Using Recurrent Neural Networks to Predict Future Events in a Case with Application to Cyber Security

Stephen Jacob
Dept. of Electronics and Informatics
Athlone Institute of Technology
Athlone, Ireland
Email: s.jacob@research.ait.ie

Yuansong Qiao
Dept. of Electronics and Informatics
Athlone Institute of Technology
Athlone, Ireland
Email: ysqiao@research.ait.ie

Paul Jacob
Dept. of Electronics and Informatics
Athlone Institute of Technology
Athlone, Ireland
Email: pjacob@ait.ie

Brian Lee
Dept. of Electronics and Informatics
Athlone Institute of Technology
Athlone, Ireland
Email: blee@ait.ie

***Abstract*—Due to the number of cyber attacks targeting business organisations daily, anomaly detection software generates large numbers of alerts. While this information is invaluable to Incident Response Teams, one problem is to prioritize these alerts and to distinguish between those that signal a serious threat to network enterprises and low priority alerts. One approach is to use a model that relates an organisation's missions, processes, services and infrastructure. By predicting future events in existing business processes, and subsequently using this model to identify associated services and infrastructure, cyber security personnel can prioritize critical alerts that threaten these assets. Long Short Term Memory based deep learning models are suited to modeling sequential data, and in particular can model long term dependencies in sequences. This paper evaluates the use of such models to predict subsequent events in ongoing cases. Two training techniques are applied to four data sets. The techniques are evaluated with respect to the accuracy of the predictions and their performance on predicting frequent and infrequent events.**

Keywords-Process Mining; Deep Learning; Recurrent Neural Networks; LSTM; Cyber Security.

I. Introduction

Most business organisations are constantly targeted by cyber security attacks. Anomaly detection software generates huge volumes of alerts and Computer Security Incidence Response Teams (CSIRT) struggle to follow up on all of these alerts. They require means of distinguishing alerts that signal attacks on critical business processes from low-priority alerts [1].

One way to do this is to predict future events in currently executing business processes and with the aid of a mission dependency model as outlined in Section II, identify critical services and infrastructure in the organisation. Security alerts, which target these critical services and assets can then be prioritized for the attention of the CSIRTs.

The main aim of this paper is to investigate the application of deep learning to process mining as a means to indicate likely high priority security events. The objective is to use Recurrent Neural Networks (RNN), in particular Long Short Term Memory (LSTM) networks, to model event traces with a view to using the resulting model to predict future events. The use of process mining for cyber security attack and anomaly detection has been demonstrated by the work of Mauser et al. [2], Alvarenga et al. [3] and van der Aalst et al. [4]. Process mining techniques have also been used for visualisation of cyber attacks [1]. Recent research undertaken by Tax et al. [5] and Evermann et al. [6] highlights that using deep learning applications to model and predict process sequences is an effective and increasingly popular approach.

In this paper, we investigate two methods to train LSTM networks, which model ongoing business processes. We evaluate both methods and determine which one is the more effective at training an LSTM network to predict subsequent events. The first method generates prefixes from every sequence in the data and trains the network to predict the next event from these prefixes. The second method, Teacher Forcing [7], trains the network at every time step as a case/sequence of events is passed through the network. Both these methods are applied to four data sets, the Business Process Intelligence (BPI) challenges from [8]–[10], and the Helpdesk data set used as supplementary material for Tax et al. [5].

The paper is structured as follows: Section II explores previous related approaches to the use of deep learning to monitor process sequences. Section III provides relevant background information in Mission Dependency Modeling, Process Mining and RNN/LSTM Neural Networks. Section IV describes the two approaches to training a LSTM network mentioned above. Section V outlines the experimental setup and evaluation. Section VI presents the results obtained from the experiment. Section VII is the conclusion.

II. Related Work

Alvarenga et al., [1] addressed the concept of alert correlation, as well as the issue that an Intrusion Detection System (IDS) produces an unmanageable amount of alerts and most are low-level annoying alerts incorrectly categorized as malicious. The overwhelming number of alerts results in keeping network administrators from responding appropriately to the more critical attack forms used by cyber attackers. Alvarenga

 ISBN: 978-1-61208-785-6

proposed a process mining method to produce process models to assist administrators to identify and investigate multistage cyber attacks.

Alvarenga et al., [3] carried out a study to discover cyber attack strategies targeting networks using traditional process mining and the open-source process mining framework ProM [11]. A data log is generated and taken from an IDS deployed at University of Maryland. The data was loaded into the ProM framework and a process model was generated to visualize the process paths extracted from the data set. Further analysis identified the causal dependencies between events by comparing different cases of sequential events in the model. A benefit of this was that the discovered model could be used to visualize alerts consistent with that of a cyber attacker's perspective when attempting to compromise a targeted network.

Mauser et al., [2] used process mining discovery to detect and identify cyber security attacks on enterprise systems. Mauser identified cyber attacks by detecting anomalies in process executions in a software system. By visualizing said execution paths using Petri Nets, irregular processes paths could be isolated by comparing them to a model of regular process activity.

Tax et al., [5] applied LSTM networks to the BPI 2012 and Helpdesk data sets to learn from predicting both the subsequent event and the time until the next event. They evaluate a number of different neural network architectures with two or more network layers ranging from completely separate networks to predict activity and time to various combinations of shared and specific layers for both predictions. The method used in this paper to train the neural network is the prefix method. We re-implement this method and also implement the alternative teacher forcing method, which results in substantial improvements in training times.

Evermann et al., [6] also used an LSTM network to predict future events in a case. Evermann's approach is motivated by identifying the associated resources for an event and detecting the long-lasting dependencies within cases to subsequently predict future events. Associated resources for an event include the duration of an event, and the related resources (personnel, attributes) assigned to them. Evermann applied this approach to the BPI 2012 and 2013 data sets [8][9]. In addition, we make predictions for the BPI 2014 data set [10]. To the best of our knowledge, event prediction has not been previously applied to this data set.

III. Background Information

In this section, we first describe the mission dependency metamodel we use in our machine learning approach. We then provide an overview of the technologies used: process mining, neural networks, recurrent neural networks and LSTM models.

A. Mission Dependency Metamodel

Mission dependency modelling is a technique used as part of cyber risk assessment. This model makes explicit the relationships between mission objectives, business processes, IT services and computing assets of an organisation.

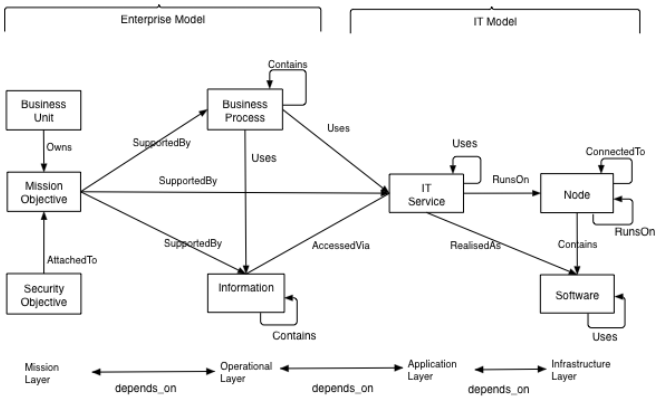

Figure 1. Mission Dependency Metamodel.

One such approach is the dependency model shown in Figure 1 that was introduced in [12]. The four layers in the model are the Mission, Operation, Application and the Infrastructure Layers. We are focused on the Operation Layer and are interested in predicting future events in currently executing business processes. This information when mapped through the dependency graph to the underlying layers can be used to identify critical services and infrastructure that may be liable to attack in the short term arising from the current cyber security situation. This, in turn, can help security response teams to prioritize security alerts in such a way as to best protect critical processes in an organisation.

B. Process Mining

Process mining can be collectively defined as the analysis, discovery and modeling of information extracted from process data sets [2][13]. These data sets are comprised of cases, which are process execution paths or sequences of events. Traditional business process mining can discover process models from event data using, e.g., the Alpha algorithm [14]. Graphical models can be generated and observed using a variety of tools including the open-source framework ProM. Once a model exists, conformance checking can be carried out to determine if logged cases conform to the model. Other insights include the ability to audit and analyze the data process, as well as how to improve it.

C. Neural Networks

An alternative to the traditional process mining approach described above is to train a neural network model to learn the behaviour of the event sequences, then use the trained model to make predictions [6]. Neural networks are trained using a set of data as follows. When a neural network outputs a value in response to some input, this predicted output value is then compared with the actual output value in the data. A loss function is defined as the function for the difference between the predicted and actual values. An algorithm called back-propagation is used to minimize this loss value.

 ISBN: 978-1-61208-785-6

A Dense layer in a neural network is a layer where all the nodes are connected to all the nodes in the previous layer. Normally, the output layer of a neural network will be a Dense layer. In the case of a regression problem, with numerical output, each node in the output layer outputs a numeric value. For the case of classification, each node will correspond to a different class, and the softmax activation function is used to convert every numerical output to a probability of that class occurring. In effect, the output layer outputs a probability distribution vector over the number of different classes. The loss function used for (non-binary) classification is known as the categorical-crossentropy loss function.

D. Recurrent Neural Networks (RNN)

When processing sequences using a neural network, the sequence is fed into the network over a number of time-steps. The network is required to learn sequential behaviour, how events at one time-step affect the subsequent events in the sequence. An RNN is a neural network where the output from the hidden layer is fed back into the hidden layer on the subsequent time step as shown in Figure 2. In this way, the occurrence of particular events in a sequence can affect the likelihood of other events occurring later in the sequence.

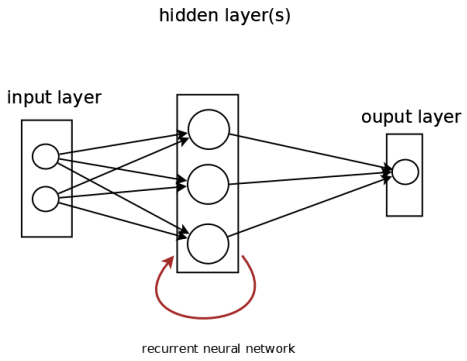

Figure 2. RNN with a single hidden layer.

In order to train the RNN, the required input is a multi-dimensional array of the shape (sequences, time-steps, features). The length of the first dimension is the number of sequences, or cases, to train. The length of the second dimension is the number of time steps within a case of chronological ordered events. When the input is a categorical variable, (an event type in this project), this categorical variable is one-hot encoded and the length of the third input dimension is equal to the total number of event types.

The shape of an RNN network is normally the size of a probability distribution over unique event types for every sequence. Note that it is possible to configure a network to output a prediction at each time step of a sequence. In that case, the output is of the shape (sequences, time-steps, features) and is essentially a sequence of predictions/probability distributions, one for every time step in every sequence. In Keras, this is achieved by including a TimeDistributed wrapper layer around the Dense layer that produces a prediction at every time slice. This is used in the teacher forcing method outlined in Section IV below.

E. Long Short Term Memory (LSTM)

A LSTM model is a RNN model that supports long-term dependencies in noisy, sequential data [15]. LSTM nodes are no longer simple single nodes but rather a sub-network of other nodes and activation functions. Long term dependencies are captured using three **gates** in an LSTM node, an input gate, a forget gate and an output gate. These gates control how data from a previous time step is used, stored or thrown away. The forget gate determines which data is to be discarded. The output gate determines the output based on previous input and the state of the LSTM node. Event sequences in process mining data can contain long term dependencies and hence LSTM networks are useful for modeling such data.

IV. Model Training Methods

We examine two approaches taken when training an LSTM model to predict future events within a case, the prefix method and the teacher forcing method.

A. The Prefix Method

This approach [5] generates a set of all possible prefixes longer than the length of a single event from all sequences to train the model. For example, for the sequence of event types 1,2,3,4,5 the input and output is shown in Table I.

TABLE I. Input Prefixes and Output.

X (input)	*y (target)*
[1, 2]	3
[1, 2, 3]	4
[1, 2, 3, 4]	5
[1, 2, 3, 4, 5]	!

Effectively, the network is trained to predict a target value *y* given the input value *X*. Note that the model is trained to predict a *!* character, which denotes the completion of a case. The trained model is then used to predict a single suffix event following the prefix. Following the work of Tax et al. [5] the shortest prefix used is of length two, so predictions start after the second event.

The architecture for the model is shown in Figure 3.

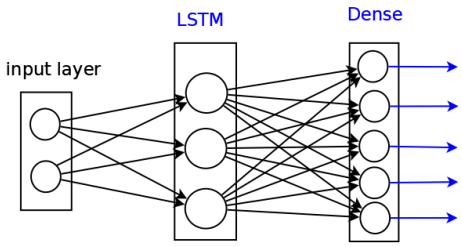

Figure 3. LSTM network with a Dense output layer.

B. Teacher Forcing Method

Teacher forcing uses the ground truth from a prior time step in a sequence as input [7][16]. For the sequence 1,2,3,4,5, the input and output for the neural network are shown in Table II.

TABLE II. INPUT AND OUTPUT FOR TEACHER FORCING.

X (input)	*y (target)*
[1, 2, 3, 4, 5]	[2, 3, 4, 5, !]

Notice that input sequence *X* is the event sequence with the last event removed and output *y* is the sequence with the first removed. For training, *X* is passed through the LSTM network once. The input at each of these time steps is the ground truth value at the previous time step. For example, while predicting 2 the input is 1, and so on.

The architecture of the model is as shown in Figure 3 except that the output Dense layer has a TimeDistributed wrapper layer. The existence of a TimeDistributed layer in the network distributes the output Dense layer with the softmax activation function to every time step in a sequence, resulting in a prediction at every time step. Note that training can occur on one pass of the sequence through the network, as opposed to a pass for every prefix of a sequence as with the prefix method.

V. EXPERIMENT AND EVALUATION

In this section we introduce the data sets used. We then outline the model training and evaluation approaches for both the prefix method and the teacher forcing method.

A. Data Sets

The above techniques are applied to four data sets used in the project, the BPI challenges from 2012, 2013, 2014, and the Helpdesk data set used by [5].

1) BPI 2012 Data Set: The BPI 2012 data set comes from the BPI challenge workshop in 2012 [8]. The data set is an event log taken from an application procedure for financial services, such as a personal loan or overdraft, at a large Dutch financial institution. Originally comprised of several sub-processes, the event log is narrowed down to the 'work item' sub-process and only cases with the work item events types *start* and *complete* were included. This reduction of the sub-processes has been previously used by [5] and [17].

The resulting event log contains a vocabulary of 6 event types and 7469 cases. Each event type is defined by an *Activity ID* in the data. Each case, or process sequence of *Activity ID*s are identified and grouped by their *Case ID*, which acts as a unique case identifier.

2) Helpdesk Data Set: The Helpdesk data set is an event log from a ticket management process for an Italian software company's help desk. Tax et al., [5] used this data set as supplementary material. The log consists of 9 different event types, 3804 cases and 13710 events. The different event types are represented by their *Activity ID* with the *Case ID* being the unique case identifier. The list of cases are returned by grouping all *ActivityID*s by their respective *CaseID*s.

3) BPI 2013 Data Set: The BPI data set for 2013, provided by the BPI 2013 workshop [9], is an event log for an incident management system called VINST. VINST solves IT related problems for Volvo Information Technology. Each problem or IT service request made to VINST is treated as a case with the Service Request number being the case identifier. The resulting event log was provided by Volvo IT Belgium and lists 7553 cases and the designated case identifier is the column labeled *SR Number*. For both this research project and [18], the vocabulary of event types are defined by generating every unique possible combination of the two columns *Status* and *Sub Status*, returning a vocabulary of 13 event types. Every event type is then mapped to an event number. Each different trace of event numbers is then grouped by the *SR Number* to return the data set.

4) BPI 2014 Data Set: The BPI 2014 data set is an event log selected from a collection of three different processes investigated by the ICT department for Rabobank Group, a banking and financial services company. A service management tool logs customer support calls for software support. The service management tool logs three main sub-processes, which are outlined below and are provided in CSV by Rabobank.

- *Interactions*: calls are made by customers (Rabobank colleagues) to the Service Desk where a Service Desk Agent (SDA) answers, resolves the issue for the customer and logs these calls as an *Interaction* or assigns the technical issue to an Assignment Group
- *Incidents*: the SDA is unable to resolve a customer call, and based on a given urgency and impact, assigns the issue to an Assignment Group to solve and the process to solve the issue is logged by Rabobank as an *Incident*; each incident is treated as a case of logged activities an assignment group takes to resolve said disruption
- *Changes*: if a service disruption were to occur more than once then a problem analysis investigation is launched that will lead to an improvement plan to prevent the service disruption from happening again subsequently logging a *Change* record

The primary sub-process investigated by Rabobank selected for this research is the *Incident* data. The actual data set used is a translated event log built from csv files relating to every incident. The files downloaded from the BPI 2014 workshop for this project are: *Detail Incident.csv*, a list of 46607 unique incidents, and the *Detail Incident Activity.csv* file, an activity log of recorded events related to 46605 incidents in the list of incidents. For each individual incident, the column *IncidentID* is the designated case identifier. To define the vocabulary of unique events for the data set, the columns *Category* and *IncidentActivity-Type* are selected from the list of cases and incident activity log respectively. The two files are then merged into a new singular event log using the *IncidentID* column as a joining key. The two aforementioned columns *Category* and *IncidentActivity-Type* are now both in the same event log. The vocabulary of different event types can now be now defined using every possible unique combination of the two columns.

 ISBN: 978-1-61208-785-6

This returns a vocabulary of 91 event types in total. Each trace of events is then grouped by the column *IncidentID* to return the data set of incidents, or cases.

B. Model Training and Evaluation for the Prefix Method

All the training prefixes are generated from the data set as outlined in Section IV above. The prefixes are pre-padded to the length of the longest case. Training parameters include batch size and the number of epochs. Finally, 20% of the training data is set aside for validation purposes allowing us to see the behaviour of the training/validation and accuracy/loss values at the end of each epoch.

When evaluating the trained model, prefixes are generated for every case in the test data set. Each prefix sequence generated is passed through the trained network and the model outputs the probability distribution vector over the number of different event types. To determine the predicted event, the index of the largest value in the probability vector is found and its respective event type is returned. The accuracy of the model's performance is found by comparing the predicted event type with the actual event type for every testing prefix.

C. Model Training and Evaluation for the Teacher Forcing Method

The training sequences are sorted in increasing size and the data is divided into mini-batches of a chosen size. The model training method is then called on each of these mini-batches. All sequences in a batch must be of the same length, so all sequences are pre-padded with zeros to the size of the longest sequence in the mini-batch. Mini-batches will typically have different lengths.

To evaluate the model using the Teacher Forcing method, each sequence in the testing data is fed through the network. The output is a sequence of probability distributions corresponding to a prediction for every time step. Accuracy is evaluated by comparing predicted events with the subsequent event in the input sequence starting after the second event.

VI. Results

Having built an LSTM based model, a range of parameter values were evaluated to find the optimum configuration of meta-parameters for the model. Table III gives model configurations and prediction accuracy for the BPI 2012 data using the Prefix version. (Other tables for the other data sets and Teacher Forcing method are not included in the paper.) Notice that the use of a second LSTM layer or adding a Dropout layer for regularization did not improve the accuracy.

These models used an Adam optimizer [19], which is efficient and requires minimal memory and parameter tuning, and works well with cases comprised of noisy data. The optimizer also uses an adaptive learning rate, a hyper-parameter that controls the step size at each iteration of the training algorithm. It is a trade off between reaching an optimal solution in a timely manner, and overshooting the optimal solution.

The maximum accuracy values for each data set are listed in Table IV.

TABLE III. Prefix Method on BPI 2012 Data Set.

LSTM	Dropout	Nodes	Batch Size	Epochs	**Accuracy**
1	0	100	10	50	65.68%
1	1	100	10	50	66.31%
2	0	100	10	20	66.34%
1	0	100	6	20	66.60%
1	0	120	32	20	67.73%
1	0	60	32	20	67.88%
1	**0**	**100**	**32**	**20**	**68.64%**

TABLE IV. Maximum Accuracy for the Data Sets.

Data Set	Cases	Events	**Max Acc.**
BPI 2012	7469	6	68.64%
Helpdesk	3803	9	81.16%
BPI 2013	7553	13	65.66%
BPI 2014	6000	69	48.28%

As expected, it is harder to make predictions for data sets with a larger number of event types. The Helpdesk data set seems to be the exception with a vocabulary of 9 different event types and the highest accuracy of 81%. We looked at the frequency distribution for event types, including the end of case character *!* in this data set. A graph of the frequency distribution is shown below in Figure 4.

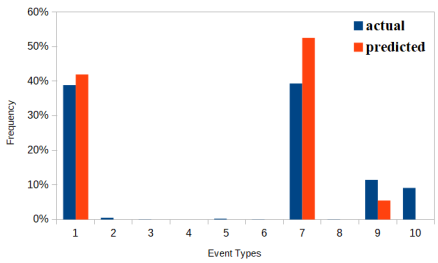

Figure 4. Frequency Distribution of Helpdesk Events.

Six event types are quite infrequent compared to the others and we could say the effective number of event types is four, including the end of case event, and this explains the high accuracy. The Teacher Forcing method makes no predictions for the other infrequent event types, while the Prefix method makes 5 predictions of these infrequent events out of a total of 1267 predictions. Notice that both models over predict the more frequent events while under predicting those rarely appearing. This is to be expected as the neural network has not seen enough of these infrequent events to learn how to predict them.

A similar situation holds for the 2013 data set where there are five infrequent event types. Figure 5 shows the frequency distributions for the actual event types in the data set and the predicted events given by the two methods. Notice that again frequent events tend to be over predicted and infrequent events tend to be under predicted. Note that the Prefix version is also less prone to this bias than the Teacher Forcing version.

Table V displays the accuracy and execution time for each of the four different data sets and two training methods.

 ISBN: 978-1-61208-785-6

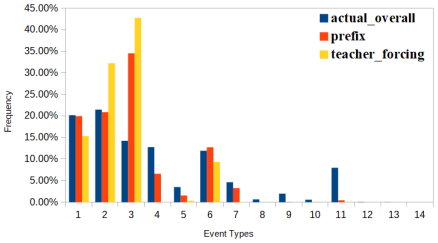

Figure 5. BPI 2013 Frequency Distribution of Test Events.

TABLE V. PREFIX AND TEACHER FORCING COMPARISON.

Data Set	Prefix Method		Teacher Forcing Method	
	Acc.	Time (mins)	Acc.	Time (mins)
BPI 2012	68.64%	17.9	68.18%	0.5
Helpdesk	81.16%	2.7	80.39%	0.97
BPI 2013	65.66%	32.1	62.94%	0.5
BPI 2014	48.28%	42.5	43.68%	0.5

The teacher forcing method is much faster to train. As shown in Table I for a case of length n, n-2 prefixes are generated for the prefix method. So, roughly speaking the number of training instances for the prefix method is an order of n times larger and this accounts for the substantial difference in training times. As shown, both techniques generally produce similar results for the 2012 and Helpdesk data sets. The prefix method produces better results for the 2013 and 2014 data sets. This is surprising as the loss function is the same. Further work is required to understand why this is so.

The full BPI 2014 data set contains 46606 cases. Training using the full data set was carried out on an NVIDIA GPU server with a four-card Tesla SXM2. For the Prefix method, using the full data set resulted in a 49.49% accuracy and took an hour and 7 minutes to train.

VII. CONCLUSION AND FUTURE WORK

CSIRTs constantly struggle to attend to the large number of alerts generated by intrusion detection software. One approach to this is to identify critical services and assets in the organisations that are being targeted. If suitable models exist linking business processes and supporting infrastructure, the ability to predict the next case activities can support CSIRTs in prioritizing the examination of intrusion alerts.

This paper evaluates the use of LSTM neural networks for predicting next activities in a case. In particular it looked at four data sets and two training methods. Prediction accuracy for the different data sets depends on, to a large extent, the number of event types in each data set. As expected, it is harder to predict event types where there is a large number of them. Also models tend to under predict rare events and over predict common events. This bias was more pronounced when the model was trained using the teacher forcing method.

The prediction accuracy for the Helpdesk data set was the highest at 81.2% for nine event types. This is high compared to the BPI 2012 data set, which only had six event types and an accuracy of 68.6%. However, five event types from the Helpdesk data set had very low frequency resulting in the model mostly choosing between four different event types. This explains the higher accuracy obtained.

Comparing the two methods of training we saw that in two cases the accuracy was nearly the same (within 1%). In the other two the prefix method was slightly higher, 2.8% better for the BPI 2013 data set and 4.6% for the BPI 2014 data set. Even though the differences were small this was slightly surprising as we expected the results to be the same. In all cases, the teacher forcing method takes an appreciably shorter time to train, by a factor of up to six times faster.

To the best of our knowledge, LSTM networks have not been previously applied to event prediction for the 2014 data set. For the Helpdesk data set, our accuracy results were 10% better than published by Tax et al. [5]. For the BPI 2012 data set our results were 8% lower. It should be stressed that we are not comparing our means of using the timestamp and event types as input for the LSTM model to Tax's method.

REFERENCES

[1] S. C. De Alvarenga, S. Barbon Jr, R. S. Miani, M. Cukier, and B. B. Zarpelão, "Process mining and hierarchical clustering to help intrusion alert visualization," *Computers & Security*, vol. 73, pp. 474–491, Mar. 2018.

[2] S. Mauser and T. Eggendorfer, "Detecting security attacks by process mining," *Algorithms and Tools for Petri Nets*, pp. 1–33, Oct. 2017.

[3] S. C. de Alvarenga, B. B. Zarpelão, S. Barbon Jr, R. S. Miani, and M. Cukier, "Discovering attack strategies using process mining," *AICT*, vol. 15, pp. 119–125, 2015, ISBN: 978-1-61208-411-4.

[4] W. M. Van der Aalst and A. K. A. de Medeiros, "Process mining and security: Detecting anomalous process executions and checking process conformance," *Electronic Notes in Theoretical Computer Science*, vol. 121, pp. 3–21, Feb. 2005.

[5] N. Tax, I. Verenich, M. La Rosa, and M. Dumas, "Predictive business process monitoring with lstm neural networks," in *International Conference on Advanced Information Systems Engineering*. Springer, Jun. 2017, pp. 477–492.

[6] J. Evermann, J.-R. Rehse, and P. Fettke, "Predicting process behaviour using deep learning," *Decision Support Systems*, vol. 100, pp. 129–140, Aug. 2017.

[7] K. Drossos, S. Gharib, P. Magron, and T. Virtanen, "Language modelling for sound event detection with teacher forcing and scheduled sampling," *arXiv preprint arXiv:1907.08506*, Jul. 2019.

[8] B. van Dongen, "Event log of a loan application process," Eindhoven University of Technology, 2012, URL: https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f/ [accessed: 2012-04-23].

[9] W. Steeman, "Bpi challenge 2013, incidents (2013)," Ghent University, 2013, URL: https://doi.org/10.4121/uuid:a7ce5c55-03a7-4583-b855-98b86e1a2b07/ [accessed: 2013-04-12].

[10] B. Van Dongen, "Bpi challenge 2014 (2014)," Rabobank Nederland, 2014, URL: https://doi.org/10.4121/uuid:c3e5d162-0cfd-4bb0-bd82-af5268819c35/ [accessed: 2014].

[11] R. J. C. Bose, E. H. Verbeek, and W. M. van der Aalst, "Discovering hierarchical process models using prom," in *International Conference on Advanced Information Systems Engineering*. Springer, Jun. 2011, pp. 33–48.

[12] F. Silva and P. Jacob, "Mission-centric risk assessment to improve cyber situational awareness," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, Aug. 2018, pp. 1–8.

[13] W. M. van der Aalst, C. Günther, J. C. Recker, and M. Reichert, "Using process mining to analyze and improve process flexibility-position paper," in *The 18th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortium*. Namur University Press, 2006, pp. 168–177.

 ISBN: 978-1-61208-785-6

[14] W. Van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–1142, Jul. 2004.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[16] A. M. Lamb *et al.*, "Professor forcing: A new algorithm for training recurrent networks," in *Advances In Neural Information Processing Systems*, 2016, pp. 4601–4609.

[17] R. J. C. Bose and W. M. van der Aalst, "Process mining applied to the bpi challenge 2012: divide and conquer while discerning resources," in *International Conference on Business Process Management*. Springer, Sep. 2012, pp. 221–222.

[18] N. Mehdiyev, J. Evermann, and P. Fettke, "A multi-stage deep learning approach for business process event prediction," in *2017 IEEE 19th Conference on Business Informatics (CBI)*, vol. 1. IEEE, Jul. 2017, pp. 119–128.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, Dec. 2014.

 ISBN: 978-1-61208-785-6