2IOI0 DBL PROCESS MINING

GROUP 10 :

ELEFTHERIA KOLOKYTHA 1820672

SONIA MAXIM 1675656

PAVEL NIKOLOV 1789023

RUBEN LAGERWERF 1853287

YICHEN NING 1787551

INTRODUCTION

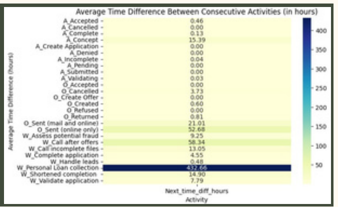
In today's data-driven world, the ability to forecast the progression of administrative processes can be really useful. Our goal is to develop a prediction model to predict the next activity in the process and its timing. It should also forecast the full sequence of forthcoming activities/suffix of a process, up to its conclusion. The core of our poster presents a series of visualisations that depict the performance and accuracy of our prediction models. We compare the baseline model with our estimators to demonstrate the improved performance of our model. (*Figure 6*).

DATA EXPLORATION

The DATASET used is the **BPI Challenge 2017**, it is an event log which contains loan applications of a Dutch financial institute. This dataset is comprised of 31.509 different process instances from January 2016 to February 2017. The process has 26 distinct activities including all the loan process logs, from submission to final decision that can be divided into three categories: application activities (A_), offer activities (O_) and workflow activities (W_). We have standardised timestamps, checked for missing values -removed rows with missing values for the features we are using-, as well as encoded categorical data with one-hot encoding, ensuring consistency in our dataset. To facilitate analysis, we split the dataset into training and test sets based on a temporal cut-off, which will allow us to evaluate the predictive performance of our models on unseen data effectively (*Figure 1*).

OBSERVATIONS

The heatmap visualisation provides a snapshot of the time dynamics within our process. Notably, "W_Personal Loan collection" offers stands out with a average time difference of 432.66 hours. In contrast, activities like "A_Cancelled" or "A_Denied" have an average time difference of 0.00 hours, suggesting that these are **end activities**. Interpreting this data could provide insights into which parts of the process are the most time-consuming and institution can improve its efficiency from streamlining and additional resources.



TRAIN - TEST SPLIT

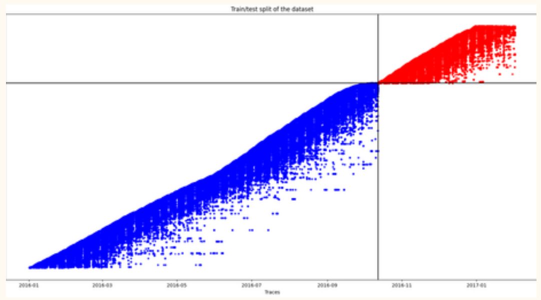


Figure 1 : Split up of Data to Train-Test

75% of the data is for **training**, and the remaining **25**% are for **testing**

This split is based on time and we have eliminated all cases that cross the time threshold. This way the models are trained and tested only on complete traces.

FEATURE ENGINEERING

Lag	The event type that precedes the current one - In event logs, the sequence of events might reveal a pattern useful for predicting future events.
Position	The position feature represents the number of previous events in the respective case - Same as above, the position in the case can correlate with the type of event
Weekday	The day of the week an event occurs (0 for Monday through 6 for Sunday). For each of these values x we take Sin(x * π/7) and Cos(x * π/7) to convert it to cyclic data. - Important since there is no work happens on Sundays and less on Saturdays, impacting the time prediction for the next event.
Days until next holiday	Indicates the days remaining until the next holiday. Since no work occurs on public holidays, this will also affect the time prediction.
Time since last event	The time that has passed since the last event in the same case as the current event
During work hours	A boolean value showing if the event took place during work hours (9:00 - 17:00, according to Everhour.com). This helps determine if a task requiring human completion will extend to the next workday.

NAIVE PREDICTOR

The naive predictor uses only the name of the event (**concept:name**), the time at which the event happened (**time:timestamp**), and the name of the case from which the event is (**case:concept:name**). A 'position' feature which indicates the position of an event in its trace, is used to get the most common event name for each position and the mean time until the next event. In figure 2.1 we present the accuracy of predictions. The prediction accuracy of the event after "A_Submitted" is so high because this type of event is only found at position 1 and is always followed by the same type of event.

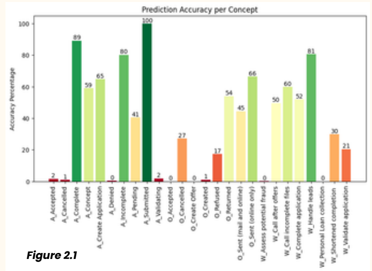


Figure 2.1

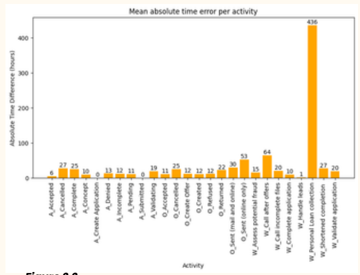


Figure 2.2

The visualisation 2.2 depicts the mean error in time prediction for different events in a process indicating the magnitude of the time difference error in hours. The visualisation can indicate the need for refinement in the predictive modelling process. The high mean absolute error for "W_Personal Loan collection" is caused by the small number of events of this type (only 22) and the time difference between them. The mean absolute error of around 0h for "A_Create Application" and "A_Submitted" happens because these events need only a few milliseconds and their positions are 0 and 1 respectively.

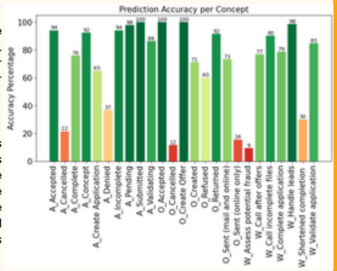
Figure 2 : Naive predictor

DECISION TREE

To predict the next event we decided to use a decision tree because it is a common and powerful tool for prediction analysis and an effective way of handling both non-numerical and numerical data. This model was trained using the entropy as a measure of impurity and the maximum depth set was 10. The features used for training were the one-hot encoded **event type** and '**lifecycle:transition**' and the engineered features '**position**', the one-hot encoded **lags** and '**next_concept:name**' which represents the event following the current one. These features were used because they may show a pattern important for predicting events. Other features, related to time, were excluded as they are not useful for this task.

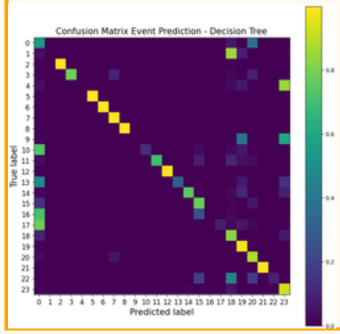
Bar Chart

The first plot depicts the accuracy of the decision tree when predicting the next event. Each bar represents how accurate our classifier was per event (the percentages of times our prediction was equal to the actual next event).



Confusion Matrix

The second graph displays the number of times the correct event label was predicted, as well as the times a different label was predicted. These numbers have been scaled in order to facilitate a meaningful comparison between the obtained values. The trend that can be spotted on the diagonal implies that our model predicts the correct event most of the time.



Limitations of using this type of classifier include sensitivity to errors and missing values. It is also known to be prone to overfitting and high variance, meaning that small changes in the input data can result in very different outcomes. This instability makes it less reliable. To tackle overfitting we made sure to perform cross-validation techniques to limit the depth and, in the future, potential high variances could be reduced further by using ensembling methods like random forests. Even though it presents very high accuracies for most of the event types, there are some for which the accuracy of our predictor is much poorer (O_sent (online), O_Cancelled), meaning that it is not always consistent, and, in practice not reliable for these particular types of events.

Figure 3 : Decision tree

LSTM

The time series prediction model recurrent neural network (RNN) is known for exploding / vanishing gradient problems. We use a long-short term memory (LSTM) model as it addresses this by having an input, output and forget gate to allow flow of gradient through time. We pass data to a doubly stacked LSTM layer to extract a hierarchy of features, then the output to a dense layer for classification (Figure 4.1).

Categorical features like **concept:name** and **lifecycle:transition** are one-hot encoded, and a standardscaler normally distributes numerical data like **requestedAmount**, **days until next holiday**, **during work hours**, and **weekday**. The sliding window method is used, where every 12 consecutive events in a case are used as an input for training, as it is memory efficient by training small chunks of data at once and limiting the use of padding, and allows thorough exploration of the data by overlapping the windows. The most common number of events in a case is 24, so the window size is set to 12, so that suffices can be predicted without instantly concluding the end of the case.

The model predicts one event upon receiving an input of 12 events, and then outputs the next predictions based on the newest 12 events. We evaluate event predictions using minimum edit distance (MED), where we find minimum editing operations to transform the predicted into the actual sequence; and the minimum alphabet distance (MAD), where we find the MED of the predicted and actual diversity of events. We achieve a MED and MAD of 3.66 and 2.26 with a suffix prediction of length 10. The feature "position" is excluded from the model as it worsens the performance according to the graph in Figure 4.2 below. Since it is clear that the next position equals current position plus one, the position should be predicted systematically (rather than by the model) for suffix predictions.

Root mean squared error (RMSE) evaluates each suffix time prediction. The first prediction has an error of +/- 34 hours on average and this gets larger over length of predictions. The feature seems to have no effect on the accuracy of time prediction possibly due to its large fluctuations in performance, despite the data being scaled down and normally distributed. This may be resolved with further outlier handling.

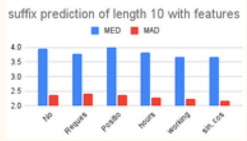


Figure 4.2 : Feature performances

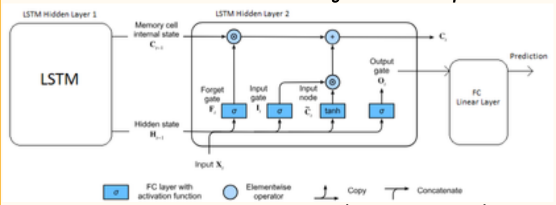
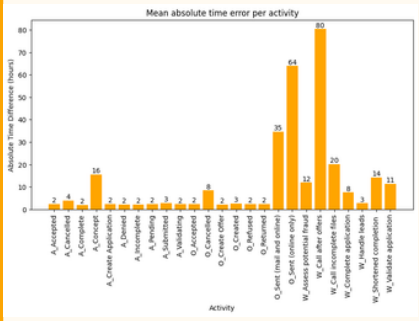


Figure 4.1 : LSTM Architecture

LINEAR REGRESSION



For the prediction of the time of the next event we chose a linear regression model due to its suitability for continuous data. The model's accuracy is measured by comparing predicted times against actual ones, using the mean absolute error for different activity types. This model factors in the **current activity type**, **time**, along with our engineered features of **weekdays** and **proximity to holidays**. These additional features where chosen because they can be used to explain why tasks performed by a human might take longer. Information about the previous trace of events and duration of the previous activity are less relevant for this prediction, that is why they are not included for this model. Generally, the model's performance is similar for most activities, with some notable exceptions where predictions significantly deviate. The activities where this is the case are often performed manually making it more difficult to predict. As can be seen when comparing the two mean absolute error graphs the linear regression is a lot better for most activities. There are a few exceptions which show us that this model is not that good for activities that take very short.

Figure 5 : Linear Regression

PERFORMANCES

NAIVE PREDICTOR

Event Accuracy : 42.1 %
Time RMSE : 65 HRS
Time MAE: 23 HRS

DECISION TREE

Event Accuracy : 84.5 %

LINEAR REGRESSION

Time RMSE : 61 HRS
Time MAE: 21 HRS

LSTM

Suffix Minimum Edit Distance: 3.66 / 10
Suffix Time RMSE: +/- 34 HRS (for first event)

Figure 6 : Perfomances

Reflection & Limitations

All our developed predictors outperform the basic model. Our final LSTM model would somewhat accurately predict the next steps in a trace, though its time predictions become less reliable further into the future. For individual cases, the decision tree and linear regression model can accurately predict the next activity in a trace but the time indication might not be as accurate. This approach might however not be as practical for real-world application.

- Test split validation accounts for potential bias due to missing longer cases in the test-set.
- The Naive Predictor fails to grasp the data's complexity, leading to potentially oversimplified predictions..
- The decision tree risks overfitting to the training data, but ensemble methods like random forests can prevent this.
- Linear regression isn't precise for instantaneous events, yet this could be improved by identifying if a task is automated or manual.
- Our LSTM's limited input length restricts its understanding of dependencies and its accuracy declines as it predicts further steps. Using a seq2seq LSTM or a transformer, which can process longer input sequences and capture complex relationships, could solve this issue.