Application of Neural Network Tools in Process Mining

László Kovács, Erika Baksáné Varga, and Péter Mileff

Abstract—**Dominant current technologies in process mining use schema induction approaches based on graph and au- tomaton methods. The paper investigates the application of neural network approaches in schema induction focusing on three alternative architectures: MLP, CNN and LSTM networks. The proposed neural network models can be used to discover XOR, loop and parallel execution templates. In the case of loop detection, the performed test analyses show the dominance of CNN approach where the string is represented with a two- dimensional similarity matrix. The usability of the proposed approach is demonstrated with test examples.**

Index Terms—**process mining, convolutional neural network, graph schema induction.**

I. Introduction

PROCESS mining is a technique to discover, analyze, and monitor processes in an objective manner. It uses log data from corporate information systems and turns them into insights and actions. In this sense process mining is a subfield of data science: it requires the availability of data and aims at improving processes [1]. The induction of complex schema or grammar models is a key challenge in knowledge engineering. The input for schema mining is a set of sequences (traces) and the engine determines the general schema graph covering the input set. The input for schema mining is given with an event log, which is a list of event traces. The traces of the event log are generalized, merged into a schema graph.

The first software systems for schema induction usually used a pattern matching approach to determine the common section in the sentences. In most cases, the induction was based on a set of transformation rules defined by human experts. Current toolsets for process mining are dominated by the graph or automaton oriented algorithms [2] usually starting with the construction of the Direct Follower Graph which is a graphical representation of a process. Traditional frequent pattern mining techniques run into their limits when dealing with massive datasets [3]. Therefore, more advanced algorithms apply tree-based representation [4]. These incremental methods derive relevant patterns recursively. One of the most widely used industrial approaches is the inductive mining algorithm [5], which uses a top-down discovery algorithm. The top-down method recursively decomposes the event log into smaller event logs. The method first converts the event log into a corresponding DFG (direct follow graph), then it simplifies the initial graph into a compact schema graph. Beside the graph-based standard approaches, we can find some recent approaches on application of neural networks in process mining [6]. We can consider graph schema induction as a special type of classification problem, where the final output category corresponds to a complete schema graph. The schema graph is usually constructed in an incremental way by selecting the next winner event or events and the edges in the graph denote the adjacency relation. Considering simple event sequences, the dominating approach for next event prediction is to use Recurrent Neural Networks (RNN), where the output signal of a processing step is used as input component for the prediction of the next element in the sequence. In [7], the RNN network model was applied for the process discovery task. Current solutions are able to detect XOR branches in the trace log, but these methods still miss some crucial functionality required by industrial problems. The main goal of this paper is to show our proposals on the adaptation of neural networks in process mining to cover some missing functionalities, like AND branches and loop detection [8].

The authors are with the Department of Information Technology, University of Miskolc, Miskolc, Egyetemváros, Hungary. E-mail: {laszlo.kovacs, erika.b.varga, peter.mileff}@uni-miskolc.hu

DOI: 10.36244/ICJ.2023.5.3

II. Background

A. Process Mining with Graph Methods

Today's era is about automation which also affects complex business and administrative processes. For the automation of processes, the most important prerequisite is to have process models which can be transformed into running systems. Thanks to process mining methods, the required models can be discovered by extracting knowledge from event logs recorded by information systems. The general process discovery problem is defined in [1] as finding an algorithm that maps an event log (L) into a process model so that the model is representative for the behavior seen in the event log. This task is highly challenging, since an input event log contains a collection of historical cases of a given process, called traces, which are sequences of actions while the expected output model should be compact and meet the requirements of fitness, precision, generalization and simplicity. In order to produce a generalized model, the control-flow structures underlying in the data sequences should be discovered.

The control-flow patterns used in process modelling are presented in [9] and [10]. The identified 43 patterns are classified into 8 classes, amongst which the basic patterns that can be explored in the event log are:

- sequence,
- parallel split,
- synchronization,
- exclusive choice, and
- simple merge.

In sequential routing, a task in a process is enabled after the completion of the preceding task. There can be two different types of splitting nodes in a process model. Parallel split is applied when a single thread of execution is split into two or more branches which are triggered concurrently (AND node). Exclusive choice corresponds to conditional routing. In this case, as soon as the incoming branch is enabled, the thread of control is passed to exactly one of the outgoing branches (XOR node). The splitted branches need to be merged after a while, i.e. an AND or XOR node are generally succeeded by a synchronization or simple merge node, respectively. The difference between these nodes is that a synchronizing node means that all the activated incoming branches should be completed before the thread of control is passed forward; while a simple merge node waits for only one branch to be completed [11].

Graphical process models usually have special notations for the elements that influence the flow of control. These nodes interconnect activities in the conceptual model, but there is no sign of them in the event log. This makes the process discovery problem challenging, since the developed algorithms have to explore the control structures hidden in the event sequences.

B. Standard Tools in Process Mining

Process mining tools utilize existing data from corporate information systems to provide dynamic visualization of the processes and to perform process mining tasks such as process discovery and conformance checking.

The first framework that has been designed to combine different process mining algorithms, is ProM [12]. This framework is flexible regarding the input and output format, and allows for easy implementation of process mining methods. Three types of graph-based algorithms can be plugged in. One can explore the process perspective to find a general characterization of all possible paths, expressed for example in terms of a Petri net or Event-driven Process Chain (EPC). With respect to the organizational perspective, the mining method either structures the organization by classifying people in terms of roles and organizational units, or maps the relations between them. Thirdly, cases can also be investigated by characterizing them with their path in the process, by the agents working on them, or by the values of the corresponding data elements.

A more sophisticated tool is the open source PM4PY developed by the Fraunhofer Institute using Python [13]. It supports several input and output formats, and a variety of process discovery algorithms creating procedural process models (such as the Alpha Miner, the Inductive Miner and the Heuristics Miner algorithms) or methods producing descriptive models. It also allows for conformance checking, object-centric process mining, organizational network analysis and provides some features useful for the application of machine learning techniques. Most recently, PM4PY offers some integrations with OpenAI (e.g. with ChatGPT) for getting insights automatically.

However, a great disadvantage of traditional graph-based process mining methods is the use of recursion which results in high memory consumption and long execution time. In our approach, we apply approximation in discovering a process model by means of neural networks.

C. Process Mining with Neural Network Methods

Neural networks can be utilized in process mining to enhance various aspects of the analysis and optimization of business processes. One of their applications is when they are used to predict future events in a business process. Compared to traditional process mining techniques, such as Petri nets and the Business Process Model Notation (BPMN), deep learning methods have proven to achieve better performance in terms of accuracy and generalization power.

In [14], Hanga et al. propose a method that combines the benefits of visually explainable graph-based methods with more accurate deep learning methods. Among neural networks, the RNN architecture has the capability to provide the context for each following prediction that helps in preserving the state in which the decision was made. Therefore, in this approach, an LSTM model is employed first to find probabilities for each known event to appear in the process next. These probabilities are then used to generate a graphical process model graph.

Obodoekwe et al. [15] used convolutional neural network (CNN) for predicting the next activity in an event trace. The method first detects the spatial structure within the order of historical event sequences and then transforms them into 2D images. The images are then trained using the CNN network to generate a deep learning model that can predict the next activity in an ongoing process. The feasibility of the approach was evaluated using Helpdesk event logs and the results show that the proposed CNN-based method provides highly accurate next activity prediction and is faster in training and inference than the LSTM-based approach.

The most widely studied problem of process mining is automatic process discovery. Several approaches have been proposed, but the applicability and effectiveness of these approaches depend on event log features and the structure of the processes. When applied to real-life event logs, the majority of traditional process discovery methods produce broad and spaghetti-like models, or models with poor fitness and precision [1].

Shunin et al. [7] try to find patterns in event logs using a neural network. The algorithm extracts an RNN's internal state as the desired transition system that describes the behaviour present in the log. One of the main advantages of using this architecture is its natural ability to detect and merge common behavioural parts that are scattered across the log. Another benefit is that the models derived by the approach absolutely fit to the event log.

Sommers et al. [16] applied graph neural networks in process discovery. They encode the discovery problem as a graph of three parts. The first part of the graph is the trace graph representing the event log. The second part of the graph is a candidate Petri-net, which is a possible result model. The third part of the graph are the links from the event nodes of the trace graph to transitions in the candidate model. Their approach is to utilize a collection of neural networks, each of

which takes a segment of the graph as input and simulates a distinct stage in the gradual construction of a Petri-net model from an event log. The method was evaluated on synthetic and real-life data and compared to other methods, achieving the highest simplicity while competing with Inductive Miner, Heuristic Miner, and Split Miner methods in terms of F-score.

In this paper, we present a neural network architecture for the detection of AND (parallel) branches in the event log. With the help of this approach, the engine can determine the parallel processes in the traces and it builds up a process schema graph related to more actors. The proposed engine uses a two-level representation approach where the bottom level corresponds to the single actor level activity chain. These homogeneous segments are merged by using synchronization nodes. The top level contains the synchronization graph of the agent level segments. The second neural network architecture presents a novel loop detection approach which can be used to discover tandem repeat sections. The proposed method first converts the sequence into an image matrix format and this matrix will be sent to a CNN convolutional network. The category labels correspond to the different loop kernel positions in the sequence. The next model integrates these two neural network models to have a more general schema induction engine for process mining. The presented novel network models were implemented in Python Tensorflow/Keras framework. In order to test the induction engines on event logs of different complexity levels, a test set generator application was developed in the research project. The target schema is constructed with a visual editor, and the engine generates the related event log of given size. In the test experiments, we compared the efficiency of our proposed models with some standard graph based engines. In the paper, we analyze the test results showing both the benefits and limitations of the neural network approach.

In the tests we used three main neural network architectures:

- multi-layer perceptron model (MLP), which is a standard NN tool for both general classification and regression,
- recurrent neural model (LSTM), and
- convolutional neural model (CNN).

The detection of branching nodes of the schema graph can be implemented either with the standard MLP network or with the LSTM recurrent model. The main goal of the neural network is to predict the next element of the investigated sequence. The prediction is based on the previous elements, thus the input vector is given as the description vector of the preceding elements. In the standard approach, the engine outputs only the winner category, in this way generating only one next element. According to the literature [17], the standard way of generating sequence branching is to consider not only a single winner category, but a group of best candidates. The size of the winner group is usually an input parameter in the algorithm.

III. Discovery of Parallel Branches

Parallelism denotes parallel workflows of different actors and resources, and the split and join control nodes denote an artifact level dependency among the different branches. For example, we consider a workflow to produce a mobile phone. In this process the production of the different components can be executed in a parallel way. A synchronization join node denotes the case when the next assembly step requires the availability of all components produced in the preceding steps. These control nodes can be automatically discovered if the event log contains an artifact attribute as well. The artifact attribute identifies the target, i.e. the object of the given action. Using this parameter we can discover the artifact level dependency between the different actions of the event log.

Beside the actor events, the extended input event graph for the training process contains also synchronization control nodes which describe the adjacency relationship among the event sequences. We assume that every control node has an input set of event sequences and an output set of event sequences. Similarly to Petri-nets, the join control node is triggered only when all of the input sequences are finished. If the transition is triggered, all output sequences will start the execution.

The event graph structure is defined as

$$\sigma = (N_\sigma, \rightarrow_\sigma)$$

where

- $e \in W \times A$ is an actor event where $A(e)$ denotes the actor of the event, $T(e)$ denotes the timestamp of the event and W denotes the set of event types (activities)
- $c \in C$ is a control event, every c has a timestamp denoted by $T(c)$;
- $E_\sigma = (e_1 e_2 ... e_k : e_i \in W \times A, \forall i, j : A(e_i) = A(e_j))$: the actor event sequence node;
- $C(c)$: the control event nodes;
- $N_\sigma = E_\sigma \cup C_\sigma$: the nodes in the graph instance;
- $\rightarrow_E \subseteq E_\sigma \times C_\sigma$: the edges from actor events to control events;
- $\rightarrow_C \subseteq C_\sigma \times E_\sigma$: the edges from control events to actor events;
- $\rightarrow_\sigma = \rightarrow_E \cup \rightarrow_C$: the edges in the graph.

In the preprocessing phase of the proposed method, the engine determines the related synchronization nodes first. Node mining is based on the following considerations:

1) Event nodes are processed in temporal order, and the current event is denoted by e_i.
2) The set of output artifacts of e_i are stored in out_i.
3) The set of adjacent events $E_j = \{e_j\}$ are determined, where the input artifacts correspond to some elements of out_i, and there is no other intermediate event processing these artifacts.
4) If the actors of the matching e_i and e_j event pair are different, a synchronization node e_s is needed between them.
5) The actor of e_i (a_i) is added to the input-actors of e_s, and the output actor set of e_s is extended with a_j.
6) If an actor is not present in the actor set of e_s while being active at e_s through an artifact-level dependency with some events in the output-actor list of e_s, then it will be included into the input-actor set of e_s. A similar method can be used to extend the output-actor list of e_s.
7) The previous two steps are repeated until a closure of the input-output artifact relationships is achieved at e_s.

TABLE I
ACCURACY COMPARISON OF SEQUENCE PREDICTION NETWORKS

Dataset	LSTM	MLP	NH-MLP	BE-MLP	BE_LSTM
pdc_2016_1.xes	63.5%	63.4%	63.4%	63.9%	64.1%
load_random	58.2%	57.5%	56.6%	58.7%	58.0%
pdc_2016_9.xes	82%	82.5%	81.2%	81.8%	82.6%
pdc_2017_5.xes	62.4%	62.8%	63.7%	64.8%	64.2%
pdc_2019_2.xes	64.6%	65.2%	63.9%	66.27%	65.6%

Considering the efficiency of the proposed models, we performed a comparison test on benchmark sequences. The investigated network models are:

- MLP : baseline MLP,
- LSTM : baseline LSTM,
- BE-MLP : MLP with union-based reduction,
- BE-LSTM : LSTM with union-based reduction,
- HN-MLP : MLP with NN-based reduction.

The first architecture is based on the standard multi layer perceptron neural network model (MLP) which is suitable to perform feature vector based value prediction. This model uses a relatively simple architecture, thus the training has a lower cost. The second version uses a recurrent neural network architecture (LSTM) that is used for prediction on value sequences of arbitrary length. It contains a more complex architecture that enables remembering of previous events in an efficient way. Due to the higher complexity, it is usually harder to find the optimal network parameters. The three other versions apply a sequence reduction preprocessing step. In this reduction phase, the long event sequence is reduced to a shorter one, where the reduction can be based on the value aggregation (MLP with union or LSTM with union) or an extra MLP neural network is trained to perform the reduction step (MLP with NN-based reduction).

Regarding the implementation parameters, the model consists of beside the output or hidden Dense layers, also an LSTM layer. For the hidden layers, we have used the relu activation function, while the output layer used the softmax activation function. Regarding the optimisation parameters, we applied the Adam optimizer with the categorical crossentropy loss function with the accuracy metrics. For the test evaluation of the different methods, we used the benchmark datasets for the Process Discovery Contest events. These competitions are organized by the IEEE Task Force on Process Mining Group . The name of the data file refers to the year of the contest and to the index of the data file. The dataset can be downloaded from the homepage of the contest (*https://www.tf-pm.org/competitions-awards/discovery-contest*). The datafiles contain the event logs in XES standard format.

The results of the comparison tests are presented in Table I. The accuracy values are given in percentage unit. The main conclusion is that the BE_MLP network type is a good choice for our architecture as

- it provides the best accuracy for most of the benchmark datasets, and
- it has the lowest execution cost.

IV. LOOP DETECTION

Loop detection is a key task in schema mining of event logs. The basic operation in loop detection is the discovery of tandem substrings. The task of tandem substring detection can be given with the following formal description. Having an alphabet A, strings are the final sequences based on A:

$$s = a_1, a_2, ..., a_m, a_i \in A.$$

A substring of s is

$$s' = a'_1, a'_2, .., a'_k$$

if

$$\exists i : a_i = a'_1, ..., a_{i+j-1} = a'_j, ..., a_{i+k-1} = a'_k.$$

A substring s' is a tandem substring if

$$\exists i : a_i = a_{i+k} = a'_1, ..., a_{i+j-1} = a_{i+k+j-1} = a'_j, ..., a_{i+k-1} = a_{i+2*k-1} = a'_k.$$

In order to show the ability of a Neural Network architecture to detect loop structures in sequences, we have developed a special network type using the CNN convolutional architecture. The main drawback of the baseline MLP approach is that it is very sensitive to the specific values at each position of the sequence. On the other hand, loop detection should be insensitive to the actual character values. Thus, for example, the sequences ”abcbch” and ”bagagb” need to be equivalent as both contain a loop at the second position. Otherwise it would take a long effort to generate a suitable training set with a good covering for cases of repetition.

In the initial tests, we have compared three candidate neural network architectures: multi-layer perceptron, convolutional CNN and recurrent LSTM. In the case of MLP, the input vector is the one-hot encoded version of the investigated string. For each position, a one-hot encoded vector of size M is given where each position denotes an element of the alphabet. The output vector describes all possible repetition positions, where each position corresponds to a (start position, end position) pair.

The input vector for CNN is given by a two-dimensional similarity matrix. The position (i, j) contains the similarity value of the symbols at the position i and at the position j. Thus, the string is converted into a two-dimensional image matrix, where the repeat sections can be characterized by a set of special lines in the image. The output format is similar to the vectors used in the MLP network architecture.

Considering the applied neural network architecture for loop detection, we used a CNN model with 14 layers including the core convolutional layers and the maxpooling layers. The architecture model is presented in Fig 1.

In the case of LSTM, the input vector contains the one-hot encoded format of the string, just like in the MLP network. Here, the output is a number for each position, denoting the size of the repeat section at the given position. Thus, the processing of the string will generate a sequence of numbers where a value greater than 1 denotes a repetition.

The results of the performed accuracy tests are summarized in Table II.

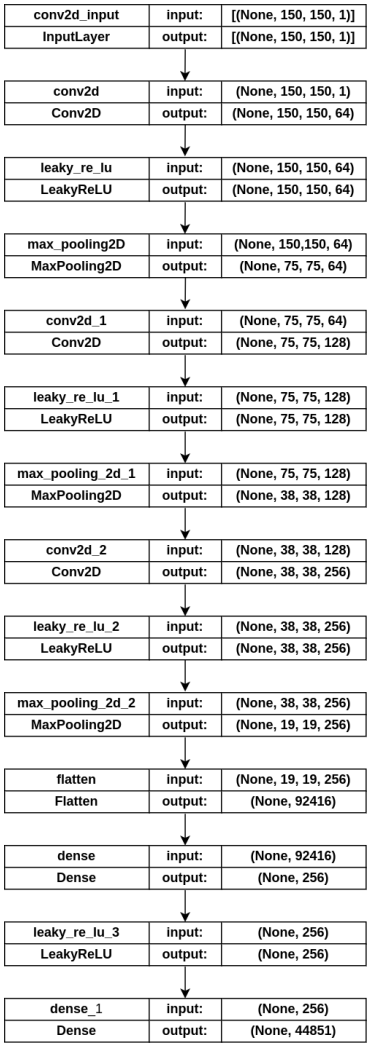

Fig. 1. Architecture of the CNN neural network for loop detection.

TABLE II
EFFICIENCY COMPARISON OF THE EXAMINED NN ARCHITECTURES

Method	N parameters	Time [s]	V accuracy	T accuracy
MLP	27,590,861	26	96%	8%
CNN	175,631,337	288	100%	99%
LSTM	2,527,212	359	96%	94%

In Table II the columns denote the following measures:

- N parameters: the complexity of the network, i.e. the number of graph parameters.
- Time: execution time of an epoch in seconds.
- V accuracy: validation accuracy.
- T accuracy: test accuracy.

Based on the preformed tests, we can see that the CNN network is the winner of the neural network approach. One important remark on the results is the very large differences between the test and verification accuracy values for the MLP architecture. This shows that there is significant overfitting in this case.

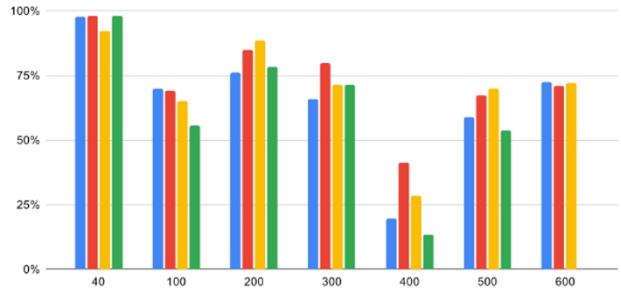

Fig. 2. Accuracy test of the different reduction methods.

Considering the winner CNN variant, repetition appears in the image as a line of pixels parallel to the main diagonal. Using this representation format, CNN can be used to recognize repetitions. One key issue in the CNN approach is the high memory cost, as for a string of length m the number of input neurons of the network will be m^2. Thus, we can reach our hardware limits relatively quickly during CNN training or search for repetitions. In order to overcome this problem, we have applied the following reduction methods:

- Bitmap-based reduction,
- Average aggregator, and
- Adjusted reduction.

In the tests, we have compared three reduction methods. In the first one, (bitmap reduction), in the initial bit matrix, the submatrices are replaced with a single cell where the value is calculated with the max operator. In the second version, the content of the submatrix considered as a matrix of float values and the average value will be stored in the reduced matrix. The third variant applies a special calculation which is sensitive to the diagonal directions as the loops generate specific lines (parallel to the diagonal) in the input matrix.

The test results for comparing the efficiency of the different reduction methods are presented in Fig. 2. In this figure, the X axis shows the length of the pattern window, and the Y axis denotes the achieved classification accuracy. The leftmost bar is for the Adjusted reduction method, the second is for the Average aggregator, the next is for the Bitmap-based reduction method, and the last one is for the baseline method without reduction. Based on the performed tests, the winner approach is the Average aggregation method.

Another way for reducing memory usage is to apply a different representation format. We have tested two additional variants, namely

- A: Neural network to detect whether the word contains repetition or not. In this case, the output vector contains only two dimensions.
- B: Neural network to detect the start (or end) position of the repeat section, where the size of the input vector is of linear cost.

Based on the performed tests, we can say that the CNN method significantly dominates the other variants in accuracy parameters. The test results are summarized in Table III.

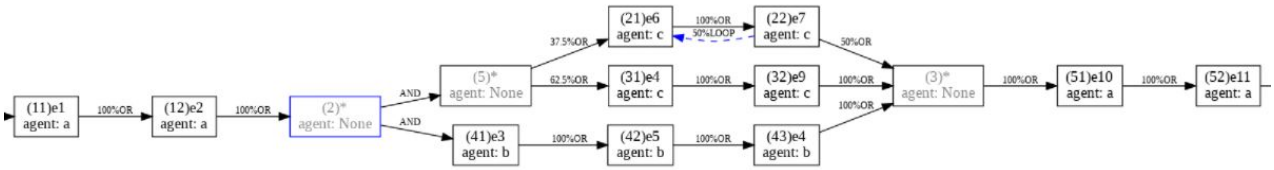

Fig. 3. Sample workflow schema B.

TABLE III
EFFICIENCY COMPARISON OF THE EXAMINED NN ARCHITECTURES USING ALTERNATIVE REPRESENTATION FORMATS

Task	Method	Time [s]	V accuracy	T accuracy
A	MLP	17	100%	87%
A	CNN	12	100%	100%
B	MLP	30	100%	6%
B	CNN	50	100%	100%

V. Implementation and Tests

In the tested model, three actors are defined: a,b and c. The list of available events, which are given by their IDs, actors and the related execution time intervals are as follows:

$$e1(a),(1,2) \quad e2(a),(3,5) \quad e3(b),(1,4) \quad e4(c),(2,3)$$
$$e5(b),(4,5) \quad e4(b),(5,6) \quad e6(c),(1,3) \quad e7(c),(2,4)$$
$$e9(c),(1,4) \quad e10(a),(3,4) \quad e11(a),(2,4)$$

The graph structure of the schema is presented in Fig. 3. For the training phase, 1000 cases were generated for schema B.

In the first processing phase, the engine determined the occurrences of synchronization events. The module then processed the traces in the log, and generated a list of synchronization events for each trace.

Based on the generated synchronization event sequences, as input training set, the constructed neural network model M will predict the following synchronization sequence:

```
['EMPTY', 'C2', 'C3', 'EOS']
```

In the next phase, the engine predicts the agent-level sequences in the following form:

```
a : ['e1', 'e2', 'EOS']
b : ['e3', 'e5', 'e4', 'EOS']
c : ['e4', 'e9', 'EOS']
a : ['e10', 'e11', 'EOS']
```

After performing more trace generation experiments, the action list of agent c may vary. A typical output is the action chain containing actions 'e6' and 'e7':

```
c : ['e6', 'e7', 'e6', 'e7', 'e6', 'e7',
     'e6', 'e7', 'e6', 'e7', 'e6', 'e7',
     'e6', 'e7', 'EOS']
```

During the prediction, the neural network outputs the prediction weights for the different event categories. The proposed measure weight ratio expresses the relation of these weights,

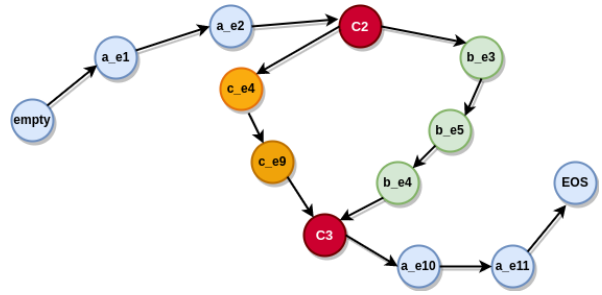

Fig. 4. Generated schema graph for single selection case.

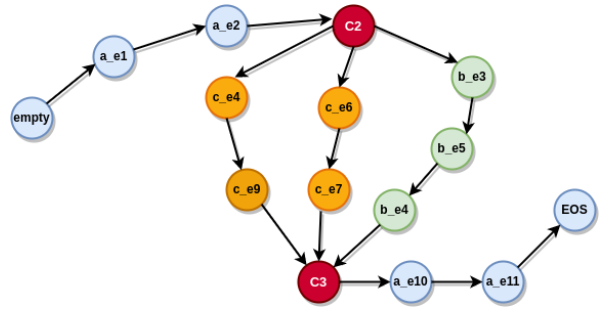

Fig. 5. Generated schema graph for multi-selection case.

the higher is this ratio the better is the prediction quality. If we consider the weight ratio between the best and the second best event sequences for agent c, we can see that most of the steps are unambiguous, except for the starting event which has a very low weight ratio. Namely, the higher the ratio, the stronger is the unambiguity. The next list shows these ratio values at the different steps for the best event sequence of agent c:

```
y e4  , weight ratio: 1.464207
y e9  , weight ratio: 234.37918
y EOS , weight ratio: 237.77512
```

Considering the resulting graph for a single selection case (only the best candidate is selected as next event), we can see that it contains a parallel execution section, where both agents b and c are active (Fig. 4). If we use a multi-selection prediction approach, where alternative routes are allowed, we get the graph presented in Fig. 5.

If we compare these graph results with the standard inductive miner approaches, we see that the proposed variant can manage more functionality and provides a more accurate result for the tested example than the standard solution (Fig. 6).

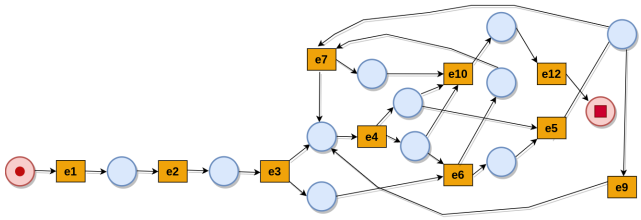

Fig. 6. Schema generated by the Inductive Miner.

VI. Conclusion

Beside the standard automaton oriented approaches, the neural network architectures are good alternatives for automated process schema mining. The analysis presented in this paper shows that the proposed neural network architectures are suitable to perform the same schema induction task as the standard graph-based tools. The proposed methods build up the correct schema graphs. The key benefit of the proposed NN approach is that it can discover also parallel sequences related to different agents in the schema graph, unlike the usual schema induction methods. The further research is aimed at optimisation of the neural network architecture for larger graph schema.

References

[1] W. Van der Aalst, *Process mining: Data science in action*. Springer: Heidelberg, 2016. https://doi.org/10.1007/978-3-662-49851-4

[2] J. Liu, S. Yan, Y. Wang, and J. Ren, "Incremental mining algorithm of sequential patterns based on sequence tree," *Advances in Intelligent Systems*, pp. 61–67, 2012. DOI: 10.1007/978-3-642-27869-3_8

[3] T. Truong-Chi and P. Fournier-Viger, "High-utility pattern mining: Theory, algorithms and applications," *A Survey of High Utility Sequential Pattern Mining*, pp. 97–129, 2019. DOI: 10.1007/978-3-030-04921-8

[4] X. Liu, L. Zheng, W. Zhang, J. Zhou, S. Cao, and S. Yu, "An evolutive frequent pattern tree-based incremental knowledge discovery algorithm," *ACM Transactions on Management Information Systems*, pp. 1–20, 2022. DOI: 10.1145/3495213

[5] Y. Lu, Q. Chen, and S. Poon, "A novel approach to discover switch behaviours in process mining," *International Conference on Process Mining*, pp. 57–68, 2021. DOI: 10.1007/978-3-030-72693-5_5

[6] H. Weytjens and J. D. Weerdt, "Process outcome prediction: CNN vs. LSTM (with attention)," *International Conference on Business Process Management*, pp. 321–333, 2020. DOI: 10.1007/978-3-030-66498-5_24

[7] T. Shunin, N. Zubkova, and S. Shershakov, "Neural approach to the discovery problem in process mining," in *Analysis of Images, Social Networks and Texts*, 07 2018, pp. 261–273. DOI: 10.1007/978-3-030-11027-7_25

[8] M. Kirchmer and P. Franz, "Value-driven robotic process automation (RPA)," *Business Modeling and Software Design*, vol. 356, pp. 31–46, 2019. DOI: 10.1007/978-3-030-24854-3_3

[9] W. Van der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow patterns," *Distributed and Parallel Databases*, vol. 14, no. 1, pp. 5–51, 2003. DOI: 10.1023/A:1022883727209

[10] N. Russell, A. H. M. Hofstede, W. van der Aalst, and N. Mulyar, "Workflow control-flow patters – A Revised View," *Business*, vol. 2, pp. 06–22, 2006. DOI: 10.1.1.93.6974

[11] N. Russell, W. van der Aalst, and A. Ter Hofstede, *Workflow patterns : the definitive guide*. MIT Press, 2016. DOI: 10.7551/mitpress/8085.001.0001

[12] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. P. van der Aalst, "The prom framework: A new era in process mining tool support," in *Applications and Theory of Petri Nets 2005*, G. Ciardo and P. Darondeau, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 444–454. DOI: 10.1007/11494744_25

[13] A. Berti, S. J. van Zelst, and W. van der Aalst, "Process mining for Python (pm4py): Bridging the gap between process-and data science," in *Proceedings of the ICPM Demo Track 2019, co-located with 1st International Conference on Process Mining (ICPM 2019)*, 2019. DOI: 10.48550/arXiv.1905.06169

[14] K. M. Hanga, Y. Kovalchuk, and M. M. Gaber, "A graph-based approach to interpreting recurrent neural networks in process mining," *IEEE Access*, vol. 8, pp. 172 923–172 938, 2020. DOI: 10.1109/ACCESS.2020.3025999

[15] E. Obodoekwe, X. Fang, and K. Lu, "Convolutional neural networks in process mining and data analytics for prediction accuracy," *Electronics*, vol. 11, no. 14, 2022. DOI: 10.3390/electronics11142128

[16] D. Sommers, V. Menkovski, and D. Fahland, "Process discovery using graph neural networks," in *IEEE International Conference on Process Mining (ICPM)*, 2021. DOI: 10.48550/arXiv.2109.05835

[17] J. Abonyi, R. Károly, and G. Dörgö, "Event-tree based sequence mining using lstm deep-learning model," *Complexity*, vol. 2021, p. 24, 2021. DOI: 10.1155/2021/7887159

Prof. László Kovács from University of Miskolc, Department of Information Technology is a specialist in knowledge modeling and data mining. He obtained a PhD degree in technical sciences from Uni. Miskolc. Main teaching areas: DB Systems, Data Mining and Ontology Management. His research interests involve soft computing, concept set management, heuristic optimizations, ontology modeling, stat. grammar induction and rough set models. He has about 250 publications with 450 references. He is the Head of the Dept. and leader of the Research Group on Machine Learning at Uni. Miskolc.

Erika Baksáné Varga is an associate professor at the Institute of Informatics, University of Miskolc, Hungary. She received a Ph.D. degree in Computer Science from the University of Miskolc in 2011. She has academic experience in teaching procedural and object oriented programming, and data analysis and data mining. Her research interests include teaching methodologies of programming, data and process modeling, data analysis and data mining, ontological modeling, NLP and text mining.

Péter Mileff is a senior SW developer and an associate professor at the Institute of Informatics, University of Miskolc, Hungary. He obtained his Ph.D. in Computer Science in 2008. He has more than 10 years of teaching experience in software development and the building and design of software systems. He has professional and research experiences in computer visualization and game development; and 15 years of industrial experience in the design and development of digital payment systems and platform-independent technologies.