0.0.1 Generating Counterfactuals

The topic of counterfactual generation as explanation method was introduced by Wachter, Mittelstadt, and Russell in 2017[24]. 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, Mittelstadt, and Russell did not account for the minimisation of feature changes and does not penalize unrealistic features. Furthermore, their solution cannot incorporate categorical variables.

A newer approach by Dandl *et al.* incorporates four main criteria for counterfactuals (see **??**) by applying a genetic algorithm with a multi-objective fitness function[5]. This approach strongly differs from gradient-based methods, as it does not require a differentiable objective function. However, their solution was only tested on static data.

0.0.2 Generating Counterfactual Sequences

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 and Provost and later Krause, Perer, and Ng are early examples of counterfactual NLP research[11, 14]. Their approach strongly focuses on the manipulation of sentences to achieve the desired outcome. However, as Robeer, Bex, and Feelders puts it, their counterfactuals do not comply with *realisticness*[21].

Instead, Robeer, Bex, and Feelders showed that it is possible to generate realistic counterfactuals with a Generative Adversarial Model (GAN)[21]. 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[15, 25]. By modelling the latent space, we can simply sample from a distribution conditioned on the original instance. Bond-Taylor *et al.* provide an overview of the strengths and weaknesses of common generative models.

Even though, a single latent vector model can theoretically produce multivariate sequences, it may still be too restrictive to capture the combinatorial space of multivariate sequences. Hence, most of the models within Natural Language Processing (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*[13]. Early models of this type of dynamic latent state-space models are the well-known *Kalman-Filter* for continuous 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, Shalit, and Sontag[11, 12]. Their Deep Kalman Filter (DKF) and subsequent Deep Markov Model (DMM) approximate the dynamic latent state-space by modelling the latent space given the data sequence and all previous latent vectors in the sequence. There are many variations[4, 7, 13] of Krishnan, Shalit, and Sontag's model, but most use Evidence Lower-Bound (ELBO) of the posterior for the current Z_t given all previous $\{Z_{t-1}, \dots, Z_1\}$ and X_t[8].

0.0.3 Generating Counterfactual Time-Series

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

First, Delaney, Greene, and Keane introduce a case-based reasoning to generate counterfactuals[6]. 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[16, p. 192]. In other words, if the model displays behaviour, which is not captured within the set of prototypical instances, most case-based techniques will fail to provide viable 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 continuous attributes can take values on a domain within infinite negative and positive bounds. These issues have not been explored in the paper of Delaney, Greene, and Keane, as it mainly deals with time series classification[6]. 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 *et al.* also uses a case-based approach[1]. 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 also introduce a framework that allows for flexibility depending on the dataset.

0.0.4 Generating Counterfactuals for Business Processes

So far, none of the techniques 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[2, 9, 22, 26]. Among these, Narendra *et al.* approach is one of the first to include counterfactual reasoning for process optimization[18]. Oberst and Sontag use counterfactuals to generate alternative solutions to treatments, which lead to a desired outcome[19]. Again, the authors do not attempt to provide an explanation of the models outcome and therefore, disregard multiple viability criterions for counterfactuals in eXplainable AI (XAI). Qafari and van der Aalst published the most recent paper on the counterfactual generation of explanations[20]. The authors use a known Structural Causal Model (SCM) to guide the generation of their 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, De, and Gomez-Rodriguez develop the first explanation method for process data[23]. However, their work closely resembles the work of Oberst and Sontag and treat the task as Markov Decision Process (MDP)[19]. 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. 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, De, and Gomez-Rodriguez and Oberst and Sontag contribute an important idea. The idea of incrementally generating the counterfactual instead of the full sequence. Hsieh, Moreira, and Ouyang has recently published an approach that builds on the same notion of incremental generation. Their approach has a very similar structure to our approach and appears to be the only one that we can compare our counterfactuals against.

For this reason, this thesis highlights some key differences and similarities. However, to understand the differences and similarities, we first have to establish some core concepts. In this section, we only discuss their approach, briefly.

The authors recognised that some processes have critical events which govern the overall outcome. Hence, by simply avoiding the undesired outcome from critical event to critical event, it is possible to limit the search space and compute viable counterfactuals. They use an extension of DiCE[17] to generate counterfactuals. However, their approach requires concrete knowledge about these critical points. We propose a Framework that avoids this constraint.

To our knowledge, the authors are also the first authors that try to optimize their counterfactual process generation based on criterions that ensure their viability. However, in our approach, we use different operationalisations to quantify the criterions.