Praktikumsleiter (Prof. Dr. Peter Loos)

Betreuer (Peter Pfeifer)

Projektarbeit PO 2023

Weingardt Oliver

[Anschrift]

7033544

I. Inhaltsverzeichnis
II. Abkürzungsverzeichnis
DFG – Directly Follows Graph
III. Tabellenverzeichnis

1. task and goal

Process Prediction is a technique of Process Mining, that deals with the prediction of running (not finnished) process instances. In the scope of this project general methods of process mining, especially the process prediction, should be implemented in Python and be visualized using a Dashboard. The created analysis should be evaluated using public datasets of the business process Intelligence Challenge (BPIC)

2. explanation of the topic

3. current state of science and technology

4. own contributions (with explanation of the current state)

4.1. Schedule and procedure

4.2. Central results

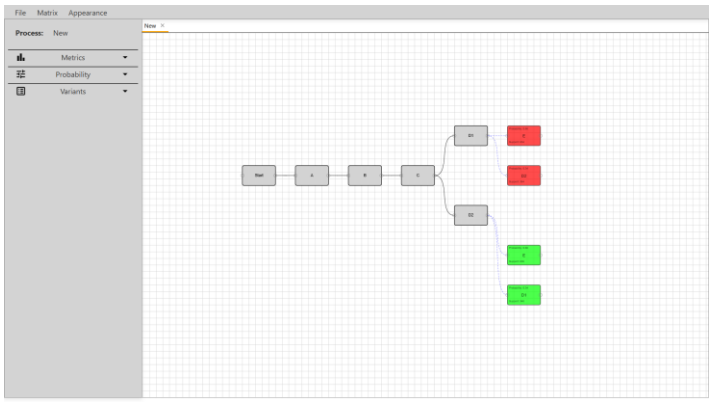
The result is a process mining tool which is based on a prediction matrix. A dashboard can be used to discover a process using the predicted next actions in the process. This uses a matrix containing prefixes and the possible actions following these prefixes with their respective probabilities.

It runs as a website in the browser and most computations are done on the server. This ensures that the tool is scalable to weak hardware. On the front end only computations considering the UI and the interpretation of the results of the backend computations are being handled.

4.2.2. Explanations of the Functions

The already discovered process model is displayed as a DFG. Activities are displayed as nodes in the DFG. The arcs between nodes should display, which activities can follow another. The predictions for following activities are being shown as slightly transparent, colorful nodes. The color of a prediction is determined by the predecessor of the prediction. This helps differentiate different groups of predictions. The calculation of how many predictions should be shown occurs either by an automated algorithm or is chosen by adjusting the minimal support and probability a prediction should have. This can be done using a slider. Furthermore, the probability and the support of the predicted activity are displayed on the node of the activity.

It is also possible to create arcs, without them being predicted by the matrix. For this, both the outgoing and the ingoing activity must be already discovered in the model, and we simply need to drag from the circle on the right of the activity node, we are starting from to the activity, the arc should end. This can be used to create actions sequences not possible by the predictions. This can give the user more freedom when creating the process model.



The DFG can be changed in multiple ways. Activities and arcs can be right clicked to open their respective pop-up menu. Here they can be deleted, and a comment can be given for activities. The comment feature has a wide variety of use cases, like further describing an activity, for example if the activity name is misleading. Another use case can be to give comments about the activity in the process model itself, for example if the user wants to note that the activity could be omitted for the process to work. Deleting an activity automatically deletes all their incoming and outgoing arcs. Using a lasso tool by holding the right mouse button can be used to select multiple activities, which will be highlighted in light blue. Selected activities can be deleted all at once by deleting a single activity. When dragging the selected activities, all of them get moved. This can be used if the user wants to move a large batch of activities without having to move each one individually. Navigating in the graph display can be done by dragging the background using the left mouse button. Zooming is possible by using the scroll wheel.

Multiple models can be worked on at the same time using the tabs. When staring at the site, there is already a tab open by default. Each tab can be closed using the x symbol

and when closing the last open tab, a completely new tab gets created automatically. In the file button in the header, entirely new tabs can be opened, or already saved models can be loaded from a file. Furthermore, the open model can be saved as a Json file, which can be loaded later. The tabs can be used if the user wants to compare multiple models of the same process, by testing multiple deviations, when choosing the predicted actions.

In the sidebar on the left, there is the option to name the current process model. There are also three dropdown menus containing different functionalities. The first dropdown menu shows performance metrics, like variant coverage, log coverage and fitness. Fitness can only be displayed for process models where the process log also exists. In the current implementation this only exists for the PDC_2020... matrix. In the second dropdown menu the minimum probability and support can be adjusted. By default, the algorithm to calculate the number of displayed nodes and therefore the support and the probability is used. But if it is needed to manually adjust the number of displayed predictions, then it can be disabled using the checkbox. When enabled, we use the sliders for support and probability to change the number of predictions displayed. This is used to control the number of predictions shown, by only choosing predictions whose support or probability is larger or equal than the selected values. Therefore, a smaller support and probability value lead to more predictions being shown. In the third dropdown menu, all variants in the matrix are being displayed with their support. There are options to search for variants and to sort them by support. Searching for variants works by giving the name of the actions included in a variant. When inserting multiple names divided by commas of spaces, then only variants including all the given actions are shown in the list. If a variant is already completely covered, then it is displayed in a different color.

In the header there is a “File”, a “Matrix” and an “Appearance” button. These open further dropdown menus. The file dropdown menu has the opening, storing and loading features as mentioned above. But it also contains the function to export the discovered process as a petri net. Here the petri net can be exported either as a picture or as a pnml file. The matrix dropdown menu has the functionality of changing between the different available matrices. There can also be more matrices uploaded to discover different processes. These get stored as cookies in the browser. It is also possible to delete uploaded matrices if they are not being used anymore. In the “Appearance” dropdown menu there are multiple settings for the appearance of the site and the graph. Here the grid in the DFG display can be disabled and the feature to give predictions a different color can be disabled. Furthermore, there is the functionality to auto position the graph. This can be used to reposition the activities in very large graphs such that they are more readable.

4.2.3. Workflow example

When trying to discover the process model from a given matrix, the user starts by choosing the matrix they want to use. The user can upload a matrix or use one of the predefined matrices. Then they can see the starting activity on the dashboard. From this there should be multiple outgoing predictions displayed. Based on the support and the probability, the user can now choose which activity should be happening next, by clicking the prediction. If the user is already familiar with the process, he can also choose the following activity based on prior knowledge and not only based on the support and probability values. This can be repeated until the user is content with the created process.

Metrics and the variants displayed in the sidebar can be used to give an overview over how far the process discovery has advanced. Based on the variants we can see whether a possible, important sequence of actions could have been ignored. The feature to open multiple tabs allows to create multiple process models using the same matrix, such that these models can be easily compared to each other. This can help to get a better understanding of the discovered model and the underlying process. Matrices can also be changed while working on the process model. This can be used, if multiple matrices of similar or the same process are available. Then the discovered model can be further expanded by activities from the other model, or simply the metrics of the different matrix can be displayed to check if the model is still applicable on different matrices (to fix overfitting for example).

If the process model is completely discovered, it can be exported as a petri net. It is possible to export it as a petri net picture. This can be used, if easy and fast access to the model is needed. We can also export it as a pnml file, such that it can be easily opened in other process mining tools, like pm4py applications. There it can be further analyzed and further metircs like fitness, generalisation, precision and [] can be viewed.

4.2.3. Used algorithms

4.2.3.1 Choosing predictions

First all prefixes in the matrix are being checked, to see if they are possible in the discovered model. We do this by iterating over each action in the prefix and checking whether it is possible in the currently discovered model. If a prefix is possible, we return all the predictions with a support and probability value, that is higher than the given minimum support and probability value. When using the auto probability algorithm, the check for the support and probability is omitted and which predictions to show is done later.

The predicted actions are grouped by the predecessor action, since multiple prefixes can lead to the same prediction. This can happen if there are multiple paths inside the process model that lead to the same action. The support for the grouped prediction is being calculated by adding all the support values and the probability is being calculated as the weighted average of all the probabilities. Here the weight for the weighted average calculation is the support of each probability in the group.

After that we have all the predictions possible in the current model. Now we need to check if the prediction needs to be added. If it already exists in the discovered model, it does not need to be added. So, if there exists an arc from the predecessor action to the action with the same name as the prediction. Otherwise, the prediction will be added.

4.2.3.2. Selecting a prediction

When clicking on a predicted action (on the node), this prediction should be added to the discovered model. For this, we first check if an action with the same name already exists. If not, then we first must add the action to the discovered model. After that, we add the arc from the predecessor of the prediction to the action.

4.2.3.3. Auto probability algorithm

The number of predicted actions is calculated as a function of the already discovered actions:

$$f(x) = 2 \cdot \ln(x)^2 + 3$$

This ensures that the number of predicted nodes grows fast for few actions in the DFG, while not growing too much for high number of nodes. The function assures a minimum number of three predicted nodes.

Then we get the predictions as described in 4.2.3.1. These are returned as a dictionary with their support. Now we must reduce the number of displayed predictions to the calculated numbers of displayed prediction nodes, while only showing predictions with the highest support. For this we first order a copied list of the support values of each prediction descending by support. If the calculated number of predictions is n, then we output the n-th support value. Since the predictions are returned as a dictionary, we cannot sort the dictionary itself and therefore we cannot extract the predictions directly from the ordered list. Therefore, we must check for each prediction in the dictionary, if its support is larger than the calculated minimum support.

After that, to further increase the readability of the graph, we ensure that after each already discovered action in the model, there are a maximum of three predictions following. For this also the nodes with the lowest support are removed.

4.2.5 Used tools and frameworks

The frontend of the site has been coded in Typescript using React and Vite. API calls are handled by Axios. The work area, where the process model is displayed as a DFG is handled using the Conva.js library.

The backend is done in Python using flask. The conversion to a petri net and the fitness calculation is handled by pm4py and the creation of the picture of the petri net is done using graphviz. The matrices from the csv files are opened using pandas data frames.

4.2.6. Used performance metrics

4.3. Discussion of the results

5. summary