A Graph-Based Approach to Interpreting Recurrent Neural Networks in Process Mining

Article *in* IEEE Access · November 2020

DOI: 10.1109/ACCESS.2020.3025999

CITATIONS
30

READS
2,174

3 authors:

Khadijah Hanga
Birmingham City University
3 PUBLICATIONS **179** CITATIONS
SEE PROFILE

Yevgeniya Kovalchuk
University College London
41 PUBLICATIONS **533** CITATIONS
SEE PROFILE

Mohamed Medhat Gaber
Birmingham City University
283 PUBLICATIONS **9,455** CITATIONS
SEE PROFILE

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 14, 2020, accepted September 19, 2020, date of publication September 22, 2020,
date of current version October 1, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3025999

A Graph-Based Approach to Interpreting Recurrent Neural Networks in Process Mining

KHADIJAH MUZZAMMIL HANGA, (Member, IEEE), YEVGENIYA KOVALCHUK, AND MOHAMED MEDHAT GABER

School of Computing and Digital Technology, Birmingham City University, Birmingham B4 7XG, U.K.

Corresponding author: Khadijah Muzzammil Hanga (khadijah.hanga@mail.bcu.ac.uk)

This work was supported by the Petroleum Technology Development Fund of Nigeria.

ABSTRACT Process mining is often used by organisations to audit their business processes and improve their services and customer relations. Indeed, process execution (or event) logs constantly generated through various information systems can be employed to derive valuable insights about business operations. Compared to traditional process mining techniques such as Petri nets and the Business Process Model and Notation (BPMN), deep learning methods such as Recurrent Neural Networks, and Long Short-Term Memory (LSTM) in particular, have proven to achieve a better performance in terms of accuracy and generalising ability when predicting next events in business processes. However, unlike the traditional network-based process mining techniques that can be used to visually present the entire discovered process, the existing deep learning-based methods for process mining lack a mechanism explaining how the predictions of next events are made. This study proposes a new approach to process mining by combining the benefits of the earlier, visually explainable graph-based methods and later, more accurate but unexplainable deep learning methods. According to the proposed 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 visually interpretable process model graph that represents the decision-making process of the LSTM model. The level of detail in this graph can be adjusted using a probability threshold, allowing to address a range of process mining tasks such as business process discovery and conformance checking. The advantages of the proposed approach over existing LSTM-based process mining methods in terms of both accuracy and explainability are demonstrated using real-world event logs.

INDEX TERMS Directly-follows graph, explainable AI, long short term memory, process mining, recurrent neural network.

I. INTRODUCTION

Many modern businesses realise that keeping up with the current growing and competing markets entails developing and utilising novel ways of managing business processes [1]. While event data captured by various information systems can be employed for performing accounting, auditing and business analytics, the challenge is not only about deriving useful insights from a large amount of collected data, but also having the ability to oversee and improve workflows. Process mining is an approach that allows addressing this latter challenge. In particular, it is used to automatically extract process models from event logs so as to analyse how processes are being implemented in reality, thereby providing the avenue for their monitoring and improvement.

Process mining exposes how processes are actually being executed as opposed to how they ought to be executed. A process mining model is distinct from a manually drafted process model [2] due to its ability to provide an accurate accounting of what is happening in reality. It brings about transparency, delivers timely factual evidence and insights from transaction records.

The most widely studied type of process mining is *process discovery*. Process discovery relies on data collected from an information system over a period of time, or real-time data output by running processes, widely known as an *event log*. From an event log, process discovery methods can automatically construct a process model appropriately displaying the observed behaviour as it is captured in the log without

The associate editor coordinating the review of this manuscript and approving it for publication was Qingli Li.

any inferred information. These models enable businesses to make performance evaluations, check compliance, spot anomalies and suggest improvements. Real-life applications of process discovery methods are reported, for example, in studies by Van Der Aalst *et al.* [3], Veit *et al.* [4], Rojas *et al.* [5] and Ostovar [6], where the discovered process models helped management teams in (1) spotting abnormal behaviours, process drifts and manipulations that negatively impacted the business performance, (2) investigating the root causes of the problems and (3) planning for certain organisational benchmarks.

Several approaches to process discovery have been proposed in the literature, including probabilistic-based approaches, frequency-based heuristics, genetic-based heuristics, theory of regions and hybrid methods [7]. The applicability and effectiveness of these approaches depend on the event log features and structure of processes. When applied to real-life event logs, the majority of existing process discovery methods demonstrate such limitations as producing broad and spaghetti-like models or models with poor *fitness* and *precision* (i.e. these methods are unable to discover process models that would express observed behaviour in the best possible way). It has proven difficult to attain a compromise between the four process discovery quality metrics, *fitness, precision, generalisation* and *complexity* [7].

More recently, deep learning (DL) approaches have gained popularity in the process mining field. In particular, several Recurrent Neural Network (RNN) architectures such as Long Short-Term Memory (LSTM) have been applied to predict next events in business processes, time of occurrence and completion of events, and resources that trigger the events. Owing to its ability to retain information over a long period of time, LSTM can learn long-term dependencies in a sequence, preventing prior information from being lost [8]. This enables an LSTM model to consider all process instances into account when making a prediction. Compared to the earlier process mining techniques, LSTM-based methods have shown better performance in terms of accuracy and generalising ability.

Noise, duplicate tasks, undetected or missing tasks that can often be found in real-life event logs are challenging issues for process mining algorithms to address. Thus, algorithm performance is highly dependent on the event log quality and associated processes [9]. Because of that, process mining methods applied to unstructured logs tend to discover process models that are often purposely unsound representations of the actual processes, thus compromising on the quality criteria. While process models are visually explainable, their implausibility makes them less reliable for making prediction [10]. As a result, there are debates in the process mining community on which methods to use. Recent studies such as [10] and [11] consider the DL's ability to generalise and yield high accuracy more imperative than generating visually explainable process models represented using graphical notation such as Petri-nets, Business Process Model and Notation (BPMN) language, Event-driven Process Chains or UML activity diagrams [12]. Unlike these studies, we argue that both accuracy and explainability are crucial in the process mining field. In models generated by DL-based methods, process structures are implicitly reflected, while no visually explainable process graphs are produced, which limits the value of such models.

To address this issue, the present study proposes a graph-based approach to explaining the decision-making process of an LSTM model when generating a sequence of events representing a business process. According to this approach, an LSTM model is trained on an event log first. This model is then employed to find probabilities for each event present in the log to appear in the business process next. Finally, these probabilities are used to generate a visually explainable process model graph representing the decision-making process of the LSTM model. A probability threshold is introduced as a parameter to manage the graph complexity and thus enable faster and directed business processes analysis, including discovering most common event sequences and unusual or suspicious behaviours.

In summary, the contribution of this study is two-fold. First, we propose two LSTM models that achieve higher accuracy when predicting next events in business processes than existing LSTM models. The better performance of the proposed models can be attributed to a different way of pre-processing event logs to generate inputs for the models and the model network architectures that employ an embedding and a dense layer, in addition to an LSTM layer (unidirectional and bidirectional for the first and second model, respectively). Second, we propose a way of generating graphs representing each model's decision-making process. These graphs can be used for exploring the LSTM model performance and identifying cases that are difficult for the model to deal with so that measures could be taken to improve the model performance in such cases. Furthermore, the graphs can be used for various process mining purposes such as business process discovery and conformance checking as an alternative to the existing graph-based methods that learn process models directly from data rather than through a machine learning model. The advantage of the proposed approach, in this case, is the ability of the generated graph to explain the decision process of the accurate LSTM to a degree of generality set by the user through a probability threshold. While we demonstrate the predictive performance of the proposed LSTM models through experiments employing real-life logs, the capability of the resulting graphs to explain the LSTM models is demonstrated visually. Another advantage of the proposed approach is its model independence; any other machine learning model (or deep learning architecture) can be used instead of LSTM.

The rest of the paper is structured as follows. Section 2 presents background and related work on process mining and methods for business process discovery and prediction. Section 3 details the proposed approach. Section 4 describes the experimental setup, while Section 5 presents and discusses the experimental results. Finally, Section 6 concludes the paper and outlines future work.

II. BACKGROUND AND RELATED WORK

A. PROCESS MINING

Process mining is concerned with the automatic extraction of process models from event logs so as to analyse how processes are being implemented in reality [13]. It is also used to ensure conformance with business rules and regulations and identify improvement opportunities [5]. The main purpose of process mining is to derive knowledge and insights from transaction records. Process mining can facilitate business processes analysis from the following perspectives: control-flow (i.e. discovering sequences of activities in a process), organisation (i.e. discovering the interplay and collaborations among the participants of a business process) and performance (e.g. identifying the maximum or average time it takes to perform an activity). The information gained from process mining can be used to analyse and gauge the performance of an organisation, detect obstacles, predict run times, spot anomalies such as control flow deviations, discover roles and relationships, and improve the usage of resources [14].

Event logs are the basis of all process mining techniques [6]. They contain cases (also called process instances), which are traces or sequences of events. An example of a typical event log is shown in Table 1, where each row corresponds to an event, while each column corresponds to an attribute of the listed events. Events belonging to the same process instance are grouped using the same Case ID. Time-stamp refers to the date and time of task completion, Activity refers to the performed task and Resource refers to the participant who performed the task. An event log may contain other optional columns such as transaction type and cost.

TABLE 1. Example of an event log.

Case ID	Time-stamp	Activity	Resource	...
01	30-12-2010 11:02	register request	Pete	-
01	31-12-2010 11:32	check ticket	Sue	-
01	05-01-2011 12:00	make decision	Sarah	-
01	07-01-2011 12:50	reject request	Pete	-
02	30-12-2010 11:15	register request	Mike	-
02	30-12-2010 11:30	check ticket	Mike	-
02	05-01-2011 11:55	make decision	Sean	-
02	08-01-2011 12:55	pay compensation	Ellen	-
...	...	...	...	...

Process discovery, conformance checking and process model enhancement are the commonly performed process mining tasks. Process discovery is concerned with building a process model from an event log automatically aiming at representing the actual business process as close as possible. Conformance checking is used to verify the compliance of the business process participants by comparing a discovered process model with the actual logs and vice-versa. Finally, model enhancement is concerned with adding value to or extending the originally discovered process model using information about the actual process recorded in an event log [15].

Business process mining has been highlighted as a potential way of managing organisational challenges. For example, Van Der Aalst *et al.* [3] used an event log from the work flow management system of a department in charge of construction and maintenance of water and road infrastructure in the Netherlands to analyse the performance of the business from the process (i.e. how), organisational (i.e. who) and case (i.e. what) perspectives. The authors discovered the main flow in the invoice handling, which led to the identification of exceptional loops that had a great impact on the process performance. The outcome of that analysis helped the management team to plan and aim for certain organisational benchmarks. It also made it possible for them to spot abnormal behaviours.

Veit *et al.* [4] applied a conformance checker to detected manipulations in purchase prices and identified the potential root cause of the violation demonstrating how people working in the process collaborated with each other to manipulate the prices. According to a survey by Rojas *et al.* [5], process mining is gaining increased interest and acceptance in healthcare to discover process models, check conformance and analyse social networks.

Another applicability of process mining techniques is detecting business process drifts. Business processes are subject to unplanned changes and disruptions that can adversely affect the performance of the processes. Examples of such changes include changes in regulations, resource capacity, technological capabilities and seasonal factors [6].

B. DEEP LEARNING FOR SEQUENCE PREDICTION

Deep learning (DL) has allowed to achieve ground-breaking results in various tasks, including image recognition, speech recognition, anomaly detection, disease diagnosis and natural language processing. DL is powered by neural networks composed of several interconnected layers of neurons performing nonlinear transformation of data to enable the network to learn patterns observed in the data.

Being one of the most popular DL architectures, Recurrent neural network (RNN) is acknowledged for its ability to learn and generalise over sequences of inputs rather than individual patterns [16]. RNN's can analyse patterns across time, they are able to understand short and long term dependencies or temporal differences. RNN contains recurrent or repeated connections with hidden states that are distributed across time, allowing the network to keep past information effectively. In RNN, the output at the current time step is the input to the next time step. As a result, at each point of the sequence, an RNN model takes into consideration not just the current input, but also what it remembers about the previous points, thus learning temporal dependence and contextual information in input data. This memory quality allows the network to learn long-term dependencies in a sequence, preventing prior information from being lost [17]. This implies the RNN model's ability to consider the context when making a prediction, e.g. when predicting the next word in a sentence, classifying sentiments and estimating temperature measurements. At the same time, the length of contextual information that can be captured by a typical RNN model is limited, which can negatively affect the model's performance.

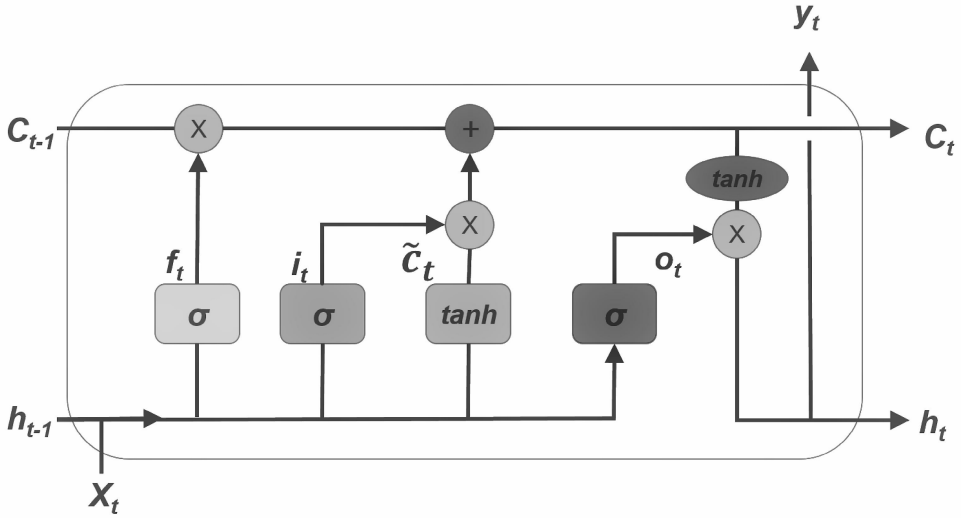


FIGURE 1. The long short-term memory (LSTM) cell.

This is due to the problem of vanishing or exploding gradients during the RNN model training using back propagation [8].

1) LONG SHORT-TERM MEMORY

A variation of the RNN architecture called long short-term memory (LSTM) has been proposed to overcome the problem of vanishing or exploding gradients. The control-flow of LSTM is the same as that of RNN; the main distinction is in operations within LSTM cells. The cell, also called a memory block, is the memory of the network that retains information over random time intervals (Fig. 1). Information is updated to or removed from a cell through three gate units. In particular, the *Forget gate* decides what information to keep or discards, the *Input gate* decides what relevant information to update the cell state with and the *Output gate* determines the output. In each LSTM cell, the *sigmoid* (σ) activation function is employed to decide which information to keep or disregard, while the hyperbolic tangent, *tanh*, function is employed to assist in regulating the network. The sigmoid function outputs numbers between 0 and 1, indicating how much of each component should be let through. A value of 0 means '*nothing should be let through*', while a value of 1 means '*let everything through*'.

The LSTM cell operations can be described using the following formulae:

$$
\begin{aligned}
f_t &= \sigma(W_f.[h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i.[h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_c.[h_{t-1}, x_t] + b_c) \\
C_t &= f_t * C_t - 1 + i_t * \tilde{C}_t \\
o_t &= \sigma(W_o.[h_{t-1}, x_t] + b_o) \\
h_t &= o_t * \tanh(C_t).
\end{aligned}
$$

Using information from the previous hidden state h_{t-1} and current input x_t, f_t calculates the forget gate output, i_t calculates the input gate output, $\tilde{C}_t$ calculates the eligible candidate, C_t calculates the new cell state, o_t calculates the output gate output and h_t calculates the new hidden state. W and b represent the weight and bias parameters learned during training, respectively.

2) BIDIRECTIONAL LONG SHORT-TERM MEMORY

Bidirectional LSTM (BLSTM) [18] splits the state neurons of the regular LSTM into two, the forward state (responsible for positive time direction) and backward state (responsible for negative time direction), both of which are connected to a common output layer. This architecture makes it possible to train a model using all past and future information of a specific time frame. Figure 2 shows a basic BLSTM structure, with the LSTM net at the bottom indicating the forward state, and the LSTM net at the top indicating the backward state. Without the backward state, it becomes a regular unidirectional LSTM.

Depending on the problem domain, the BLSTM structure can give better results than other network structures. For example, a BLSTM model was reported to achieve a remarkable performance in speech processing tasks, where content is crucial [19].

3) DEEP LEARNING IN PROCESS MINING

Several recent studies employed RNN- and LSTM-based models for monitoring and predicting business processes. Motivated by the application of RNN to predicting the next word in a sentence, Evermann *et al.* [10] proposed several RNN models to predict future process events from an event log. The authors considered the event log as text, traces as sentences and events in a trace as words in a sentence. The models built in this study were able to predict the most probable remaining sequence of events in an ongoing case. The limitation of the models was their inability to handle

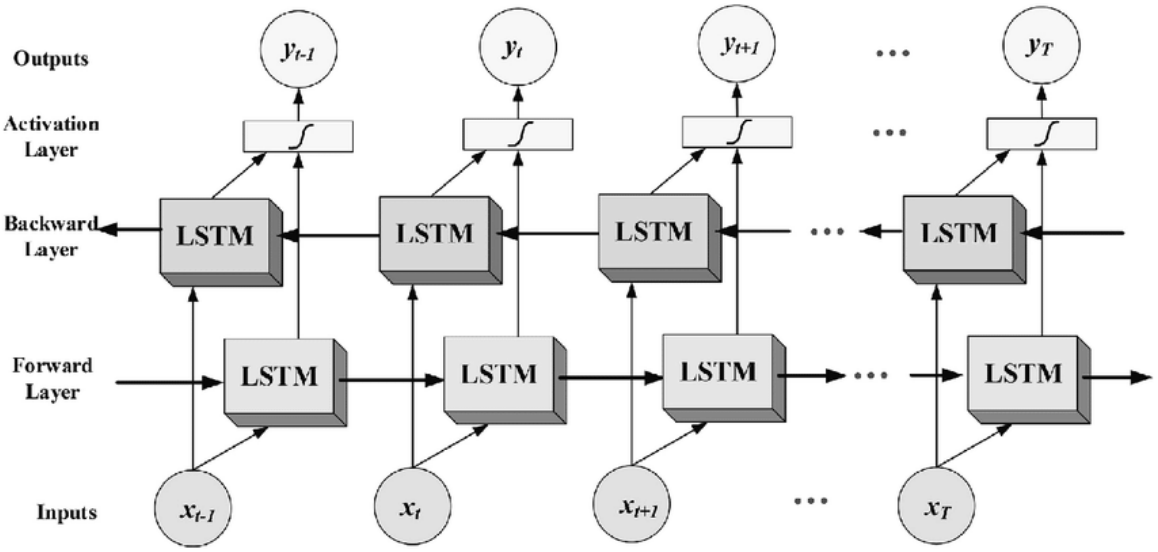


FIGURE 2. Basic unfolded bidirectional LSTM (BLSTM) Structure.

numerical variables, which made generating sequences of timestamped events impossible.

Tax *et al.* [20] used LSTM to build accurate models that could achieve multi-task learning, namely, predicting the next event of a running case and its timestamp. The models were able to predict the continuation of a case up to completion and remaining cycle time of running cases. However, the models did not perform well when making long-term predictions and when used on event logs with many repeated events. To overcome these limitations, Camargo *et al.* [21] proposed an LSTM architecture with embedded dimensions to predict traces of events, time-stamps and the role associated with each event. The phases in their approach included scaling and extracting n-grams of fixed sizes for each event log trace to create input and target sequences for training the predictive model, and a post-processing phase, where predicted next events were randomly selected from the likely ones to generate a greater number of different traces. Francescomarino *et al.* [22] also employed the LSTM architecture proposed by Tax *et al.* [20] and improved its performance by accounting for a-priori knowledge using *A-PRIORI algorithm* on the traces of the test set and dealing with cycles using the *NOCYCLE algorithm* to overcome issues encountered when dealing with traces containing a high number of cycles. Tello-Leal *et al.* [12] presented an LSTM network applied to event logs from the Internet of Things (IoT) domain within the context of industry 4.0. While the network could successfully predict the next activity in a business process, its performance in predicting continuous traces was not always good.

The most recent study in sequence modelling [23] presented an RNN-based modulated model for multi-task prediction of event sequences and event attributes. The model (called MM-Pred) includes an LSTM encoder-decoder and a modulator capable of learning inter-dependencies among event attributes. The MM-Pred model was compared to two other models: S-Pred for predicting next events based on event sequences alone (using the same LSTM encoder-decoder architecture but without the modulator) and M-Pred for predicting next events and their attributes together (employing the modulator).

TABLE 2. Summary of RNN- and LSTM-based models for monitoring and predicting business processes.

Study	Dataset	Model Architecture	Prediction Type	Quality Metric
Tax et al. [20]	Helpdesk BPIC12w	2 LSTM Layers A Dense Output Layer	Next activity, its Timestamp	Accuracy, MAE
Camargo et al. [21]	BPIC12 BPIC13 Helpdesk BPIC15	Input Layer 2 LSTM Layers A Dense Output Layer	Next event, Suffixes	D-L distance, Accuracy, MAE
Lin et al. [23]	BPIC12 BPIC14 BPIC17 Helpdesk	Encoder-Decoder LSTM Modulator A Dense Output Layer	Next activity, Attributes	D-L distance Accuracy

C. PROCESS DISCOVERY VERSUS PROCESS PREDICTION METHODS

Process discovery methods are often used to improve the comprehensibility of control-flow relations between tasks as observed in event logs. Discovered process models can help to understand how a process should be performed or is performed in reality. But most importantly, they provide a foundation for further analysis. Transition diagrams such as Petri-nets, BPMN diagrams, causal nets, state machines and directed acyclic graphs are often used to represent the outcomes of process discovery methods [7] (Fig. 3). Using process modelling notations for the visualisation of business

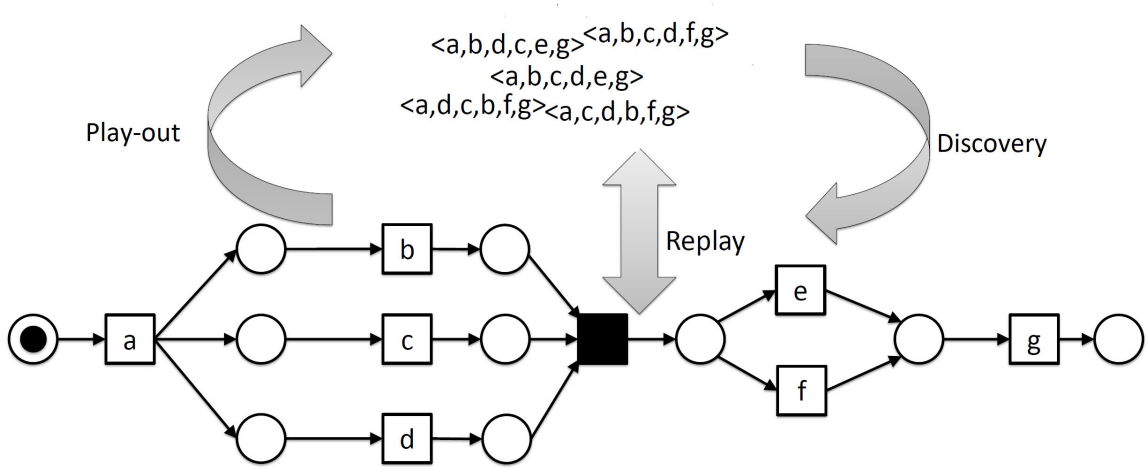


FIGURE 3. Graph-based representation of a process model.

process models ensures that domain experts, who may not have technical knowledge to comprehend how different process mining methods work, could understand the generated graphs and decide whether the model's structure reflects or contradicts the experts' domain knowledge.

Often presented as an alternative to process discovery methods [11], DL-based sequence modelling techniques such as LSTM have gained momentum in the process mining field. Sequence prediction is concerned with predicting future events based on past events and can be applied to both ongoing and completed cases of an event log. Next event predictions produced by DL models have been demonstrated to be more accurate than those made through process models discovered using event logs [10], [11], [21]. However, the decision-making process of how these predictions are made remains unclear when using DL models. This is unlike process discovery methods, the outputs of which can be represented as visually explainable process model graphs clearly demonstrating the sequence of events and instances of events executed in parallel.

This study addresses the problem of the lack of a mechanism capable to explain the decision-making process of DL models when predicting the next events by offering an approach to generating graphs representing DL outputs. Similar to graphs representing process models discovered through event logs directly, the graphs generated using the proposed approach can be used to perform various process mining tasks.

III. PROPOSED APPROACH TO PROCESS MINING

The proposed approach to process mining consists of two stages: building an accurate LSTM model for predicting business process event sequences based on event logs and generating a directly follows graph (DFG) explaining the decision-making process of the LSTM model when predicting business process event sequences. Figure 4 summarises the steps of the two stages of the proposed approach.

A. DATA PREPARATION

The proposed approach takes as input an event log defined as follows.

Definition 1 (Event Log): An event log $\mathbf{L}$ is a set of traces $\mathcal{T}$ holding a set of events $\mathcal{E}$ recorded as the result of each execution of a process instance. Each trace t contains a sequence of events $t = e_1, e_2, \ldots, e_n$, where $e_i \in \mathcal{E}$ and $1 \leq i \leq n$. Each event can have a set of attributes $\mathcal{P}$ providing additional details of the event, e.g. timestamp or resource executing the specified activity.

1) EVENT LOG PRE-PROCESSING

The concept of Natural Language Processing (NLP) was employed to parse event logs and train the proposed LSTM model. The model was trained to establish the most likely activity to come next in a given event sequence over time. This problem also resembles a time-series problem due to the existence of sequence dependence among input variables. In NLP, when predicting the next word to come in a sentence, the context of the whole sentence is considered to avoid ambiguity [24]. The same concept was used to frame the input sequences for the LSTM model from event logs. Similar to free text, event logs are also prone to ambiguity due to the varied length of sequences and the existence of repeated activities. To improve the model's training process, the context was broadened by phrasing the problem in such a way that multiple previous time-steps were considered when predicting the next time-step. In particular, event logs were pre-processed according to the following protocol and definitions.

The *NULL* label is used to mark the beginning of each case (or process instance). It becomes the first input activity x_0 at the current time t_0, and the target activity y_0 becomes the

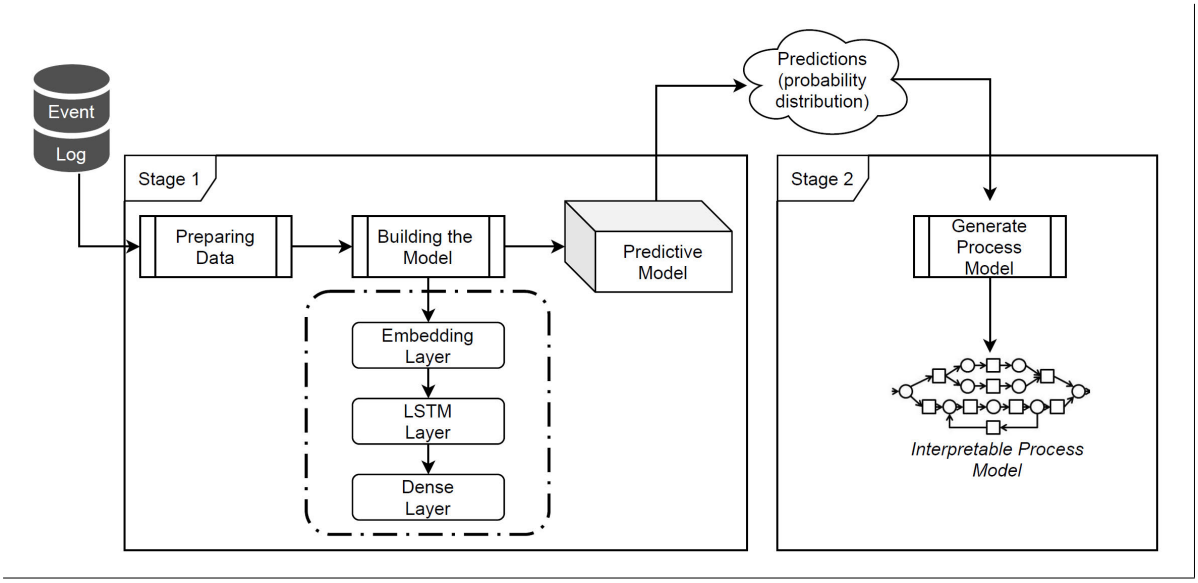


FIGURE 4. Flowchart summarising the proposed approach to process mining.

activity at time t_1 (this is the first activity occurring in each case of the event log). The next input activity sequence x_1 becomes the activity at the prior times $\{t_0, t_1\}$, and the target y_1 becomes the next activity at time t_2. The next input activity sequence x_2 becomes the activity at the prior times $\{t_0, t_1, t_2\}$, and the target y_2 becomes the next activity at time t_3; and so on. The inputs X build-up for each next input sequence as prior activities are joined with the current activity. The targets Y are always the activities at the next time-step t_{n+1} until the last input sequence contains all the activities at prior time-steps, including the current activity, at which point, the *END* label is added to mark the end of a case. This procedure is repeated until all cases in the event log are pre-processed.

Definition 2 (Predicting the Next Activity): Given a trace of activities $t = a_1, a_2, \ldots, a_t$, the output of the predictive model is the next activity $\{a_{t+1}\}$.

Definition 3 (Predicting Complete Traces): Given the prefix of a trace $t = a_1, a_2, \ldots, a_t$ and **END** value to mark the end of each case, the output of the predictive model is the sequence of activities $\{a_{t+1}, a_{t+2}, \ldots, END/a_t\}$.

2) ENCODING AND PADDING

The input sequences are encoded using the *Tokenizer class* from the *Keras* library. The tokenizer maps each activity in an event log to a unique integer creating a sequence of integers. Next, the prepared sequences are padded using the *pad sequences()* function from Keras. The function first finds the longest sequence and then uses the length to pad other sequences, so as to have a uniform length. The targets were dummy-encoded using the *pd.get_dummies()* function from the Pandas library. The function converts categorical variables into dummy variables by placing a value of one when a categorical event occurs and zero when it does not occur. For instance, if $A = [a, b, c]$, the respective tokens are $A \rightarrow [1, 2, 3]$, where $a = 1$, $b = 2$ and $c = 3$. Each activity $a_i \in \mathcal{A}$ is encoded as a vector (A_i) of length $|A| + 3$ such that the first $|A|$ features are all set to zero, except the one occurring at the index of the current activity a_i, which is set to one. Table 3 shows an example of prepared input sequences, where [1, 2, 3, 4, 5, 6] are the resulting tokens or integers of the tokenized activities $[NULL, a_1, a_2, a_3, a_4, a_5]$, and the target column contains the encoded dummies (i.e. the converted categorical labels) $[a_1, a_2, a_3, a_4, a_5, END]$, where every activity is set to zero except the target activity, which is set to one. Whenever the target is *END* encoded as [1000000] in the target column, the next process instance is visited. This procedure is repeated for all process instances in the event log.

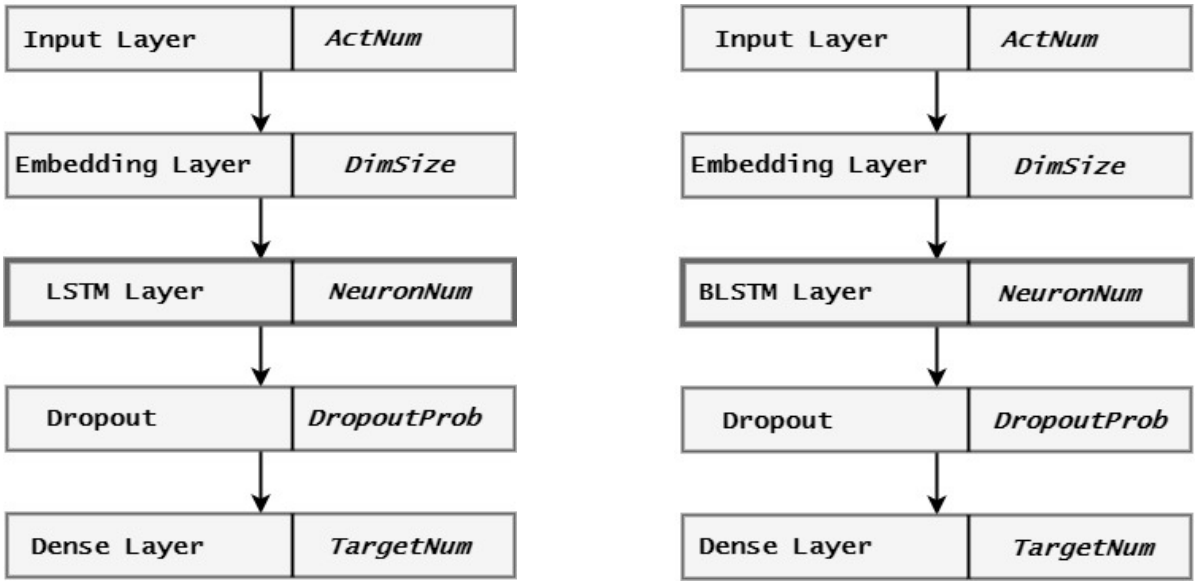
TABLE 3. Example of prepared data for training the proposed LSTM models.

Time-step (t)	Input	Target
t_0	[0, 0, 0, 0, 0, 0]	[0100000]
t_1	[0, 0, 0, 0, 0, 1]	[0010000]
t_2	[0, 0, 0, 0, 1, 2]	[0001000]
t_3	[0, 0, 0, 1, 2, 3]	[0000010]
t_4	[0, 0, 1, 2, 3, 5]	[0000100]
t_5	[0, 1, 2, 3, 5, 4]	[0000001]
t_6	[1, 2, 3, 5, 4, 6]	[1000000]
t_0	[0, 0, 0, 0, 0, 0]	[0100000]
t_1	[0, 0, 0, 0, 0, 1]	[0000010]
t_2	[0, 0, 0, 0, 1, 5]	[0001000]
...	...	...

B. PROPOSED MODELS FOR PREDICTING NEXT EVENTS OF BUSINESS PROCESSES

1) MODEL ARCHITECTURES

In addition to the unique way of pre-processing event logs as described above, another distinctive feature of the proposed models compared to other LSTM models for process mining



(a) Model 1. (b) Model 2.

FIGURE 5. Deep learning network architectures of the two proposed LSTM models: (a) Model 1 - unidirectional LSTM; (b) Model 2 - bidirectional LSTM.

presented in the literature is their architectures. In particular, an additional embedding layer is employed as an interface between the input and LSTM layers of the network. The embedding layer takes three arguments: the vocabulary size (i.e. set of unique activities in a log), embedding vector space size, and length of input sequences. These parameters are dependent on event log characteristics. The output of the embedding layer serves as input to the LSTM layer. A single LSTM layer is followed by an additional fully connected dense layer as the output layer. The output layer uses the softmax activation function to ensure the output takes the form of probability distributions. While some of the existing LSTM models for process mining add a dense layer on top of the LSTM layer, they either do not use embedding at all or implement it differently (e.g., see [20] and [21]). Furthermore, the existing LSTM models for process mining employs two LSTM layers, whereas the proposed models have only one LSTM layer. The first model (Model 1) employs a unidirectional LSTM, while the second model (Model 2) employs a BLSTM. The latter trains two LSTMs instead of one on the input sequence: the first – on the input sequence as it is and the second – on a reversed version of the input sequence. This gives additional context to the network, which can result in faster and improved learning.

Figures 5a and 5b illustrates the network architectures of the proposed DL models, where the output of each layer is the input to the next layer. In both architectures, the first *Input Layer* contains an activity sequence as constructed during the log event pre-processing procedure (see the previous section for details). The size of (ActNum) is determined by the number of unique activities in the event log. The *Embedding Layer* encodes the input sequence into a sequence of dense vectors of dimension (DimSize). While the dimension size can be tuned for better performance of the model, the choice should be guided by the number of unique activities in the log. The *LSTM Layer* in Figure 5a and *BLSTM Layer* in Figure 5b are characterised by the number of neurons (NeuronNum). Here, the vector sequence is transformed into a single vector of size (NeuronNum), containing information about the entire sequence. There is no set criteria on how many hidden neurons an LSTM layer should have, it is problem-dependant. For the problem considered here, the number of neurons is chosen considering the size of the event log and the number of unique activities in the log. A *Dropout* probability can be added to prevent overfitting and improve learning. The *Dense Layer*, which is the last layer, outputs the probability for each unique activity in a log to appear next in the sequence, including the extra *END* activity added to mark the end of the process. Hence, the output takes the form of a vector of size TargetNum (i.e. the number of expected targets), which is (ActNum + 1).

2) TRAINING THE LSTM MODELS

Providing the multi-class classification problem (i.e. predicting an activity from a list of activities), the categorical cross-entropy loss function was used during the training process of each model. The Adam optimisation algorithm (which is an implementation of the gradient descent algorithm) was used to track the loss and accuracy at the end of each epoch. The goal of the training process was to minimise the log loss by adjusting the trainable parameters (i.e. weights of the network).

Each model was fitted using training data and a variable number of training epochs and batch sizes. Each epoch

trained the model on the entire training event log while maintaining the weights and biases learned from the previous epochs. At the end of each epoch, test data were run through the model and the average accuracy across all possible next event cases in the test set was returned.

3) PREDICTING EVENT SEQUENCES OF BUSINESS PROCESSES

After the training phase, the LSTM models are ready to be used to predict the next activity and eventually the complete trace. Algorithm 1 lists the pseudo-code of the prediction algorithm. The algorithm takes the prefix $\mathcal{T}_p$ and trained LSTM model μ as inputs and returns the complete trace $\mathcal{T}$. First, $\mathcal{T}_p$ is encoded using the *tokenizer*. Then, the tokenized prefix is padded to the left with zeros to form a sequence of the same length and fed into the embedding layer. The resulting vector-matrix from the embedding layer is fed into the model μ, which then generates the probability distribution over different possible activities that can occur in the next position of the trace. The activity with the highest probability is returned as the next activity β. A new trace is obtained by concatenating the current prefix with the new predicted activity. The procedure is repeated until the model predicts the end activity denoted as *END* and until it does the same for all traces in the event log.

Algorithm 1: Next Activity Prediction

Input: Prefix: $\mathcal{T}_p$, LSTM model: μ
Output: Complete trace: $\mathcal{T}$
$h \leftarrow 0$
$\mathcal{T} \leftarrow \mathcal{T}_p$
while $(\beta <> END)$ ***and*** $(h < Log\_End)$ **do**
  $\mathcal{T}_p \leftarrow tokenize(\mathcal{T}_p)$
  $\mathcal{T}_p \leftarrow pad(\mathcal{T}_p)$
  $\mathcal{T}_p \leftarrow embed(\mathcal{T}_p)$
  $P \leftarrow Predict(\mu, \mathcal{T}_p)$
  if $\beta \leftarrow highest\_probability$ **then**
    $\mathcal{T} \leftarrow \mathcal{T}_p \cdot \beta$
      /* concatenate the current prefix with the new next activity */
    $h \leftarrow h + 1$
    return $\mathcal{T}$
  end
end

C. GRAPHS EXPLAINING THE PREDICTION PROCESS OF THE LSTM MODELS

A DFG is used in this study to explain the decision-making process of each LSTM model when predicting the next events of a business process. The DFG is generated according to Algorithm 2 based on the probabilities output at each iteration of Algorithm 1. Each arc (a directed edge) of the DFG is annotated with a prediction probability based on the following definitions.

Algorithm 2: Generating a Process Graph Based on LSTM Predictions

Input: Predicted Activities:
$\{\{a_{1,1}, a_{1,2}, \ldots a_{1,x}\} \ldots \{a_{n,1}, \ldots a_{n,x}\}\}$,
Threshold: ϕ
Output: Process graph: $\mathcal{G}$

Create *START* node
Create *END* node
$\eta_0 \leftarrow \emptyset$
  /* list of all start events */
$N \leftarrow \emptyset$
  /* set of established nodes */
for $a \in A$ **do**
  $\eta_0\langle i \rangle \leftarrow a_{i,1}$
  $E : START \rightarrow \eta_0\langle i \rangle$
    /* Create a transition */
  $counter \leftarrow 0$;
  $event\_list \leftarrow \emptyset$
    /* $event\_list \subseteq a$ */
  $o \leftarrow \eta_0\langle i \rangle$
  if $\beta \leftarrow highest\_probability$ ***and*** $\beta \notin N$ **then**
    $N \leftarrow (N, \beta)$
  end
  if $P \leq \phi$ **then**
    $E : o \rightarrow \beta$
    $E : Append(P)$
      /* add probability to the edge */
    $o \leftarrow \beta$
    $E : o \rightarrow END$
    return $\mathcal{G}$
  end
end

Definition 4 (Directly-Follows Probability): Given a sequence of activities $\{a_1, a_2, \ldots, a_n \in \mathcal{E}\}$, the directly-follows probability between a_1 and a_2 is the LSTM model's next activity probability assignment for a_2.

Definition 5 (Directly-Follows Graph): Given a directly follows probability matrix, its DFG is a directed graph $\mathcal{G} = (i, o, N, E)$, where i is the start event, o is the end event, N is a non-empty set of nodes and $E \subseteq (x, y)|x, y \in N$ is the set of edges.

A process is a directed graph if there exists a graph node for each activity of the process ($N = A$). The graph edges represent the possible transitions between the activities. For any two parallel activities $a, b \in A$, there will be directed arcs (a, b) and (b, a) between them.

The *START* and *END* nodes are introduced to produce sound process graphs. This is necessary to prevent deadlocks or lack of synchronisation [25]. The prediction probability matrix is generated first and then used to build a graph from a set of traces. The matrix describes the activities that succeed other activities in the set of traces. The rows in the matrix

represent the trace prefixes, while the columns represent the activity that directly follows. The numbers in the matrix cells represent the next activity prediction probabilities.

Generating the probability matrix entails feeding the first activity a_1 of each trace to the LSTM model and getting the predictions of each possible activity being the next one in the sequence. The activity with the highest probability is selected as the most likely next event a_2. Next, the combination of the first activity and predicted next activity $\{a_1, a_2\}$ are fed into the LSTM model to predict the next activity a_3; and so on. For each trace, this continues until the LSTM model predicts the next activity to be *END*, which automatically marks the end of each process instance.

The process graph is generated by traversing each row in the matrix, picking the column with the highest probability as the most likely next activity, and creating a transition between preceding and succeeding activity (through drawing an edge E between the nodes N). This is repeated until all rows in the matrix are visited. Algorithm 2 highlights the procedure. To begin with, a set of all start activities (η_0) (i.e. activities that appears first in each trace), a set (N) of all established nodes(to avoid having duplicate activities in the graph), and a set of edges (E) are initialised. From the *START* node, a transition is created to the start activity of the first trace. The start activity is used to determine the next and subsequent activities, creating a transition between each preceding and succeeding activity β (i.e. from the matrix, the activity with the highest probability in each row). This is repeated for each case until the end of the matrix is reached. The activity that appears as the last activity o in each trace is joined to the *END* node. Parallel transitions are observed if two or more activities have the same prediction probabilities, which indicates that the process may be branching (i.e. two or more activities may follow the previous activity in the process).

To make the graph more informative, the transition probabilities P are appended to its edges. Furthermore, a probability threshold ϕ is introduced as a parameter to allow tuning the complexity of the graph. While the main purpose of the graph is to explain the decision-making process of the LSTM model when predicting the next events in a business process, it can also be used to perform various process mining tasks. For example, by setting a high probability threshold, one can visualise the most common way of the business process execution. In contrast, by setting the threshold to a low value, one can capture less common instances on the business process execution, potentially indicating cases on non-conformance.

Another benefit of the proposed approach is its ability to generate traces for building a *single process graph* for each case by simply specifying the *case id*. This can be of great benefit when the interest is in investigating particular cases.

IV. EXPERIMENTS

Algorithms 1 and 2 were implemented as a set of Python scripts using Python 3.6. The LSTM models were built using the Keras [26] and Tensorflow [27] libraries. The process graphs were generated using the Graphviz library [28], while trace graphs for each process instance were generated using the NetworkX library [29]. The experiments were carried out using the Google Colab free Tesla K80 GPU.

A. EVALUATION

To evaluate the predictive performance of the proposed LSTM models, they were applied to the following five real-life logs: Helpdesk [30], BPIC 2012 [31], BPIC 2013I [32], BPIC 2013C [32] and Road traffic fine management process [33]. The details of these logs are provided in the next section and Table 4. The accuracy of the next event predictions of the proposed two models was compared to that of three other LSTM models proposed by Tax *et al.* [20], Camargo *et al.* [21] and Lin *et al.* [23]. From the five real-life logs listed above, these studies employed only the Helpdesk and BPIC 2012 logs.

Tax *et al.* [20] used one-hot encoding to encode events as inputs to an LSTM model that had two layers and 100 neurons in each layer. Camargo *et al.* [21] extracted n-grams of fixed sizes for each event log trace, which they used as inputs to their LSTM model consisting of two stacked LSTM layers and a dense output layer. Also, Camargo *et al.* [21] used pre-trained embedded dimensions in all their experiments. Lin *et al.* [23] used a two-layer LSTM-based encoder-decoder architecture with 32 neurons per layer for each encoder and decoder cell. Random embedding was established for each event. Only the S-Pred model proposed by Lin *et al.* [23] was considered in this study as it is concerned with predicting next events based on event sequences alone (i.e. without considering additional information such as event attributes).

Being the most popular, the Helpdesk log was employed to generate process graphs to visually demonstrate the capability of the approach presented in this article to explain the LSTM decision-making process when predicting event sequences of business processes. To the best of our knowledge, we are the first to propose such an approach.

B. REAL-LIFE EVENT LOGS

The proposed approach was evaluated on the following five real-life event logs. The original log files were presented in the xes format but for the purpose of this study, were converted to csv files using the ProM tool. The sizes of these event logs vary, Table 4 shows the statistics. #Traces, #Events and #Activities stand for the total count of traces, events and activities in each log respectively.

Helpdesk [30] - This event log contains records of real-life events from a ticketing management process of the help desk of an Italian software company.

BPIC 2012 [31] - This event log represents a loan application process of a German financial institution. It contains three intertwined sub-processes. The first alphabet of each task name identifies from which sub-process this task originated from.

TABLE 4. Event log description.

Event log	#Traces	#Events	#Activities
Helpdesk	4,580	21,348	14
BPIC 2012	13,087	262,200	25
BPIC 2013I	7,554	65,533	13
BPIC 2013C	1,487	6,660	7
Road Traffic Mgt Process	150,370	561,470	11

BPIC 2013I and BPIC 2013C [32] - These are the real-life event logs of the Volvo IT incident and problem management system called VINST. The first includes Incident cases pertaining to the incident management process, while the second includes Closed cases pertaining to closed problems in the problem management process, both of which are used in the experiment.

Road Traffic Fine Management Process [33] - This is a real-life event log of an information system managing road traffic fines.

C. EXPERIMENTAL SETUP

The following experimental protocol was adopted to closely follow the experiments conducted by Tax *et al.* [20], Camargo *et al.* [21] and Lin *et al.* [23] for a fair comparison with their models:

1) Similar to Tax *et al.* [20] and Camargo *et al.* [21], each event log was divided into two sets: a *training set* composed of 70% of traces used as input for training the LSTM models and a *test set* composed of the remaining 30% used to evaluate the predictions and assess the models' ability to generalise (Lin *et al.* [23] also used 70% of traces for training but only 20% of traces to test their models' accuracy in predicting next events). The splitting was performed using the algorithm proposed by [10], which is consistent with the method used by Tax *et al.* [20].
2) For the embedding layer, the vocabulary size was set to the number of unique activities in the log. The number of embedding dimensions was guided by the vocabulary size, hence it varied across the event logs.
3) The networks had one hidden LSTM layer with a dropout probability of 0.2. The dropout technique was used to avoid over-fitting and improve learning by temporarily removing a cell from the network in a random manner.
4) The number of training epochs was set to 50 for all logs.

The parameters of the proposed models, namely, the number of embedding dimensions and LSTM neurons were tuned for each event log separately for better performance. In particular, 32 embedding dimensions were used for the BPIC13 log and 50 dimensions were used for the three other logs; 100 neurons were used in the LSTM layer for all logs. For a fair comparison with the other three LSTM models proposed by Tax *et al.* [20], Camargo *et al.* [21] and Lin *et al.* [23], each model's performance was evaluated based on the average accuracy score achieved across all possible next event cases in the test set.

V. RESULTS AND DISCUSSION

A. MODEL PREDICTIVE PERFORMANCE

Table 5 summarises the performance of the proposed LSTM models on the considered event logs based on the accuracy of predicting the next activity over the test set. The results are compared to those achieved by the other three models proposed by Tax *et al.* [20], Camargo *et al.* [21] and Lin *et al.* [23]. For the comparison, only the Helpdesk and BPIC2012 event logs were used since they are the only logs out of the five considered here, for which results are reported in the other three studies.

TABLE 5. Prediction results over the test sets.

Implementation	Event Log	Next Activity Prediction Acurracy
Model 1: Unidirectional LSTM	Helpdesk	**0.912**
	BPIC 2012	**0.854**
	BPIC 2013I	0.694
	BPIC 2013C	0.640
	RTFMP	0.814
Model 2: Bidirectional LSTM	Helpdesk	0.907
	BPIC 2012	0.853
	BPIC 2013I	**0.698**
	BPIC 2013C	**0.644**
	RTFMP	**0.815**
Tax et al. [20]	Helpdesk	0.712
	BPIC 2012	0.760
	BPIC 2013I	-
	BPIC 2013C	-
	RTFMP	-
Camargo et al. [21]	Helpdesk	0.789
	BPIC 2012	0.786
	BPIC 2013I	-
	BPIC 2013C	-
	RTFMP	-
Lin et al. [23]	Helpdesk	0.808
	BPIC 2012	0.814
	BPIC 2013I	-
	BPIC 2013C	-
	RTFMP	-

It can be noticed from Table 5 that the proposed Model 1 and Model 2 outperform the other three models on the Helpdesk event log, achieving accuracy scores of 91.2% and 90.7% compared to 71.2%, 78.9% and 80.8% achieved by Tax's *et al.* [20], Camargo's *et al.* [21] and Lin's *et al.* [23] models, respectively. On the BPIC2012 event log, the proposed Model 1 and Model 2 achieve 85.4% and 85.3%, respectively, compared to 76.0%, 78.6% and 81.4% achieved by Tax's *et al.* [20], Camargo's *et al.* [21] and Lin's *et al.* [23] models, respectively.

The better performance of the proposed models can be explained by the difference in the event log pre-processing approach and models' architecture. In particular, Tax *et al.* [20] encoded activities in event logs using one-hot encoding. This type of encoding might be suitable for logs with a small number of activities but it becomes inefficient for logs with a large number of activities. Furthermore, while

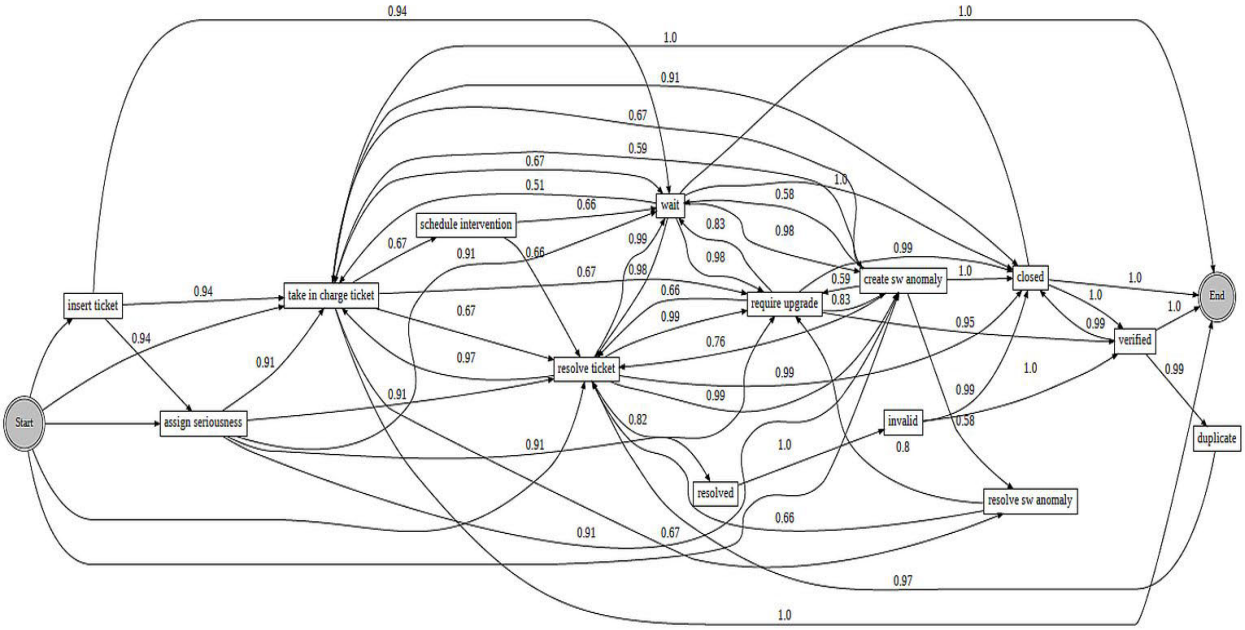


FIGURE 6. Graph demonstrating the decision-making process of the unidirectional LSTM model (Model 1) when generating likely event sequences representing a business process based on the training set of the Helpdesk log with the probability threshold set to 0. The nodes of the graph represent activities, while the edges represent transitions between the activities. The numbers on the edges represent the probabilities of the transitions as predicted by the LSTM model.

Camargo *et al.* [21] employed the word embedding technique, they trained an independent network to coordinate the embedded dimension, which they used throughout their experiments as non-trainable parameters (which is different from our approach of employing an embedding layer as part of one network architecture). They probably did this to shorten the training time while improving the quality of the predictive model, but it is clear that this approach cannot guarantee a compatible prediction accuracy. Finally, the proposed models employ only one LSTM layer, as opposed to two LSTM layers used by Tax *et al.* [20], Camargo *et al.* [21] and Lin *et al.* [23]. We believe that this simplification contributes to the better performance of our models in the case of the Helpdesk log, which has a simple sequence flow.

Another observation that can be made from Tables 4 and 5 is that the proposed models perform well on the larger RTFMP log (Model 1: 81.4%; Model 2: 81.5%) but not so good on the smaller BPIC2013I and BPIC2013C logs (Model 1: 69.4% and 64.0%; Model 2: 69.8% and 64.4%). The latter result can be explained by the relatively small number of traces (e.g. compared to the BPIC 2012 log) and the relatively large number of events (e.g. BPIC2013C compared to the Helpdesk log) included in the BPIC2013 logs (i.e. the logs included many repetitive traces). This means that the models are expected to pick up the sequences of activities that can happen in reality among a large number of possible combinations of these activities when provided only with a small number of examples of these combinations that happened in reality. Model 2 performs slightly better than Model 1 on the BPIC2013I, BPIC2013C, and RTFMP logs in terms of both the training and test accuracy. It can thus be deduced that the forward and backward learning in Model 2 is beneficial in the cases of large logs and complex logs with repetitive activities. At the same time, it took longer to train Model 2 than Model 1. Hence, there is a trade-off between the accuracy and training time of the models. Since Model 2 improves accuracy only slightly compared to Model 1, the latter is preferred as being both simple and accurate.

B. GRAPHICAL REPRESENTATION OF MODEL PREDICTIONS

In addition to presenting better performing LSTM models for process mining, another contribution of this study is proposing an idea of generating graphs of different complexity to visually explain the decision-making process of an LSTM model when predicting the next events in business processes. These graphs can be used for exploring the LSTM model performance and identifying cases that are difficult for the model to deal with so that measures could be taken to improve the model performance in such cases. Furthermore, the graphs can be used to perform various process mining tasks such as model discovery, conformance checking, and investigating cases of non-compliance.

Figures 6, 7, 8 and 9 illustrate the graphs generated using the outputs of Model 1 for the Helpdesk log. In particular, Fig. 6 shows the graph generated based on the predictions of Model 1 over the training set, while Fig. 7 shows the same for the test set. The probability threshold was set to 0 for both graphs so that all transitions present in the log could be represented in the graphs. The numbers on the edges indicate

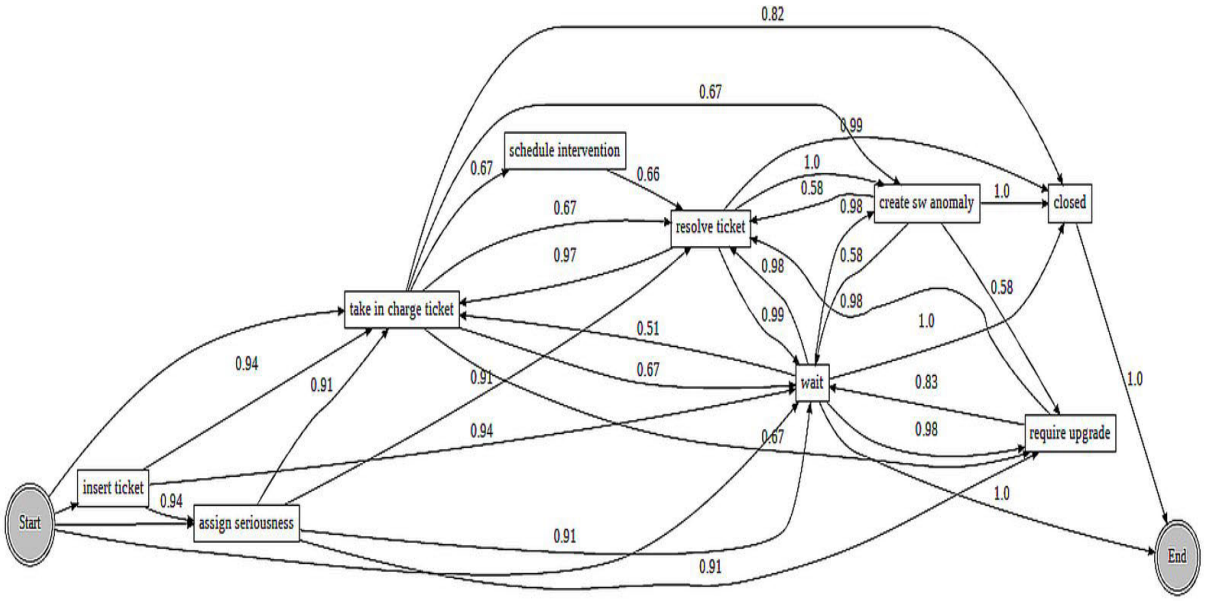


FIGURE 7. Graph demonstrating the decision-making process of the unidirectional LSTM model (Model 1) when generating likely event sequences representing a business process based on the test set of the Helpdesk log with the probability threshold set to 0. The features of the graph are the same as in Fig. 6.

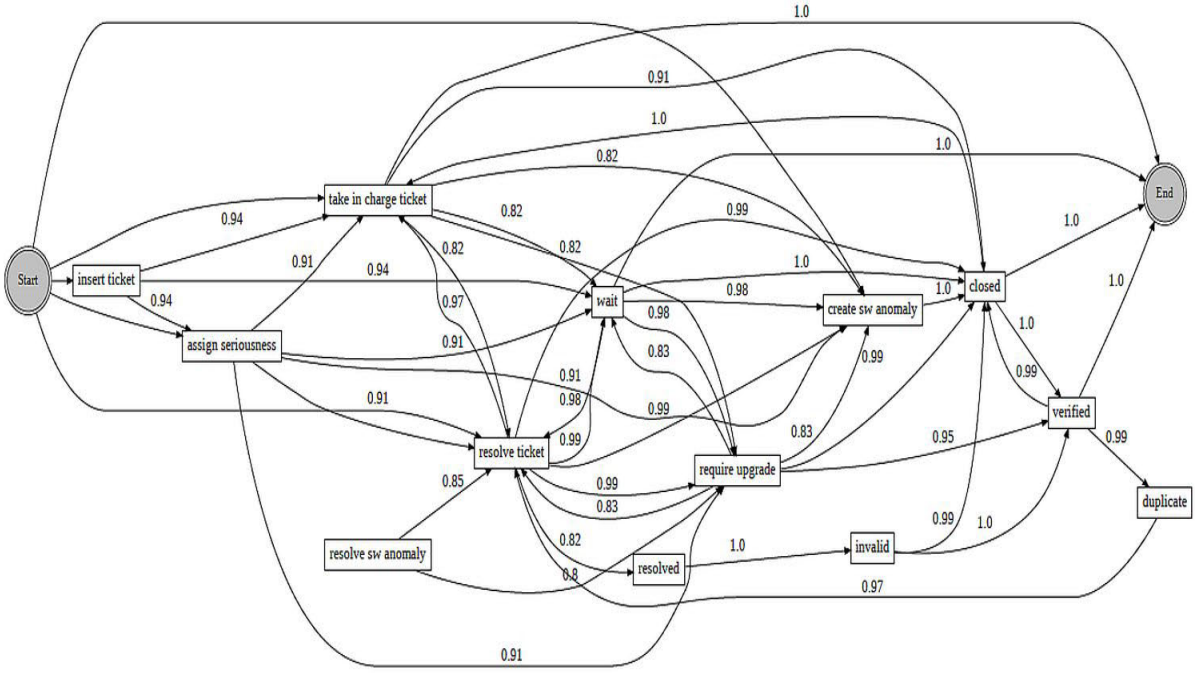


FIGURE 8. Graph demonstrating the decision-making process of the unidirectional LSTM model (Model 1) when generating likely event sequences representing a business process based on the test set of the Helpdesk log with the probability threshold set to 0.8. The features of the graph are the same as in Fig. 6.

the probability scores of the respective activities to be next in the process as predicted by the LSTM model.

When comparing the two graphs, one can notice their consistency, which indicates a good ability of both the model to generalise and graph built over the training set to interpret the model's behaviour. To formally verify the similarity between the training and test graphs, they were converted into adjacency matrices first, and then the similarity score between the matrices was calculated by taking the sum of the differences between the values in corresponding cells of the two matrices and dividing it by the number of non-zero values in the test matrix. Since we are interested in evaluating the interpreting

TABLE 6. Case 1: Training matrix created from the graph generated using the *train set* of the Helpdesk log. The rows and columns represent activities. The values on the intersections of columns and rows represent the transitions between the corresponding activities, with 1 indicating the presence of the transition and 0 indicating its absence.

0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	0	1	0	1	1
0	1	0	0	1	0	1	0	1	0	1	1
0	0	0	1	0	0	1	0	1	0	1	1
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	1
0	0	0	1	1	0	1	0	0	1	1	0

TABLE 7. Case 1: Test matrix created from the graph generated using the *test set* of the Helpdesk log. The rows and columns represent activities. The values on the intersections of columns and rows represent the transitions between the corresponding activities, with 1 indicating the presence of the transition and 0 indicating its absence.

0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1
0	0	0	1	1	0	0	0	1	0	1	1
1	1	0	0	1	0	1	0	1	0	1	1
0	0	0	1	0	0	1	0	1	0	1	1
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0
1	0	0	1	1	0	1	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	1	0	1	0	1	0	0	1
0	0	0	1	1	0	1	0	1	1	0	0

ability of the training graph, The training matrix was reduced to the test matrix size, and the transitions that were captured during the training process but did not appear during testing were excluded from the calculation. A score of 0 obtained this way would mean the two graphs are identical, wheres a score of 1 would mean they are completely different. Two cases were considered for the calculation. In *Case 1*, binary matrices were constructed, where the value of 1 in a cell corresponded to the fact that the transition between the corresponding two activities existed, and the value of 0 represented

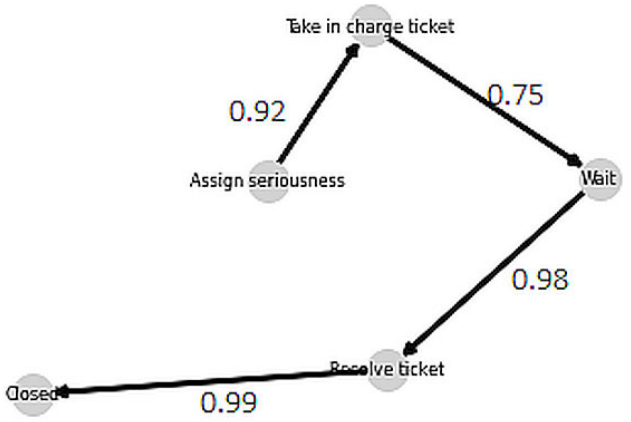


FIGURE 9. Single trace process graph generated based on the decisions of the unidirectional LSTM model (Model 1) for the test set of the Helpdesk log. The features of the graph are the same as in Fig. 6.

the fact that there was no transition between the two activities. In *Case 2*, the matrices were constructed in the same way as in *Case 1*, except that instead of the value 1, the probability of the transition as predicted by the LSTM model was recorded.

Tables 6 and 7 represent the adjacency matrices built for the binary-based *Case 1* using the process graphs illustrated in Figs. 6 and 7, respectively. Tables 8 and 9 represent the same for the probability-based *Case 2*. After dividing the sum of the difference between the training and the test matrices by the number of non-zero values in the test matrix (41), scores of 0.19512 and 0.18292 were obtained for *Case 1* and *Case 2*, respectively. Providing that the scores are close to zero, we can conclude that the two graphs are similar, which indicates a good ability of both the model to generalise and training graph to interpret the model's behaviour.

Figure 8 shows a less complex version of the graph presented in Fig. 6, which was achieved by setting the probability threshold to 0.8. This means that all predicted transitions with a probability less than the set threshold are pruned and not included in the graph, which results in a simpler visualisation. The probability threshold can be adjusted to any value as required by a task at hand, for example, to understand the LSTM model performance or perform process mining tasks. Finally, Figure 9 shows the graph of a single trace from the

TABLE 8. Case 2: Training matrix created from the graph generated using the *train set* of the Helpdesk log. The rows and columns represent activities. The values on the intersections of columns and rows represent the probabilities of the transitions between the corresponding activities.

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.79	0.86
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.69
0.0	0.0	0.0	0.92	0.92	0.0	0.0	0.0	0.92	0.0	0.92	0.92
0.0	0.73	0.0	0.0	0.73	0.0	0.81	0.0	0.73	0.0	0.73	0.73
0.0	0.0	0.0	0.54	0.0	0.0	1.0	0.0	0.99	0.0	0.99	0.99
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.97	0.97	0.97	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.72	0.0	0.0	0.0	0.72	0.0	0.0	0.0	0.0	0.0	0.72	0.72
0.0	0.0	0.0	0.0	0.0	0.82	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.85	0.0	0.0	0.0	0.0	0.0	0.0	0.87
0.0	0.0	0.0	0.97	0.97	0.0	0.97	0.0	0.0	0.99	0.0	0.0

TABLE 9. Case 2: Test matrix created from the graph generated using the *test set* of the Helpdesk log. The rows and columns represent activities. The values on the intersections of columns and rows represent the probabilities of the transitions between the corresponding activities.

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.79	0.89
0.0	0.0	0.0	0.0	0.69	0.0	0.0	0.0	0.0	0.0	0.0	0.69
0.0	0.0	0.0	0.92	0.92	0.0	0.0	0.0	0.92	0.0	0.92	0.92
0.73	0.73	0.0	0.0	0.73	0.0	0.84	0.0	0.73	0.0	0.73	0.73
0.0	0.0	0.0	0.54	0.0	0.0	1.0	0.0	0.99	0.0	0.99	0.99
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.97	0.97	0.97	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.48	0.0	0.0	0.48	0.72	0.0	1.0	0.0	0.0	0.0	0.72	0.54
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.85	0.0	0.99	0.0	0.85	0.0	0.0	0.97
0.0	0.0	0.0	0.97	0.97	0.0	0.97	0.0	1.0	0.99	0.0	0.0

Helpdesk log. This type of graph can be useful for investigating specific instances of the LSTM model predictions or business process execution.

VI. CONCLUSION AND FUTURE WORK

This article presented a new approach to process mining by combining the benefits of widely used graph-based methods for process discovery and deep learning methods for predicting event sequences. The proposed approach consists of two stages: building an accurate LSTM model for predicting business process event sequences based on event logs and generating a directly follows graph explaining the decision-making process of the LSTM model when predicting business process event sequences. Two model architectures were proposed, one using a unidirectional LSTM and another using a bidirectional LSTM, both of which were demonstrated to outperform the state-of-the-art LSTM models for process mining based on two real-life event logs. Further three real-life logs were employed to demonstrate the advantages and limitations of the proposed LSTM models. The bidirectional LSTM model achieved slightly better results than the unidirectional LSTM on more complex or larger logs but at the cost of training time. The capability of generated graphs to explain the LSTM models was demonstrated visually. An approach to determine the similarity between the graphs was proposed to further validate the generalising ability of the model, which gave a satisfactory result. In particular, the graphs can be used to understand the LSTM models' performance and perform a range of process mining tasks such as process discovery, conformance checking, and investigating cases of non-compliance.

One of the advantages of the proposed approach is its model-agnostic nature. This means that the graph-generation part is independent of the predictive modelling part. As such, any model can be used in place of LSTM. To exploit this advantage, as part of our future work, we plan to experiment with other deep learning architectures such as encoder-decoder networks. Furthermore, we plan to investigate the possibility of using the proposed approach as a process discovery method. To achieve this, we will evaluate the generated graphs using the metrics widely used in the process mining community for evaluating discovered process models, including model fitness, precision, generalising ability, and complexity.

REFERENCES

[1] B. Jokonowo, J. Claes, R. Sarno, and S. Rochimah, "Process mining in supply chains: A systematic," *Int. J. Electr. Comput. Eng. (IJECE)*, vol. 8, no. 6, pp. 4626–4636, 2018.

[2] H. C. W. Lau, G. T. S. Ho, Y. Zhao, and N. S. H. Chung, "Development of a process mining system for supporting knowledge discovery in a supply chain network," *Int. J. Prod. Econ.*, vol. 122, no. 1, pp. 176–187, Nov. 2009.

[3] W. M. P. van der Aalst, H. A. Reijers, A. J. M. M. Weijters, B. F. van Dongen, A. K. Alves de Medeiros, M. Song, and H. M. W. Verbeek, "Business process mining: An industrial application," *Inf. Syst.*, vol. 32, no. 5, pp. 713–732, Jul. 2007.

[4] F. Veit, J. Geyer-Klingeberg, J. Madrzak, M. Haug, and J. Thomson, "The proactive insights engine: Process mining meets machine learning and artificial intelligence," in *Proc. Int. Conf. Bus. Process Manage.*, Aug. 2017, pp. 1–6.

[5] E. Rojas, J. Munoz-Gama, M. Sepúlveda, and D. Capurro, "Process mining in healthcare: A literature review," *J. Biomed. Informat.*, vol. 61, pp. 224–236, Jun. 2016.

[6] A. Ostovar, "Business process drift: Detection and characterization," Ph.D. dissertation, Dept. Sci. Eng., Queensland Univ. Technol., Brisbane, Qld, Australia, 2019.

[7] A. Augusto, R. Conforti, M. Dumas, M. L. Rosa, F. M. Maggi, A. Marrella, M. Mecella, and A. Soo, "Automated discovery of process models from event logs: Review and benchmark," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 4, pp. 686–705, Apr. 2019.

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

[9] D. Pérez-Alfonso, O. Fundora-Ramírez, M. S. Lazo-Cortés, and R. Roche-Escobar, "Recommendation of process discovery algorithms through event log classification," in *Proc. Mex. Conf. Pattern Recognit.* New York, NY, USA: Springer, 2015, pp. 3–12.

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

[11] N. Tax, I. Teinemaa, and S. J. van Zelst, "An interdisciplinary comparison of sequence modeling methods for next-element prediction," 2018, *arXiv:1811.00062*. [Online]. Available: http://arxiv.org/abs/1811.00062

[12] E. Tello-Leal, J. Roa, M. Rubiolo, and U. M. Ramirez-Alcocer, "Predicting activities in business processes with LSTM recurrent neural networks," in *Proc. ITU Kaleidoscope, Mach. Learn. 5G Future (ITU K)*, Nov. 2018, pp. 1–7.

[13] W. Van Der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, vol. 2. Springer, 2011.

[14] D. R. Ferreira, *A Primer Process Mining*. Springer, 2017.

[15] W. V. D. Aalst, "Using process mining to bridge the gap between BI and BPM," *Computer*, vol. 44, no. 12, pp. 77–80, Dec. 2011.

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[17] N. D. C. Lewis, *Deep Time Series Forecasting with Python: An Intuitive Introduction to Deep Learning for Applied Time Series Modeling*. Philadelphia, PA, USA: ND Lewis, 2016.
[18] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.
[19] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Netw.*, vol. 18, nos. 5–6, pp. 602–610, Jul. 2005.
[20] N. Tax, I. Verenich, M. La Rosa, and M. Dumas, "Predictive business process monitoring with LSTM neural networks," in *Proc. Int. Conf. Adv. Inf. Syst. Eng.*, Springer, 2017, pp. 477–492.
[21] M. Camargo, M. Dumas, and O. González-Rojas, "Learning accurate lstm models of business processes," in *Proc. Int. Conf. Bus. Process Manage.*, Springer, 2019, pp. 286–302.
[22] C. Di Francescomarino, C. Ghidini, F. M. Maggi, G. Petrucci, and A. Yeshchenko, "An eye into the future: Leveraging a-priori knowledge in predictive business process monitoring," in *Proc. Int. Conf. Bus. Process Manage.*, Springer, 2017, pp. 252–268.
[23] L. Lin, L. Wen, and J. Wang, "Mm-pred: A deep predictive model for multi-attribute event sequence," in *Proc. SIAM Int. Conf. Data Mining*. Philadelphia, PA, USA: SIAM, 2019, pp. 118–126.
[24] J. Brownlee, *Deep Learning for Natural Language Processing: Develop Deep Learning Models for your Natural Language Problems*. Victoria, BC, Canada: Machine Learning Mastery, 2017.
[25] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1128–1142, Sep. 2004.
[26] A. Gulli and S. Pal, *Deep Learning With Keras*. Birmingham, U.K.: Packt Publishing Ltd, 2017.
[27] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, and and M. Kudlur, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.
[28] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull, "Graphviz–open source graph drawing tools," in *Proc. Int. Symp. Graph Drawing*, Springer, 2001, pp. 483–484.
[29] A. Hagberg, P. Swart, and D. S. Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM, USA, Tech. Rep. LA-UR-08-05495; LA-UR-08-5495, 2008.
[30] *U. of Padova, and D. of Mathematics*, Mirko, Santa Ana, CA, USA, Jan. 2017.
[31] *V. Dongen, E. U. of Technology, and Boudewijn*, Eindhoven Univ. Technol., Eindhoven, The Netherlands, Apr. 2012.
[32] W. Steeman, "Volvo IT," Ghent Univ., Ghent, Belgium, Apr. 2013.
[33] *Massimiliano, Mannhardt, and Felix*, Eindhoven Univ. Technol., Eindhoven, The Netherlands, Jan. 2015.

KHADIJAH MUZZAMMIL HANGA (Member, IEEE) received the B.Tech. degree (Hons.) in computer science from Abubakar Tafawa Balewa University, Bauchi, Nigeria, in 2010, and the M.Sc. degree in software development from Coventry University, U.K., in 2016. She is currently pursuing the Ph.D. degree in computing with Birmingham City University, U.K. Her research interests include artificial intelligence, machine learning, and data analytics.

YEVGENIYA KOVALCHUK received the M.Sc. degree in economic cybernetics from the National Technical University of Ukraine and the Ph.D. degree in computer science from the University of Essex, U.K. She is currently a Senior Lecturer in computer science with the School of Computing and Digital Technology, Birmingham City University, U.K. Her research interests include machine learning, artificial intelligence, and their applications across various business areas.

MOHAMED MEDHAT GABER received the Ph.D. degree from Monash University, Australia. He is currently a Professor in data analytics with the School of Computing and Digital Technology, Birmingham City University, U.K. He has published over 200 articles, coauthored three monograph-style books, and edited/co-edited six books on data mining and knowledge discovery. His work has attracted well over 5000 citations, with an H-index of 37. He is recognized as a Fellow of the British Higher Education Academy (HEA). He is also a member of the International Panel of Expert Advisers for the Australasian Data Mining Conferences. In 2007, he was awarded the CSIRO Teamwork Award. He has served in the program committees of major conferences related to data mining, including ICDM, PAKDD, ECML/PKDD, and ICML. He has also co-chaired numerous scientific events on various data mining topics.