessMiningManifesto˙2012**]. The discipline revolves around the analysis of Event Logs. An Event Log is a collection of Process Instances. These logs are retrievable from various sources like an Information Systems (ISs) or database. Those logs are often stored in data formats such as Comma Seperated Values (CSV) or eXtensible Event Stream (XES).

2.1.3 The Challenges of Process Mining

As mentioned in chapter 1, process data modelling and analysis is a challenging task. **vanderaalst˙ProcessMiningManifesto˙2012** mentions a number of issues that arise from processes[**vanderaalst˙ProcessMiningManifesto˙2012**].

The first issue arises from the quality of the data set. Process logs are seldomly collected with the primary goal of mining information and hence, often appear to be of subpar quality. The information is often in complete due to a lack of context information, the ommision of logged process steps or wrong levels of granularity.

This issue is exacerbated by the second major issue with process data. Mainly, its complexity. Not only does a process logs complexity arise from the variety of data sources and differing levels of complexity, but also from the datas' characteristics. The data can often be viewed as multivariate sequence with discrete and continuous features and variable length. This

characteristic alone creates problems explored in section 2.2. **[Also refer to variability in sequence section.]** However, the data is also just a *sample* of the process. Hence, it may not reflect the real process in its entirety. In fact, mining techniques need to incorporate the *open world assumption* as the original process may generate unseen Process Instances.

A third issue which contributes to the datasets incompleteness and complexity is a phenomenon called *concept drift*. This phenomenon relates possibility of a change in the *true* process. The change may occur suddenly or gradually and can appear in isolation or periodically. An expression of such a drift may be a sudden inclusion of a new process step or domain changes of certain features. These changes are not uncommon and their likelihood increases with the temporal coverage and level of granularity of the dataset CITE . In other words, the more *time* the dataset covers and the higher its detail, the more likely a change might have occured over the time.

All three issues relate to the *representativeness* of the data with regards to the unknown *true* process that generated the data. However, they also represent open challenges that require research on their own. For our purpose, we have to assume that the data is representative and its underlying process is static. These assumptions are widely applied in the body of process mining literature CITE .

2.2 Multivariate Time-Series Modelling

The data which is mined in Process Mining is typically a multivariate time-series. It is important to establish the characteristics of time-series.

2.2.1 What are Time Series Models?

A time series can be understood as a series of observable values, that depend on previous values. The causal dependence turns time-series into a special case of sequence models. Sequences do not *have to* depend on previous values. They might depend on previous and future values or not be interdependent at all. An example of a sequence model would be a language model. Results in Natural Language Processing (NLP), that the words in a sentences for many languages do not seem to only depend on prior words but also on future words CITE . Hence, we can assume that a human has formulated his sentence in the brain before expressing it in a sequence of words CITE . In contrast to sequences, time series cannot depend on future values. The general understanding of *time* is linear and forward directed CITE . The notion of time relates to our understanding of *cause and effect*. Hence, we can de-

compose any time series in a precedent (causal) and an antecedent (effect) part (CHECK [**leglaive˙RecurrentVariationalAutoencoder˙2020**]). A time series model attempts to capture the relationship between precedent and antecedent.

2.2.2 The Challenges of Time Series Modelling

The analysis of unrestricted sequential opens up a myriad of challenges. First, sequential data introduce a combinatorial set of possible realisations. For instance, a set of two objects $\{A, B\}$ yields 7 theoretical combinations ($\{\emptyset\}$, $\{A\}$, $\{B\}$, $\{A, B\}$, $\{B, A\}$, $\{A, A\}$, $\{B, B\}$). Just by adding C and D to the object set increases the number of combinations to 40 and then 341. Second, sequential data may contain cycles which increases the number of possible productions to infinity **CITE** . Both, the combinatorial increase and cycles, contain a set of a countable infinite number of possbilities for discrete sets. However, as processes may also contain additional information a third obstacle arises. Including additional information increases the set to an uncountable number of possible values. With these obstacles in mind, it often becomes intractable to compute an exact model.

Hence, we have to include restrictive assumptions to reduce the solution space to a tractable number. A common way to counter this combinatorial explosion is the inclusion of the *Granger Causality* assumption. This idea postulates the predictive capability of a sequence given its preceding sequence. In other words, if we know that D can only follow after C, then 341 possible combinations reduce to 170. All of these possible 170 combinations are now temporally-related and hence, we speak of a *time-series*.

However, the prediction of sequences raises two new questions. First, if we know the precedence of a time-series, what is the antecedent? And second, if we can predict the antecedent accurately, what caused it? At first glance, it is easy to believe that both questions are quite similar, because we could assume that the precedent causes the antecedent. However, the first question is often solved using predictive AI models that rely on data, like Hidden-Markov-Models or Deep Learning. However, the latter question is much more difficult as data cannot help explain causes of sequences that never occured. To illustrate this impossibility, if we never encountered 'D' in our dataset consisting of A, B and C, we cannot reliably say what would cause D. Answering this question requires additional tools within the XAI framework. One such method is the focus of this thesis and is further explored in section 2.3.

2.3 Counterfactuals

Counterfactuals are an important explanatory tool to understand a models' cause for decisions. Generating counter factuals is main focus of this thesis. Hence, we will establish the most important chateristics of counterfactuals in this section.

2.3.1 What are Counterfactuals?

Counterfactuals have various definitions. However, their semantic meaning refers to "a conditional whose antecedent is false"[**˙Counterfactual˙**]. A simpler definition from **starr˙Counterfactuals˙2021** states, counterfactual modality concerns itself with *what is not, but could or would have been.* Both definitions are related to linguistics and philosophy. Within AI and the mathematical framework various formal definitions can be found within causal inference[**hitchcock˙CausalModels˙2020**]. Here, citeauthor describes a counterfactual as Causal inference definition. What binds all of these definitions is the notion of causality within "what if" scenarios.

However, for this paper, we will use the understanding established within the XAI context. Within XAI, counterfactuals act as a prediction which "describes the smallest change to the feature values that changes the prediction to a predefined output"[**molnar2019**]. Note that XAI mainly concerns the explanation of models, which are always subject to inductive biases of the model itself and therefore inherently subjective. The idea behind counterfactuals as explanations[1] is that we understand the output of a model, if we know what change caused would cause a different outcome. For instance, lets denote a sequence 1 as $ABCDE\boldsymbol{FG}$, then a counterfactual $ABCDE\boldsymbol{XZ}$ would tell us that **F** (probably) caused **G** in sequence 1. As counterfactuals only address explanations of single model instances and not the model as a whole, they are called *local* explanation.

Valid counterfactuals satisfy four criteria. First, a counterfactual should be minimally different from the true instance. If the counterfactual to sequence 1 was $A\boldsymbol{A}CDE\boldsymbol{XZ}$ we would already have difficulties to discern whether B or F or both caused G at the end of sequence 1. Second, a counterfactual should produce a predefined outcome as closely as possible. This characteristic is ingrained in **molnar2019**s definition. If the counterfactual $ABCDE\boldsymbol{XZ}$ ends with Z but this sequence is highly unrealistic, then cannot be certain of our conclusion for sequence 1. Third, we typically desire multiple diverse counterfactuals. One counterfactual might not be enough to

[1]There are other explanatory techniques in XAI like *feature importances* but counterfactuals are considered the most human-understandable

understand the causal relationships in a sequence. In the example above we might have a clue that F causes G but what if G is not only caused by F? If we are able to find counterfactuals ***V****BCDEF****H*** and *ABCDE****XZ*** but all other configurations lead to G, then we know positions 1 and 6 cause G. As last criterion, each counterfactual should be possible. A sequence *ABCDE****1G*** would not be possible if numericals are not allowed. All four criteria allow us to assess the validity of each generated counterfactual and thus, help us to define an evaluation metric.

2.3.2 The Challenges of Counterfactual Sequence Generation

The current literature surounding counterfactuals expose a number of challenges when dealing with counterfactuals.

The most important disadvantage of counterfactuals is the Rashommon Effect[**molnar2019**]. If all of the counterfactuals are valid, but contradict eachother, we have to decide which of the *truths* are worth considering.

This decision reveals the next challenge of evaluation CITE . Although, the criteria can support us with the decision, it remains a question *how* to evaluate counterfactuals. Every automated measure comes with implicit assumptions and often do not guarantee a realistic explanation. We still need domain experts to assess their validity.

The generation of counterfactual sequences contribute to both former challenges, due to the combinatorial expansion of the solution space. This problem is common for counterfactual sentence generation and has been adressed within the NLP CITE . However, as process mining data not only consist of discrete objects like *words*, but also event and case features, the problem remains a daunting task. So far, little work has gone into the generation of multivariate counterfactual sequences like Process Instances CITE .

2.4 Related Literature

2.4.1 Generating Counterfactuals

The topic of counterfactual generation as explanation method was introduced by **wachter˙CounterfactualExplanationsOpening˙2017** in **wachter˙CounterfactualExpla**
The authors defined a loss function which incorporates the criteria to generate a counterfactual which maximizes the likelihood for a predefined outcome and minimizes the distance to the original instance. However, the solution of **wachter˙CounterfactualExplanationsOpening˙2017** did not account

for the minimalisation of feature changes and does not penalize unrealistic features. Furthermore, their solution cannot incorporate categorical variables.

A newer approach by **dandl˙MultiObjectiveCounterfactualExplanations˙2020** incoporates all four main criteria for counterfactuals (see section 2.3) by applying a genetic algorithm with a fitness function that incorporates all of the main criteria CITE . This approach strongly differs from gradient-based methods, as it does not require a differentiable function which requires optimization. However, their solution only works with structured data.

2.4.2 Generating Counterfactual Sequential

When it comes to sequential data most researchers work on ways to generate counterfactuals for natural language. This often entails generating univariate discrete counterfactuals with the use of Deep Learning techniques. **martens˙ExplainingDatadrivenDocument˙2014** and later **krause˙InteractingPredictions** are early examples of counterfactual NLP research. Their approach strongly focuses on the manipulation of sentences to achieve the desired outcome. However, as **robeer˙GeneratingRealisticNatural˙2021** puts it, their counterfactuals do not comply with *realisticness*.

Instead, **robeer˙GeneratingRealisticNatural˙2021** showed that it is possible to generate realistic counterfactuals with a Generative Adversarial Model (GAN). They use the model to implicitly capture a latent state space and sample counterfactuals from it. Apart from implicitly modelling the latent space with GANs, it is possible to sample data from an explicit latent space. Examples of these approaches often use an encoder-decoder pattern in which the encoder encodes a data instance into a latent vector, which will be peturbed and then decoded into a a similar instance[**melnyk˙ImprovedNeuralText˙2017**][**wang** By modelling the latent space, we can simply sample from a distribution conditioned on the original instance. **bond-taylor˙DeepGenerativeModelling˙2021** provides an overview of the strengths and weaknesses of common generative models.

Eventhough, a latent vector model can theoretically produce multivariate sequences, it a single latent-vector may be too restrictive to capture the combinatorial space of multivariate sequences. Hence, most of the models within NLP were not used to produce a sequence of vectors, but a sequence of discrete symbols. For process instances, we can assume a causal relation between state vectors in a sequential latent space. We call models that capture a sequential latent state-space which has causal relations *dynamic*[**leglaive˙RecurrentVariationalAutoencoder˙2020**]. Early models of this type of dynamic latent state-space models are the well-known

Kalman-Filter for continous states and Hidden Markov Model (HMM) for discrete states. In recent literature, many techniques use Deep Learning to model complex state-spaces. The first models of this type were developed by **krishnan˙StructuredInferenceNetworks˙2017**. Their Deep Kalman Filter (DKF) and subsequent Deep Markov Model (DMM) approximate the dynamic latent state-space by modeling the latent space given the data sequence and all previous latent vectors in the sequence. There are many variations of **krishnan˙StructuredInferenceNetworks˙2017**'s model, but most use Evidence Lower-Bound (ELBO) of the posterior for the current Z_t given all previous $\{Z_{t-1}, \ldots, Z_1\}$ and X_t.

2.4.3 Generating Counterfactual Time-Series

Within the *multivariate time-series* literature two recent approaches yield ideas worth discussing.

First, **delaney˙InstanceBasedCounterfactualExplanations˙2021** introduces a case-based reasoning to generate counterfactuals. Their method uses existing counterfactual instances, or *prototypes*, in the dataset. Therefore, it ensures, that the proposed counterfactuals are *realistic*. However, case-based approaches strongly depend on the *representativeness* of the prototypes CITE . In other words, if the model displays behaviour, which is not capture within the set of prototypical instances, most case-based techniques will fail to provide valid counterfactuals. The likelihood of such a break-down increases due to the combinatorial explosion of possible behaviours if the *true* process model has cycles or continuous event attributes. Cycles may cause infinite possible sequences and continous attributes can take values on a domain within infinite negative and positive bounds. These issues have not been explored in the paper of **delaney˙InstanceBasedCounterfactualExplanations˙2021**, as it mainly deals with time series classification. However, despite these shortcomings, case-based approaches may act as a valuable baseline against other sophisticated approaches.

The second paper within the multivariate time series field by **ates˙CounterfactualExplanati** also uses a case-based approach. However, it contrasts from other approaches, as it does not specify a particular model but proposes a general framework instead. Hence, within this framework, individual components could be substituted by better performing components. Describing a framework, rather than specifying a particular model, allows to adapt the framework, due to the heterogeneous process dataset landscape. In this paper, we will also introduce a framework that allows for flexibility depending on the dataset. **The framework will be evaluated in two steps. The first step aims to compare various model types against eachother based on the coun-**

tefactual validiy. The second step scrutinizes the best framework configurations from step one, by presenting its results to a domain expert.

2.4.4 Generating Counterfactuals for Business Processes

So far, none of the models have been applied to process data.

Within Process Mining (PM), Causal Inference has long been used to analyse and model business processes. Mainly, due to the causal relationships underlying each process. However, early work has often attempted to incorporate domain-knowledge about the causality of processes in order to improve the process model itself[**shook˙AssessmentUseStructural˙2004**, **baker˙ClosingLoopEmpirical˙2017**, **hompes˙DiscoveringCausalFactors˙2017**, **wang˙CounterfactualDataAugmentedSequential˙2021**]. Among these, **narendra˙CounterfactualReasoningProcess˙2019** approach is one of the first to include counterfactual reasoning for process optimization. **oberst˙CounterfactualOffPol** use counterfactuals to generate alternative solutions to treatments, which lead to a desired outcome. Again, the authors do not attempt to provide an explanation of the models outcome and therefore, disregard multiple [**criterions for viable counterfactuals**] in XAI. [**qafari˙CaseLevelCounterfactual˙2021**] published the most recent paper on the counterfactual generation of explanations.[**Explain their approach.**] **The authors, use a known Structural Causal Model (SCM), to guide the generations of counterfactuals. However, this approach requires a process model which is as close as possible to the *true* process model. For our approach, we assume that no knowledge about the dependencies are known.**

Within the XAI context, **tsirtsis˙CounterfactualExplanationsSequential˙2021** develop the first explanation method for process data. However, their work closely resembles the work of **oberst˙CounterfactualOffPolicyEvaluation˙2019** and treat the task as Markov Decision Process (MDP). This extension of a regular Markov Process (MP) assumes that an actor influences the outcome of a process given the state. This formalisation allows the use of Reinforcement Learning (RL) methods like Q-learning or SARSA CITE . However, this often requires additional assumptions such as a given reward function and an action-space. For counterfactual sequence generation, there is no obvious choice for the reward function or the action-space. Nonetheless, both **tsirtsis˙CounterfactualExplanationsSequential˙2021** and **oberst˙CounterfactualOffPoli** contribute an important idea to incrementally generate the counterfactual. **hsieh˙DiCE4ELInterpretingProcess˙2021** build on this concept by proposing a system that generates counterfactuals milestone-wise. Their work is the closest to our approach. The authors recognised that some processes have

critical events, which govern the overall outcome. Hence, by simply avoiding the undesired outcome from milestone to milestone, it is possible to limit the search space and compute valid counterfactuals with counterfactual generation methods, such as the DiCE alogorithm. However, their approach presupposes that the critical event points are known and the outcome is binary. Especially, the first condition makes their approach heavily dependent on the data which is used.

2.5 Research Question

Having discussed the previous work on generating counterfactual sequences, a couple of challenges emerge. First, we need to generate on a set of criteria and therefore, require complex loss and evaluation metrics, that may or may not be differiantiable. Second, they cannot to be logically impossible, given the data set. Hence, we have to restrict the space to counterfactuals to viable solutions, while being flexible enough to not just be copies of already existing data instances. Third, using domain knowledge of the process significantly reduces the practicality of any solution. Therefore, we have to develop an approach, which requires only the given log as input while not relying on process specific domain knowledge. This begs the question, whether there is a process-agnostic method to generate sequential counterfactuals that are viable. In terms of specific research questions we try to answer:

RQ: Is there a process-agnostic method to generate viable counterfactuals?

- RQ1: Is there an evaluation metric, which reflects the viability of counterfactuals?
- RQ2: Is it possible to generate viable counterfactuals, even if the counterfactual has not been present in the data set?
- RQ3: Can we generate viable counterfactuals which are logically plausible?

We approach these questions, by proposing a schematic framework which allows the exploration of several independent components. The framework contains three parts. First, we need a predictive component which needs to be explained. The component should be capable of accurately predicting the outcome of a process at any step. This condition is favorable but not necessary. If the component is accurately modelling the real world, we can draw real-world conclusions from the explanations generated. If the component is inaccurate, the counterfactuals only explain the prediction decisions and not

the real world. The second part requires a generative component. The generative component needs to generate viable sequential counterfactuals which are logically plausible. A plausible counterfactual is one that the predictive component can predict. If the predictive component cannot predict the counterfactual sequence, we can assume that the generative model is unfaithful to the predictive component, we want to explain. The third component is the evaluation metric upon which we decide the viability of the counterfactual candidates. Figure 2.2 displays this approach visually.

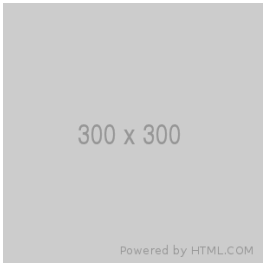

Figure 2.2: This figure shows a schematic representation of the framework which is explored in this thesis.

2.6 Formal Definitions

2.6.1 Process Logs, Cases and Instance Sequences

We start by formalising the log and its elements. Let $\mathcal{E}$ be the universe of event identifiers and $E \subseteq \mathcal{E}$ a set of events. Let C be a set of case identifiers and $\pi_\sigma : E \mapsto C$ a surjective function that links every element in E to a case $c \in C$ in which c signifies a case. For a set of events $E \subseteq \mathcal{E}$, we use a shorthand $\sigma \in C$ being a particular sequence $\sigma^c = \langle e_1, e_2, \ldots, e_t \rangle$ as a case of length t.

Furthermore, let $\mathcal{T}$ be the time domain and $\pi_t : E \mapsto \mathcal{T}$ a surjective linking function which strictly orders a set of events.

We assume that every e contains a set of data attributes. Hence, let $A = a_1, \ldots, a_i$ be a set of attribute names (e.g., timestamp, resource, action, etc.) and $\mathcal{F}$ be the universe of each attribute's representations $F \in \mathcal{F}$. For each element in A, we have a surjective function $\pi_{a_i} : A \mapsto F_i$ which links each event attribute a_i to its value $f_i \in F_i$. Each attribute value is represented with a vector space of varying dimensions $f_i \in R^d$.

We represent each event as a concatenated vector of its attribute values as $e_t^c = [f_1^c; f_2^c; \ldots; f_i^c]$, in which c specifies a particular case, a specific event

within the case at time t and i its event attributes.

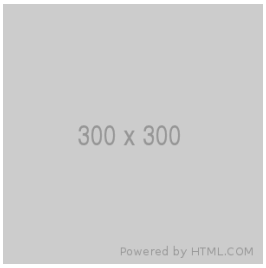

Figure 2.3: This figure show the representation of a particular event e and a case c.

2.6.2 State-Space Models

Generally speaking, every time-series can be represented as a state-space model[**kalman˙NewApproachLinear˙1960a**]. Within this framework the system consists of *input states* for *subsequent states* and *subsequent outputs*. A mathematical form of such a system is shown in Equation 2.1.

$$\begin{aligned} \mathbf{z}_{t+1} &= \mathbf{h}(t, \mathbf{z}_t, \mathbf{u}_t) \\ \mathbf{x}_t &= \mathbf{g}(t, \mathbf{z}_t, \mathbf{u}_t) \\ \dot{\mathbf{x}}(t) &:= \frac{d}{dt}\mathbf{x}(t) \end{aligned} \tag{2.1}$$

[**Here, u represents the input, x the state, t the time. The function g maps t, $z(t)$ and $u(t)$ to the next state $z(t+1)$. x acts as an output computed by function f which takes the same input as h. The variables z, u, t and y are vectors with discrete or continuous features. The distinction of $z(t+1)$ and $x(t)$ decouples *hidden*[2] states, from *observable* system outputs.**] Figure 2.4 shows a graphical representation of these equations.

The body of literature for state-space models is too vast to discuss it in detail[3]. However, for process mining we can use this representation to discuss the necessary assumptions with regards to process mining. In line with the

[2] A state does not have to be hidden. Especially, if we know the process and the transition rules. However, those are often inaccessible if we only use log data. Instead, many techniques try to approximate the hidden state given the data instead.

[3] For an introduction to state-space models see: XXX

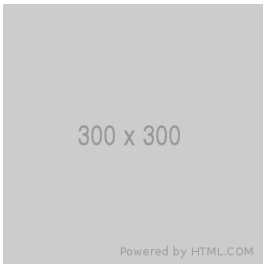

Figure 2.4: This figure shows a simplified graphical representation of a state-space model. Each arrow represents the flow of information.

process-definition in section 2.1, we can understand the Event Log as a collection of observable outputs of a state-space model. The state of the process is hidden as the *true* process which generated the data cannot be observed as well. The time t is a step within the process. Hence, we will treat t as a discrete skalar value to denote discrete sequential time steps. The input u represents all context information of the process. Here, u subsumes observable information such as the starting point and Process Instance-related features. The functions h and g determine the transition of a process' state to another state and its output over time. **Note, that this formulation disregards any effects of future timesteps on the current timestep.** As we establish in section 2.1, we can assume that a process is a discrete sequence, whose transitions are time-variant. In this framework, the goal becomes to identify the parameters of the functions h and g. Knowing the functions it is simple to infer valid counterfactuals. However, the function parameters are often unknown and therefore, require probablistic approaches.

We can formulate Equation 2.1 probablistically as shown in Equation 2.2.

$$\begin{aligned} \mathbb{E}[p\left(z_{t+1} \mid t, z_{1:T}, u_{1:T}, x_{1:T}, \theta_h\right)] &= \int z_{t+1} \cdot p\left(z_{t+1} \mid t, z_{1:T}, u_{1:T}, x_{1:T}, \theta_h\right) \\ \mathbb{E}[p\left(x_t \mid t, z_{1:T}, u_{1:T}, \theta_g\right)] &= \int x_t \cdot p\left(x_t \mid t, z_{1:T}, u_{1:T}, \theta_g\right) \end{aligned} \tag{2.2}$$

Note, that h and g are substitued with probability density functions parametrized with θ_h and θ_g. T signifies the full sequence including future timesteps. Both expectations are intractable as they require integrating over n-dimensional vectors. To solve the intractability, we characterize the system as a *Hidden Markov Process* and Probalistic Graphical Model (PGM). This framework allows us to leverage simplifying assumptions such as the

independece from future values and *d-seperation*. The stochastic process is shown in Figure 2.5.

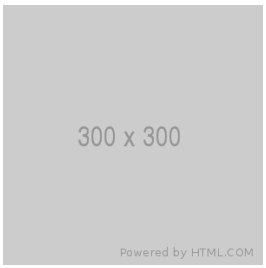

Figure 2.5: Figure shows a graphical representation of the stochastic process.

These characteristics change Equation 2.2 into Equation 2.3:

$$p\left(z_{t+1} \mid z_{1:t}, u_{1:t}, \theta_h\right) = \prod_{1}^{t} p\left(z_t \mid z_{1:t}, u_t, \theta_h\right) \tag{2.3}$$

$$p\left(x_t \mid z_{1:t}, \theta_g\right) = \prod_{1}^{t} p\left(x_{t-1} \mid z_{1:t}, \theta_f\right) \tag{2.4}$$

For $p\left(z_{t+1} \mid t, z_{1:T}, u_{1:T}, x_{1:T}, \theta_h\right)$, we ignore future timesteps, as T changes into t. *d-seperation* allows us to ignore all x of previous timesteps. The graphical form also decomposes the probability into a product of probabilities that each depend on all previous states and its current inputs. Previous x_t are ignored due to *d-seperation*. $p\left(x_t \mid t, z_{1:T}, u_{1:T}, \theta_g\right)$ only depends on its current state, which is in line with HMMs. Note, that we deliberately not assume a *strong Markov Property*, as the Deep Learning-Framework allows us to take all previous states into account. The *strong Markov Property* would assume that only the previous state suffices. At last, we assume that we do not model automatic or any other process whose state changes without a change in the input or previous states. Hence, we remove the dependency on the independet t variable. Only the previous states $z_{1:T}$ and the input information u_t remain time-dependent.

In this probablistic setting, the generation of counterfactuals, amounts to drawing samples from the likelihood of Equation 2.3. We then use the samples to reconstruct the most-likely a counterfactual $x^*_{1:t}$. Hence, our goal is to maximize both likelihoods.

[A number of AI techniques where developed to model this representation bla bla bla (HMM, Kalman, etc – Has further formalisation).]

Chapter 3

Methods

3.1 Datasets

3.2 Preprocessing

3.3 Framework

Chapter 4

Results

4.1 Evaluation

Chapter 5

Discussion

Chapter 6

Conclusion