Process Mining of Parallel Sequences with Neural Network Technologies

László Kovács, Erika Baksáné Varga, and Péter Mileff

University of Miskolc, Miskolc-Egyetemváros, H-3515
kovacs@iit.uni-miskolc.hu

Abstract. Process Mining is an important tool for automatic discovery of workflow process schemes. Dominating process mining technologies use either automaton-based engines or neural network engines. The main benefits of the machine learning based methods are the time and scale efficiency, but they have still some limitations considering schema flexibility. The paper introduces a novel approach for mining parallel sequences which is a hard problem for current neural network engines. The performed analysis and test results show that the proposed model is able to induce good quality schema, in many cases in better quality than the base methods

Keywords: process mining, event graph mining, Neural Network classifier

1 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 turn 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).

Thus, the prerequisite for process mining is to have process-related data. As a matter of fact, today's information systems can record every event that occurs during the completion of operational processes. These activities are stored in log files. An event log is basically a structured table where each record corresponds to a sequence of events comprising a process instance. An event is a complex object incorporating several data items that can be automatically captured during the execution of the activity. Event logs may have different forms and instantiations, such as database entries, audit trails, or system logs; and every system that applies some kind of logging mechanism, has developed its own storage scheme. The first approach to standardizing event logs was MXML (Mining eXtensible Markup Language), which stored timestamps, resources, and transactions in a unified format, while other ad-hoc extensions could be added to it as needed. Following that, the first XES (Extensible Event Stream) standard was introduced in 2010 (2). This standard was revised and officially published in 2016 as the 1849-2016 IEEE Standard for eXtensible Event Stream for Achieving Interoperability in Event Logs and Event Streams (3).

Event logs can be used to extract knowledge about processes. A recent research shows that process mining has various real applications (4). In order to guarantee reliable results, process mining should start with high-quality event logs (5). In practice, however, event logs are derived from heterogeneous sources and because of data recording or transformation errors, they are often far from the desired quality (6), (7). For this reason, process mining methods are usually tested on artificially generated logs.

Van der Aalst (1) identifies the following three types of process mining tasks i) process discovery, which produces a process model from an event log without using any a-priori information; ii) conformance checking, where a process model is compared with an event log; and iii) process improvement, where the comparison results in changing the original model.

In our research, we concentrate on process discovery. The general problem is formulated as finding a function that maps an event log onto a process model such that the model is representative for the behavior seen in the event log. More precisely, there are four criteria for such an algorithm (1):

- The discovered model should represent the process instances recorded in the event log (fitness).
- The discovered model should not match processes that are completely unrelated to the cases in the event log (precision).
- The discovered model should generalize the instances in the event log (generalization).
- The discovered model should be as simple as possible (simplicity).

These quality measures, however, are quite challenging to be balanced. For example, an oversimplified model is likely to have low fitness and precision indicators.

An early approach to process discovery was the α-algorithm (1). The α-algorithm works with a bottom-up discovery approach. It scans the event log for frequent patterns based on four ordering relations. More precisely, it takes an event log, parses the log only once and generates a Petri-net as output schema. It creates a transition for each activity in the event log and also adds a start transition and an end transition. Transitions are labeled with the corresponding activity. The α-algorithm can be considered as a baseline algorithm that can be used also for concurrency detection. As the literature shows, it has some shortcomings, especially in handling looping, long-term dependencies, complex concurrency, noise, infrequent and incomplete behavior, and complex routing constructs. On the other hand, due to simplicity, it is widely used as baseline algorithm in comparison with new approaches.

Another induction approach is the inductive mining algorithm 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.

Recently, more advanced techniques have been developed to find frequent patterns in real-life event logs. (8), (9) and (10) are some examples for the application of process mining algorithms in various domains.

However, traditional frequent pattern mining techniques run into their limits when dealing with massive datasets (11). Therefore, more advanced algorithms apply tree-based repesentation (12), (13), (19). These incremental methods derive relevant patterns recursively, which give rise to efficiency problems concerning pattern growth. A solution is to replace the tree with graph representation, which allows for pattern growth along both ends of the detected patterns. This results in fewer levels of recursion and faster pattern growth (20), (21). The search for frequent patterns in sequence data can also be considered as a grammar induction task, so it can be modeled by finite state automata (22), (23).

In our research, we would like to produce models including complex control-flow elements as well, like branching, parallelism and synchronization points. Therefore a tree is not an adequate form to describe the output. Consequently, we extend the previous approaches, so our neural network based solution yields complex event graphs.

2 Tools for Process Mining

In traditional business process management, processes are analyzed through process workshops and interviews, which results in a subjective picture of the ideal process. Process mining takes an entirely different - and far more objective – approach. Process mining uses existing data from corporate information systems and automatically displays a dynamic visualization of the real processes, their performance as well as their compliance.

With more and more companies investing in process mining tools, there are a variety of products available from different vendors. When choosing a process mining tool, it's important to understand how process mining works and to know the features we will need for our company's unique processes. The key capabilities of the appropriate process mining tools can be summarized in the following list (36).

1. Data connectors: Does it provide connectors for all of the underlying IT systems you use, so you can understand process data in real time?
2. Pre-built analyses and models: Does it provide out-of-the-box analyses, models, and benchmarks for your desired use case?
3. Multi-process: Can it work with multiple data models across different processes?
4. Ease of customization: Does it provide a drag-and-drop interface for customizing process analyses and views, or does it require complex knowledge of languages like HTML and SQL to use?
5. Task Mining: Does it collect not only data from transaction systems' event logs, but also user interaction data from desktop applications through task mining?
6. Cloud: Is it a cloud-based process mining tool that can scale with your business and provide automatic updates?
7. Execution Management: Does it go beyond process visibility to provide a complete set of tools for business execution, process improvement, including automation, workflows, and simulation?

8. Security: Is it certified compliant with industry security standards, such as SOC 2 and GDPR?
9. Partner Ecosystem: Have top consultancies adopted it and do they offer implementation services?

2.1 Commercial process mining softwares

There are many options available for the actual implementation of process mining. Larger companies usually choose some kind of out-of-the-box solution, i.e. software that is able to serve the company's needs with a wide range of available functions. Today, very mature software is available for companies using a wide variety of processes. However, their prices/licenses can be high in return. In the next list, we highlight the following leading commercial process mining software tools.

1. **QPR ProcessAnalyzer:** next-gen process mining for large international companies with complex business processes and billions of data rows. QPR ProcessAnalyzer is the first and only process mining solution to run natively on the Snowflake data cloud.
2. **IBM® Process Mining:** a leading process mining solution that automatically discovers, constantly monitors, and continuously optimizes the business processes.
3. **UiPath:** UiPath is a global software company that makes robotic process automation software to automatically discover business processes, and understand where automation will deliver the most value.
4. **Celonis:** Celonis process mining is now the world's highest rated and most popular process mining tool, used by Fortune 500 companies in every industry.
5. **SAP Signavio Process Insights:** is a process analytics solution that helps you rapidly discover areas for improvement and automation within your SAP business processes. The solution then provides recommended actions to implement the necessary changes.

2.2 Open source solutions

Companies that do not want or are unable to pay large sums of money for this type of solution also have many alternatives available for them. In this case, unique solutions for solving process mining are created. These systems are usually implemented by integrating several independent components, which are most often made up of free frameworks, processing, computing and other modules. The most important components of such a system are the following:

1. **Data input / output module:** the component responsible for generating, reading, and transforming data. Each of these modules usually supports multiple inputs (CSV, JSON, XML, XES), but in the case of individual software, they are often very specific. Maybe it is combined with some image or event capturing parts. Their task is to prepare the data for the processing engine.

2. **Processing engine:** usually the hearth of the process mining software having a lot of features. It usually supports automated process discovery based on neural networks and deep learning, creating process maps, analyzing logs and company processes.
3. **Visualization:** a very important module making results more user friendly for humans. It involves several visualization methods, like process map animation, detailed statistics, etc.
4. **Other tools and components:** a customized process mining software usually has several specific tools. E.g.: project management, filters, monitoring, etc.

Currently few developers are open sourcing their solutions since this is an emerging market and open source solutions tend to be more prevalent in mature markets. However, to grow their awareness, proprietary software companies are offering their solutions for free trial which helps users evaluate them.

Some open source process mining tools:

1. **Apromore:** Apromore (Advanced Process Model Repository) is an AI-driven process mining and improvement platform. It combines process mining with task mining with focusing on individual, often manual tasks that occur during almost every process.
2. **Disco:** Disco is one of the first process mining tools which is developed by Fluxicon. Along with automated process discovery, the tool enables filtering, detailed statistics and project management.
3. **Prom:** It is an extensible framework that supports a wide variety of process mining techniques in the form of plug-ins. It is platform independent and distributed in parts, which offers maximal flexibility.
4. **Process Street:** This tool provides business process conformance checking and performance analysis. With Process Street, businesses can create and run multiple instances of workflows. It has a well constructed template allowing tracking the progress and increasing collaboration within the team. Process street has options to sign up for free and pricing for more features.

One of the key issues in the development of unique process mining systems is the available data set. In general, it is not easy to get data of the right quality. Companies are generally unable to publish data because their processes contain sensitive data. The individual creation/generation of data sets is not satisfactory, because they usually do not reach the quality of real data, and in addition, a large volume of data is required. Two important websites are worth mentioning in terms of databases. Several samples and format types (CSV, OCEL, XES) are available on the website of the Process and Data Science Group (29). Large collection of datasets from the Process Discovery Contest held in 2020 are also available (30).

3 Neural Networks for Process Mining

We can consider graph schema induction as a special type of classification problem, where 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. The graph induction method can be modelled as the sequence of elementary prediction steps. As machine learning provides general and very efficient tools for prediction processes, the neural network is a favorite approach also for process schema mining.

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 (14), the RNN network model was applied for the process discovery task. The proposed model is based on the view that each state of the neural network can be interpreted as a state in a finite state machine (FSM) that is implicitly present behind the configuration of the neural network. For representation of the schema graph, a Petri net formalism was applied in the proposed system. For efficiency reasons, the presented method does not regard deriving models with concurrency. The schema graph construction is performed with RNN network using the usual steps, i.e. in each step the network predicts the probabilities of the next possible tokens and chooses the next token from the most probable ones.

If the input gap (dependency gap) is large, the traditional RNN networks with sigma or tanh cells are not suitable for appropriate prediction of the next events. The improved version of RNN, Long Short-Term Memory (LSTM) network introduces novel gate functions into the network architecture and using this mechanism it can handle the problem of long-term dependencies (24), (25). There are many experiments showing the superiority of LSTM, like (**?**), where LSTM was applied for price prediction (26). Another key application field is the prediction of sequence to sequence transformers which was first addressed by Sutskever et al. (27) when using LSTM to improve machine translation. In the literature there are numerous case studies on the application of seq2seq method. Karatzoglou et al. (32) use it to predict future human movement patterns in an effort to improve mobile location based services. Rebane et al. (33) compare the method to traditional models in cryptocurrency prediction. Baumel et al. (34) apply the method to query-focused summarization. Wu et al. (35) integrate IoT data logs and analyze them with a seq2seq based method in order to support the management of IoT systems. An important result on the extension of the baseline model is presented by Abonyi et al. (28), they expanded the algorithm so that its output is a probability tree, that describes the predicted alternative courses of events, instead of a single, most probable sequence.

In recent years, the Convolution Neural Network (CNN) became more and more popular also in the field of process mining. The main benefit of CNN is that it is very effective for the discovery of neighborhood relationships in high dimensional vector spaces. In (15), each trace of the history of event logs is converted into a set of prefix traces, which are then transformed into 2D images. The 2D images generated for the prefix traces of a historical event log are trained using the CNN architecture to generate a deep learning model to predict the next activity in an ongoing process. In the generated image matrix, the number of rows corresponds to the number of events and the number of columns is equal to the number of actions. There are two image matrices, activity matrix and performance matrix. The activity matrix denotes the number of occurrences of a given action in the prefix phase. The second matrix contains the related time duration values. These images are used as inputs for the CNN network to predict the next action in the trace.

Jamshed et al. (31) proposed a combination of convolutional neural network and long short-term memory to infer customer behavior and purchasing patterns in terms of time. CNN is used to reveal the frequent itemsets, and LSTM is used to identify the time interval among these itemsets.

A very recent approach is presented in (16), where a graph convolutional network was used in process mining. The Graph Convolutional Network (GCN) (17) is a generalization of the standard CNN network where the image matrix is substituted with a graph described with an adjacency matrix. The size of the corresponding matrix is $N \cdot A$ where N is the number of nodes and A is the number of attributes (features). In the forward propagation layer, the output is the function of the inputs and the weight values of the adjacency relation. The proposed process mining architecture (16) uses graph convolutional neural networks to translate a given input event log into a sound Petri net. The performed tests show that training on synthetically generated datasets allows the network to translate previously unseen synthetic and several real-life event logs into sound, arbitrarily structured models with comparable accuracy and simplicity as existing state of the art techniques.

Considering the main challenges in the application of machine learning tools in process mining (1), we can highlight the following problem domains:

- Inter-case predictions, where the prediction depends not only on the traces in the log files, but there are information sources from external sources
- Explainable predictions: Understanding the rationale behind predictions would certainly help users decide when to trust or not to trust them. Tools of Explainable Artificial Intelligence should be incorporated into the process mining methods.
- Predictions with a-priori knowledge: In many real-life situations, cases exist in which, together with past execution data, some case-specific additional knowledge (a-priori knowledge) about the future is available.
- Prescriptive process monitoring which supports long-term decision making at strategy level.
- Discovery of complex patterns. Current neural network methods are able to discover only sequence level patterns, new approaches are needed to extend the functionality of the neural networks on event graphs.

4 Mining parallel processes with neural networks

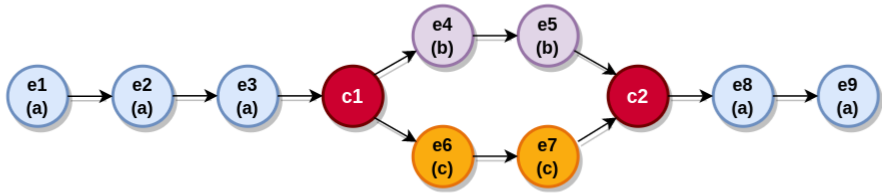
In our approach, parallelism denotes parallel workflows of different actors and resources, and the split and join control nodes denote an artifact level dependency among the different branches. For example, we consider a workflow to produce a mobile phone. In this process the production of the different 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, too. 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 of different actors. 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.

During the training phase, there is a preprocessing module where the engine determines the related synchronization nodes first and a graph of synchronization nodes is constructed. The edges between the nodes correspond to event sequences performed by a single agent. At the bottom level, the models for these sequences are generated where each model corresponds to a dedicated agent. For the agent level sequences, the prefix part consists of all events before the synchronization event independently from the active agent. Thus the global event schema is decomposed into several schema instances. There is a top level schema graph with synchronization nodes where the links represent complete sequences performed by a single agent. Beside this schema, there are agent level schema graphs, each agent is assigned to a dedicated schema graph where the nodes are atomic events.

Example 1. Let us take the following input graph given in Fig. 1. The conversion

Fig. 1: Schema graph

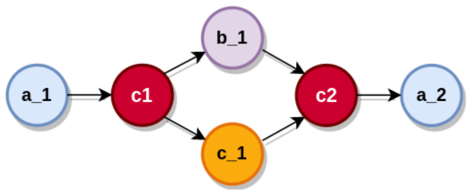


into synchronization level graph results in the graph shown in Fig. 2.

During the training process, more models are generated. There is a model for the top level synchronization schema ($model_{top}$) and there are separate models for the different agents ($models_{agent}$).

The process of schema graph prediction is decomposed into the predictions of different level sequences and the graph is constructed from these component sequences. The graph prediction algorithm performs the following steps:

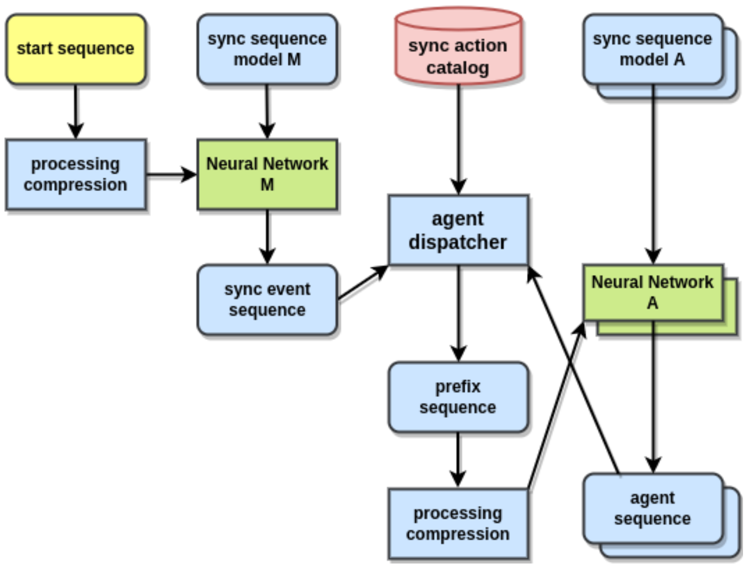
Fig. 2: Generated top level synchronization graph



1. Prediction of the top level synchronization sequence using $model_{top}$.
2. Construction of the top level schema graph.
3. Prediction of the agent level event sequences for every edge in the top level graph.
4. Merging the component schema sequences into a global schema graph.

The prediction of the event graph requires the ensemble of sequence prediction modules as the different agents cooperating in the event graph may have very different processing models. Thus we use different neural network units to model the different agents. Beside the actor level models, a main level model, the level of synchronization events is also included into the framework. The proposed architecture is presented in Fig. 3.

Fig. 3: Architecture of the NN-based prediction process



For the evaluation of the proposed method, we performed test experiments with some baseline event logs of small size. The main goal of these tests were to verify the accuracy of the schema induction and compare the quality of the generated schema with other schema results using some other base methods. The test results show that the proposed model is able to induce good quality schema, in many cases in better quality than the base methods as it is demonstrated in Fig 4 and Fig 5. In the test, the α-algorithm and the proposed engine were trained with the same event log data. The target schema used for the generation of the event log is given in Fig. 6. It can be seen that similarity between the target schema and the output schema of the proposed method is higher than the similarity with the baseline α-algorithm.

Fig. 4: Schema graph generated with the proposed engine

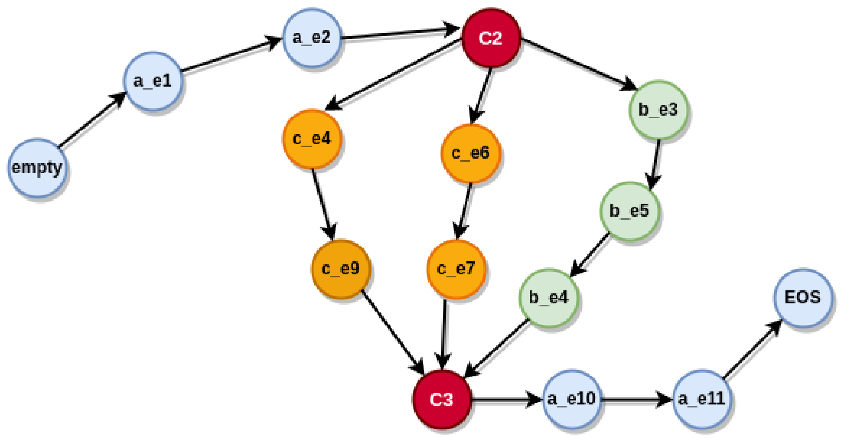


Fig. 5: Schema graph generated with the α-algorithm

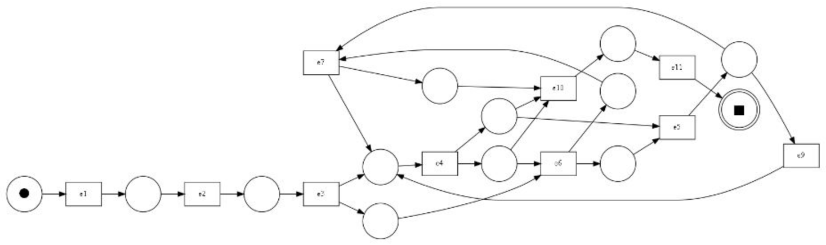
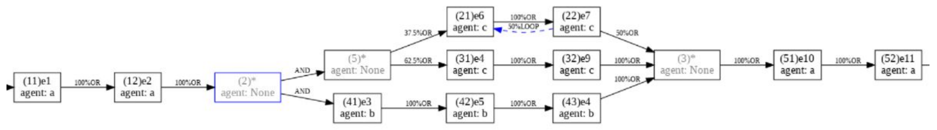


Fig. 6: Target workflow schema



5 Conclusions

The adaption of neural network tools in process mining and schema induction tasks is a actively investigated research area. Current neural network tools provide high speed and good generalization but there are still some limitations in induction capability. Such shortcoming is the processing of parallel sequences or loop section inside the schema graph. The model presented in this paper focuses on the detection of parallel sequences. The proposed engine uses a two-level analysis using a synchronization top level and an agent-based level. The related architecture consists of more neural network units devoted to the different components. The performed tests show that the engine is able to perform a flexible schema induction in many cases in better quality than the baseline schema detection methods.

Bibliography

[1] van der Aalst, W. M. P. (2016). Process Mining: Data Science in Action. Springer, Heidelberg . doi: 10.1007/978-3-662-49851-4

[2] Günther, C. W., Verbeek, E.: XES standard definition. xes-standard.org/_media/xes/xesstandarddefinition-2.0.pdf

[3] IEEE standard for extensible event stream (XES) for achieving interoperability in event logs and event streams. IEEE Std 1849-2016, pp. 1-50.

[4] Emamjome, F., Andrews, R. and ter Hofstede, A. H. (2019). A case study lens on process mining in practice. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 127-145). Springer, Cham.

[5] Andrews, R., van Dun, C. G., Wynn, M. T., Kratsch, W., Röglinger, M. K. E. and ter Hofstede, A. H. (2020). Quality-informed semi-automated event log generation for process mining. Decision Support Systems, 132, 113265.

[6] Suriadi, S., Andrews, R., ter Hofstede, A. H. and Wynn, M. T. (2017). Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. Information systems, 64, 132-150.

[7] Fischer, D. A., Goel, K., Andrews, R., Dun, C. G. J. V., Wynn, M. T. and Röglinger, M. (2020). Enhancing event log quality: Detecting and quantifying timestamp imperfections. In International Conference on Business Process Management (pp. 309-326). Springer, Cham

[8] Weiss, G. (2002). Predicting telecommunication equipment failures from sequences of network alarms. Handbook of Knowledge Discovery and Data Mining, 891-896.

[9] Kinnebrew, J. S. and Biswas, G. (2012). Identifying Learning Behaviors by Contextualizing Differential Sequence Mining with Action Features and Performance Evolution. International Educational Data Mining Society.

[10] Csalódi, R. and Abonyi, J. (2021). Integrated survival analysis and frequent pattern mining for course failure-based prediction of Student dropout. Mathematics, 9(5), 463.

[11] Truong-Chi, T., Fournier-Viger, P. (2019).High-Utility Pattern Mining: Theory, Algorithms and Applications. In: A Survey of High Utility Sequential Pattern Mining, pp. 97-129. Springer, Cham

[12] Liu, J., Yan, S., Wang, Y. and Ren, J. (2012). Incremental mining algorithm of sequential patterns based on sequence tree. In Advances in Intelligent Systems (pp. 61-67). Springer, Berlin, Heidelberg.

[13] Rizvee, R. A., Arefin, M. F. and Ahmed, C. F. (2020). Tree-miner: mining sequential patterns from sp-tree. In Pacific-Asia conference on knowledge discovery and data mining (pp. 44-56). Springer, Cham.

[14] Shunin, T., Zubkova, N. and Shershakov, S. (2018). Neural approach to the discovery problem in process mining. In International Conference on Analysis of Images, Social Networks and Texts (pp. 261-273). Springer, Cham.

[15] Obodoekwe, E., Fang, X. and Lu, K. (2022). Convolutional Neural Networks in Process Mining and Data Analytics for Prediction Accuracy. Electronics, 11(14), 2128.

[16] Sommers, D., Menkovski, V.and Fahland, D. (2021). Process discovery using graph neural networks. In 2021 3rd International Conference on Process Mining (ICPM) (pp. 40-47). IEEE.

[17] Welling, M. and Kipf, T. N. (2016). Semi-supervised classification with graph convolutional networks. In J. International Conference on Learning Representations .

[18] Di Francescomarino, C. and Ghidini, C. (2022). Predictive process monitoring. Process Mining Handbook. LNBIP, 448, 320-346.

[19] Liu, X., Zheng, L., Zhang, W., Zhou, J., Cao, S. and Yu, S. (2022). An evolutive frequent pattern tree-based incremental knowledge discovery algorithm. ACM Transactions on Management Information Systems (TMIS), 13(3), 1-20.

[20] Singh, D. K., Sharma, V.and Sharma, S. (2012). Graph based approach for mining frequent sequential access patterns of web pages. International Journal of Computer Applications, 40(10), 33-37.

[21] Dong, W., Lee, E. W., Hertzberg, V. S., Simpson, R. L. and Ho, J. C. (2021). GASP: Graph-Based Approximate Sequential Pattern Mining for Electronic Health Records. In European Conference on Advances in Databases and Information Systems (pp. 50-60). Springer, Cham.

[22] Hingston, P. (2002). Using finite state automata for sequence mining. In ACSC (pp. 105-110).

[23] Jacquemont, S., Jacquenet, F. and Sebban, M. (2009). Mining probabilistic automata: a statistical view of sequential pattern mining. Machine Learning, 75(1), 91-127.

[24] Gers, F. A., Schmidhuber, J. and Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. Neural computation, 12(10), 2451-2471.

[25] Gers, F. A. and Schmidhuber, E. (2001). LSTM recurrent networks learn simple context-free and context-sensitive languages. IEEE transactions on neural networks, 12(6), 1333-1340.

[26] Ji, S., Kim, J. and Im, H. (2019). A comparative study of bitcoin price prediction using deep learning. Mathematics, 7(10), 898.

[27] Sutskever, I., Vinyals, O. and Le, Q. V. (2014). Sequence to sequence learning with neural networks. Advances in neural information processing systems, 27.

[28] Abonyi, J., Károly, R. and Dörgö, G. (2021). Event-Tree Based Sequence Mining Using LSTM Deep-Learning Model. Complexity, 2021

[29] Process Mining Datasets: http://www.processmining.org/event-data.ht (2022)

[30] ICPM conference website, 2020 edition of the Process Discovery Contest: https://icpmconference.org/2020/process-discovery-contest/downloads, 2022

[31] Jamshed, A., Mallick, B. and Kumar, P. (2020). Deep learning-based sequential pattern mining for progressive database. Soft Computing, 24(22), 17233-17246.

[32] Karatzoglou, A., Jablonski, A. and Beigl, M. (2018, November). A Seq2Seq learning approach for modeling semantic trajectories and predicting the next location. In Proceedings of the 26th acm sigspatial international conference on advances in geographic information systems (pp. 528-531).

[33] Rebane, J., Karlsson, I., Papapetrou, P. and Denic, S. (2018). Seq2Seq RNNs and ARIMA models for cryptocurrency prediction: A comparative study. In SIGKDD Fintech'18, London, UK, August 19-23, 2018.

[34] T. Baumel, M. Eyal, M. Elhadad (2018). Query Focused Abstractive Summarization: Incorporating Query Relevance, Multi-Document Cover- age, and Summary Length Constraints into seq2seq Models, ArXiv 1801.07704v2

[35] Wu, P., Lu, Z., Zhou, Q., Lei, Z., Li, X., Qiu, M. and Hung, P. C. (2019). Bigdata logs analysis based on seq2seq networks for cognitive Internet of Things. Future Generation Computer Systems, 90, 477-488.

[36] Celonis Process Mining Software, www.celonis.com/process-mining/