

In order to explain the decisions of a prediction we have to introduce a predictive model, which needs to be explained. Any sequence model suffices. Additionally, the model's prediction do not have to be accurate. However, the more accurate the model can capture the dynamics of the process, the better the counterfactual functions as an explanation of these dynamics. This becomes particularly important if the counterfactuals are assessed by a domain expert.

In this thesis, the predictive model is an Long Short-Term Memory (LSTM) model. LSTMs are well-known models within Deep Learning, that use their structure to process sequences of variable lengths[1]. LSTMs are an extension of Recurrent Neural Networks (RNNs). We choose this model as it is simple to implement and can handle long-term dependencies well.

Generally, RNNs are Neural Networks (NNs) that maintain a state h_{t+1} . The state is computed and then propagated to act as an additional input alongside the next sequential input of the instance x_{t+1} . The hidden state h is also used to compute the prediction o_t for the current step. The formulas attached to this model are shown in

$$h_{t+1} = \sigma(Vh_t + Ux_t + b) \quad (1)$$

$$o_t = \sigma(W h_t + b) \quad (2)$$

Here, W , U and V are weight matrices that are multiplied with their respective input vectors h_t , x_t . b is a bias vector and σ is a nonlinearity function. LSTM fundamentally work similarly, but have a more complex structure that allows to handle long-term dependencies better. They manage this behaviour by introducing additional state vectors, that are also propagated to the following step. We omit discussing these specifics in detail, as their explanation is not further relevant for this thesis. For our understanding it is enough to know that h_t holds all the necessary state information. Figure ?? shows a schematic representation of an RNN.

The architecture of the prediction model is shown in Figure 2.

One input consists of an 2-dimensional event tensor containing integers. The second input is a 3-dimensional tensor containing the remaining feature attributes. The first dimension in each layer represents the variable batch size and *None* acts as a placeholder.

The next layer is primarily concerned with preparing the full vector representation. We encode each activity in the sequence into a vector-space. We chose a dense-vector representation instead of a one-hot representation. We also create positional embeddings. Then we concat the activity embedding, positional embedding and the event attribute representation to a final vector representation for the event that occurred.

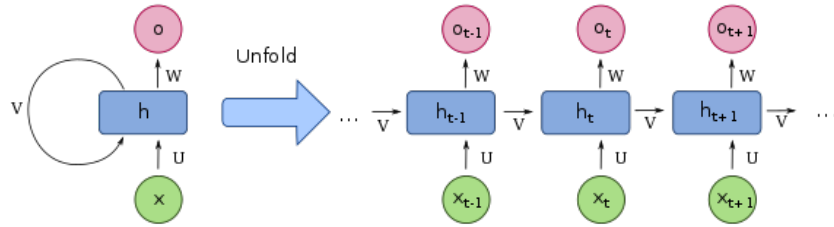


Figure 1: A schematic representation of an RNN viewed in compact and unfolded form??.

Afterwards, we pass the tensor through a LSTM module. We use the output of the last step to predict the outcome of a sequence using a fully connected neural network layer with a sigmoid activation as this is a binary classification task.

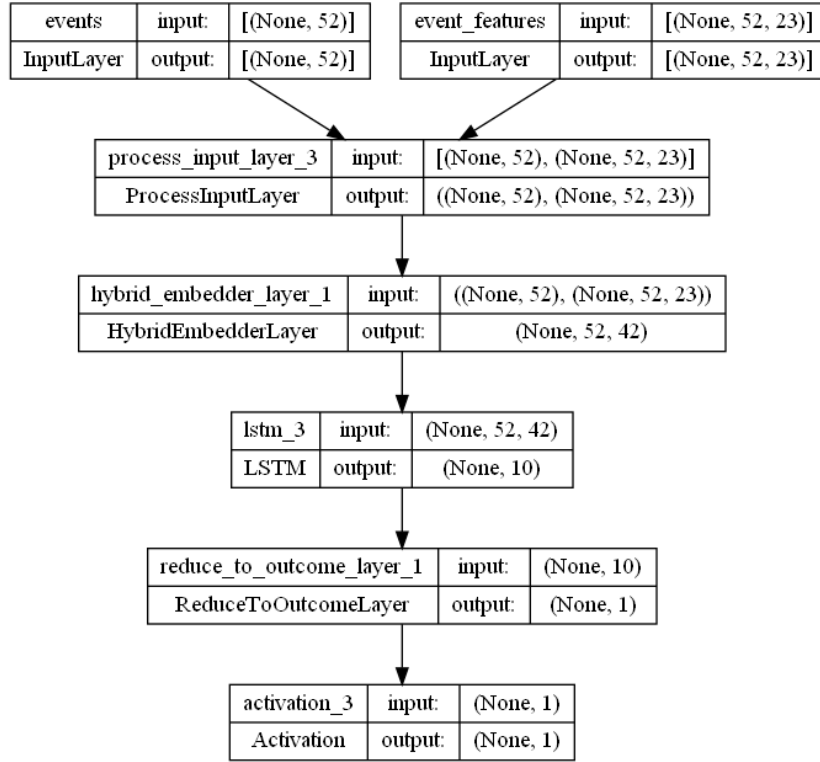


Figure 2: Shows the different components of the LSTM architecture. Each elements contains information about the input and output of a layer. None is a placeholder for the batch size.