A Multi-stage Deep Learning Approach for Business Process Event Prediction

Conference Paper · July 2017

DOI: 10.1109/CBI.2017.46

Peter Fettke
Deutsches Forschungszentrum für Künstliche Intelligenz
306 PUBLICATIONS 8,623 CITATIONS

2017 IEEE 19th Conference on Business Informatics

A Multi-Stage Deep Learning Approach for Business Process Event Prediction

Nijat Mehdiyev, Peter Fettke
Institute for Information Systems (IWi)
German Research Center for Artificial Intelligence (DFKI) and Saarland University
Saarbruecken, Germany
nijat.mehdiyev@iwi.dfki.de, peter.fettke@iwi.dfki.de

Joerg Evermann
Memorial University of Newfoundland
St. John's, NL, Canada
jevermann@mun.ca

***Abstract*—The ability to proactively monitor business processes is one of the main differentiators for firms to remain competitive. Process execution logs generated by Process Aware Information Systems (PAIS) help to make various business process specific predictions. This enables a proactive situational awareness related to the execution of business processes. The goal of the approach proposed in the current paper is to predict the next business process event, considering the past activities in the running process instance, based on the execution log data from previously completed process instances. By predicting the business process events, companies can initiate timely interventions to address undesired deviations from the desired workflow. In our study, we propose a multi-stage deep learning approach which formulates the next business process event prediction problem as a classification problem and applies deep feedforward multilayer neural networks after extracting features with feature hashing and deep stacked autoencoders. The experiments conducted on a variety of business process log datasets reveal that the proposed multi-stage deep learning approach provides promising results. The results are compared against existing deep recurrent neural networks and other approaches as well.**

Keywords-process prediction; deep learning; feature hashing; stacked autoencoders

I. INTRODUCTION

Event log data generated during process execution serves as a valuable source for designing predictive analytics approaches for various purposes such as estimation of technical production parameters for more robust production plan optimization, detecting process specific anomalies in real-time, analyzing the behavioral patterns of customers, risk management by predicting compliance violations, resource allocation and etc. Effective design and implementation of such predictive approaches are important to ensure that the business activities will run in a desired manner by avoiding the predicted failures and deviations from designed process structures at runtime.

Inspired by recent applications of deep learning for various classification problems, we explore a multi-stage deep learning approach to predict the next process events in the running process instance. A few studies have already addressed the issue of next process step forecasting by applying machine learning approaches. However, the prior studies rely heavily on assessing the performance of the proposed approaches and with a few exceptions, pay little attention to data pre-processing. Data preparation, comprising various stages such as data cleaning, encoding, dimensionality reduction, feature extraction and etc. influences the predictive ability of the classifiers significantly. This work distinguishes itself from prior research by incorporating an intensive data pre-processing stage. In our approach, we used the n-gram representation and feature hashing approach to build the numerical feature vectors from sequential business process event log data. According to our knowledge, no prior studies applied the feature hashing in this domain. Furthermore, this study for the first time applies stacked autoencoders to extract features from the pre-processed business process log data, and a deep feedforward neural networks model for multi-class classification problem. This paper aims to investigate the following research questions:

RQ 1: Does the proposed multi-stage deep learning approach improve the predictive performance of the state-of-the-art applications?

RQ 2: Do changes to parameters in various steps of the proposed approach, especially in the dimensionality reduction and deep neural networks, influence the prediction results?

The remainder of the paper is organized as follows: Section II outlines the problem identification, motivation and related work in the business process prediction domain. A broad insight to the details of the proposed multi-stage deep learning approach is provided in Section III. Section IV outlines experiment settings and empirical results. Finally, Section V concludes the paper with a discussion and summary.

II. MOTIVATION AND RELATED WORK

A growing body of the literature has recently examined the application of various predictive analytics approaches in business process management. These approaches can be categorized based on the type of the predictand. The first category comprises the approaches that investigate the regression problems, mainly the remaining processing time of incomplete cases related [1], [2], [3], [4], [5], [6], [7]. These studies apply various data mining approaches ranging from support vector regressors to annotated transition systems. The second category deals with various classification problems including business process outcome predictions, violation of service level agreements, nominal attribute prediction, next event prediction etc. [8], [9], [10], [11], [12].

In this study we focus on predicting the next business process event, considering the past events in the running process instance, based on execution log data from previously completed process instances. Predicting the next event in the running process instance is a vital issue in process analytics, since such information allows organizational decision making systems to intervene proactively. The main motivation of our study is to improve the prediction accuracy by coding and cleaning the raw input data and reducing the high dimensional input space by applying feature hashing and stacked autoencoders. Recent advances in computational efficiency of training neural networks are another motivator, since it allows us to achieve higher accuracy with reduced computational complexities.

Only a few attempts aimed at the next event prediction have been made. A multi-stage model, which starts by clustering event sequences using the k-mean algorithm combined with sequential alignment, builds the individual Markov models of different orders on the obtained clusters [13]. Experiments were conducted on records of processes obtained from a telecommunication company.

Another approach uses sequential k-nearest neighbor classification approach and an extension of Markov models to predict the next process steps by considering the temporal features [14]. Using the same process log data as in [13] revealed the superiority of their models over Markov and Hidden Markov Models (HMM).

As an alternative to Markov models, a decision tree based model is used to predict the next activity in the running instance of business processes that contains the parallel execution paths [15]. Five different models for representing the path attribute of the execution trace are presented and experiments were carried on the simulated data.

Combining the two approaches yields a hybrid model, which learns a decision tree at each individual node of the process model based on the execution traces to compute the transition probabilities and creates a Markov chain model [16]. A simulated dataset is used to verify the prediction accuracy.

Somewhat similar to a Markov model, a probabilistic finite automaton (PFA) based on Bayesian regularization which uses the Expectation Maximization (EM) approach to estimate the relevant process parameters [17]. The evaluation process was carried out using both simulated dataset and real-life datasets such as publicly available BPI Challenge 2012 and BPI Challenge 2013.

More recent work is moving away from explicit models to deep-learning approaches. In the first approach to apply deep-learning [18], the authors applied recurrent neural networks (RNN) with Long-Short Term Memory (LSTM) after transforming the input features using word embeddings. The improvement potentials of accuracy by adding the available case and event specific explanatory variables have been investigated as well. BPI Challenge 2012 and 2013 datasets were used to validate the prediction results.

As an alternative to word embeddings, one-hot encoding may be used [19]. Also applying the LSTM approach but only considering the occurrence sequence of the activities and their timestamps, the authors transformed the input activities to feature vectors using one-hot encoding. Both studies examined the prediction of process activity duration using the same approach, additionally, the latter study [19] also conducted experiments on another publicly available Helpdesk dataset from an Italian software company.

III. Multi-Stage Deep Learning Approach

In our paper, we propose a method which formulates the prediction of the next business process events as a classification problem. We apply deep learning algorithms on the extracted feature matrix from various process characteristics such as control flow, data flow, resource, and organization perspectives, after a thorough data pre-processing stage. Built upon event log data, the proposed approach is generalizable and can be implemented in domain independent settings. The proposed multi-stage deep learning approach begins with the reconstruction of the business process events (control flow) obtained from event log data and encoded in letters in the n-gram feature representation (Fig. 1). Thereafter, feature hashing is applied to map the extracted n-grams to hash keys. The hashed feature matrix is then extended by adding the relevant data flow features and resource features. Once the merged feature matrix is available, the deep stacked autoencoders are applied to extract high hierarchical feature representations in an unsupervised manner. At the final stage, the output of the autoencoders is provided as input to a fully-connected deep feedforward neural network with multiple hidden layers to predict the next process event to occur.

A. Terminology

An *event log* consists of *process traces*. Each *trace* represents the execution of one business process instance, also known as *case*. A *trace* is of a sequence of events. Events contain attributes that describe their characteristics (XES Standard). Typical attributes are the name of the executing activity, the timestamp of the event, the lifecycle transition (e.g. “start” or “complete”) and organizational resources or roles. *Events* are ordered by the timestamp of their occurrence. Other attributes may contain case specific information. The next event prediction problem is typically understood as predicting the executing activity and lifecycle transition of the next event in the running trace considering the sequence of past events from that particular trace.

B. Data Preparation

The initial step of the next business event prediction problem is the sequence encoding process, which is characterized as the conversion of character strings (business process events) into numerical input features. Reference [10] provided a comparative analysis of various sequence encoding schemas for business process outcome prediction.

Choosing the appropriate sequence encoding method is a crucial issue since it influences the accuracy and application of the machine learning approaches significantly. In the process event sequence data there are intrinsic relationships and interdependencies among the individual events. Therefore, the n-grams is a suitable approach for modelling such dependencies because of their ability to consider the relationships between neighboring elements by building all contiguous subsequences [20].

In this paper, we use the combination of n-grams of different sizes as sequence encoding schema which allows us to extract both local and global features from the business process event sequences.

Definition 1: *Given a sequence of the events* $E = (e_1, e_2, \ldots, e_{N+(n-1)})$ *over the event universe* φ*, where the N and n are the positive integers, an n-gram of the event sequence E is any n-long subsequence of the consecutive events. There are N such n-grams in E. The total number of unique n-grams is* $(|\varphi|)^n$ *where the* $|\varphi|$ *is the total number of unique events in the business process log data.*

Assume that we have the following sequence of business process events, E={A,F,G,C,L,B,A,D,A,M}. The bigram (2-gram) features are all combinations such as {AF, FG, GC …, AM}; the trigram (3-gram) features are {AFG, FGC, CLB …, DAM} and etc. As described above, in our paper, we consider the combination of n-grams of pre-defined sizes. For example, the size of our input feature space in the case of 5-grams (including unigrams (1-grams), bigrams (2-grams), trigrams (3-grams), quadgrams (4-grams)) and an alphabet size of 15 unique events is the number of all possible n-grams:

$$N_{total\_features} = 15+15^2+15^3+15^4+15^5 = 813{,}630$$

Due to their completeness (the a-priori known token (event in our case) alphabet), domain independence, efficiency (one pass processing) and simplicity, the n-grams approach has been applied for various problems ranging from protein classification to information retrieval [21]. The predictions relying on the n-gram event data require no special additional preprocessing stages such as sequence alignment. Moreover, the letter n-grams method is also very effective due to its ability not only encoding the letters but also ordering them automatically.

However, as can be seen from the above example, the major drawback of the n-gram representations is that the size of generated input features for classification problem tends to be extremely large. The number of such variables explodes polynomially with alphabet size and exponentially with n-gram length. Using all the generated features would overload the prediction system by leading to extremely high computational costs and reduced accuracy [22]. To address this challenge we have attempted two dimensionality reduction techniques, feature hashing and deep stacked autoencoder networks to diminish the size of n-gram feature vectors.

Feature hashing is an effective dimensionality reduction method that aims to scale up the classification algorithms by mapping the high dimensional input space into low dimension space [23]. The main idea is using the hash functions to map the words or letters, n-grams of events in our case, to the feature vectors which can be passed to the classification approach to train the model. In order to ensure that the hash kernel is unbiased, [23] includes a binary hash function. The formal definition is as follows:

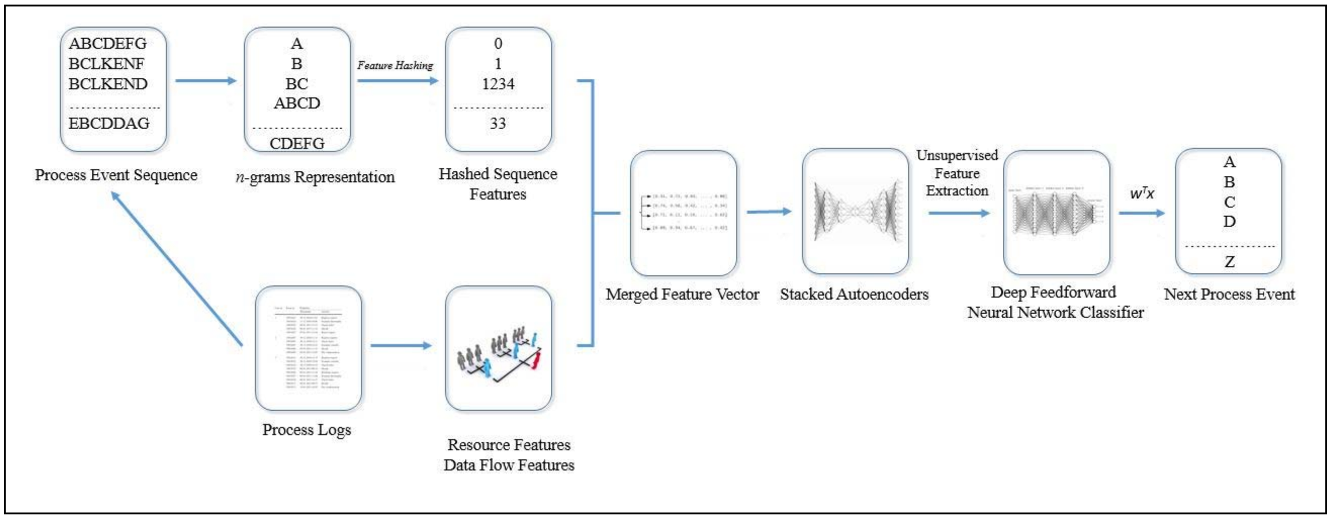

Figure 1. The Stages of the proposed approach

Definition 2: *Given a set of hashable features N, which are the n-grams obtained from the business process event sequences, h is the hash function, $h:N \rightarrow \{1,\ldots\ldots, m\}$ and ξ is the second hash function, $\xi:N \rightarrow \{\pm 1\}$. The hashed feature mapping function Φ maps the input vector $x \in R^d$ into a hashed feature space R^m where m<d. The kth dimension of hashed feature x is given as : $\Phi_i^{(h,}\xi)(x) = \sum_{i:h(i)=k} \xi(j)x_j$.*

As hash function h, we use the 32 bit murmurHash function available in the Vowpal wabbit online learning project library [24]. The dimensionality reduction via feature hashing can lead to the information loss due to hash collision, which is the mapping of multiple n-grams to the same hash keys. This problem can be prevented by using larger hash tables, by increasing the bitsize in the case of sparse data [23]. Bitsize determines the numbers of the bits when creating the hash table. The decision for choosing the optimal bitsize depends on the size of the n-gram vocabulary. Feature hashing has already found successful applications in diverse domains such news categorization, spam filtering, sentiment analysis in social networks and different areas of bioinformatics [20], [25], [26], [27]. The findings from these studies suggest that the using feature hashing to reduce the input space is especially important for a successful and effective application of predictive analytics approaches in resource constrained devices.

C. Modelling

The proposed multi-stage deep neural network approach consists of stacked autoencoders which extract the features from hashed inputs combined with data flow and resource variables in an unsupervised (or self-supervised) manner. The obtained final features are then provided to a deep neural network classifier to predict the next business process event (Fig. 2).

1) Deep Stacked Autoencoders

Although applying feature hashing reduces the input space significantly, the resulting vector size can be still too large when process event dictionaries are big. In such cases, there is a need to further reduce the input space. Principal Component Analysis (PCA) is a widely used technique for dimensionality reduction. However, PCA can only model linear interdependencies among the features of the given dataset. Autoencoders are the non-linear generalization of PCA [28]. As a feedforward neural network, an autoencoder consists of three layers including an encoder and decoder. The encoder layer takes the high-dimensional input vector $x \in [0, 1]^d$ and maps it to the hidden layer $y \in [0, 1]^{d'}$ by using a non-linear activation function (see next section). The decoder layer maps the hidden layer back to the reconstructed vector $z \in [0, 1]^d$. The main goal is the optimization of parameters ($\theta = \{W, b\}$, $\theta' = \{ W', b'\}$ - weights and biases in both encoding and reconstruction phase respectively) to minimize the reconstruction loss:

$$\theta^*,\theta'^* = \text{argmin}_{\theta, \theta'} 1/n * \sum L(x(i), z(i)) \qquad (1)$$

In our paper we used the Rectified Linear Units (ReLu) as the non-linear activation function and the Mean Squared Error (MSE) as the reconstruction loss function.

Stacked autoencoders are a greedy layer-wise approach which conduct multi-phase feature extraction by feeding the extracted primary features from previous autoencoders again as input vector to the new autoencoders. In this way, we can reduce the feature vector size significantly by preventing the severe loss of information. Once the features are extracted, we can train our classifier.

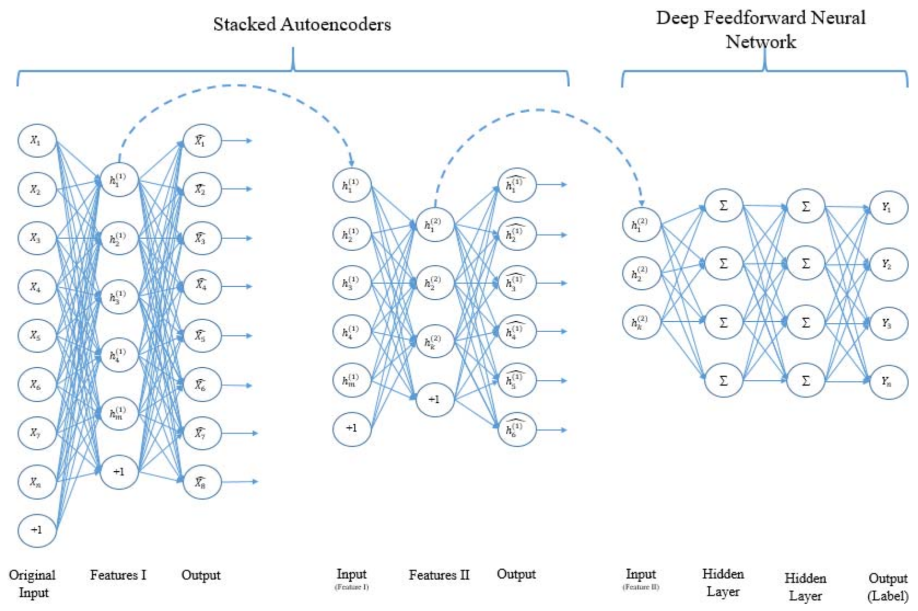

Figure 2. Multi-Stage deep neural networks

2) *Deep Feedforward Neural Networks*

We chose an artificial feedforward neural network as the classifier in our study as it offers a number of advantages over alternative classification approaches, including the need for less formal statistical training, the capability to detect complex non-linear relationships between predictors and the outcomes, the ability to model the interrelationships among the predictor variables and the availability of various training algorithms [29]. The superior performance of feedforward networks has already been documented in various comparative empirical studies and contests [30], [31], [32]. The main computational element in these neural nets is the neuron which aggregates the weighted combination, α, of its input signals,

$$\alpha = \sum w_i * x_i + b \tag{2}$$

where the w and b are the weights and bias respectively.

The neuron then transmits this aggregate as output signal to the connected neurons after transforming it with the activation function. The feed-forward networks have input, hidden and output layers consisting of the neurons presented above. The learning process is characterized as adapting the weights and biases to minimize the errors on training data. However, training deep networks with large input spaces using gradient based methods such as backpropagation leads to the vanishing gradient problem. This phenomenon results in extremely long training durations and reduced prediction performance. Breakthrough studies beginning in the mid-2000s offered solutions for training deep neural networks effectively [33], [34]. In our study we deploy the deep learning neural network framework of the H2O.ai platform [35]. We follow a supervised training protocol by starting the learning using a random initialization from a normal distribution.

We have defined the further specification of the deep learning process for predicting the next process event as follows:

a) Activation Function

The performance of the deep neural network classifier depends on the chosen non-linear activation function. Since it is difficult to identify which activation function leads to the best performance for a particular dataset and network architecture, we conducted our experiments using Tanh (3), ReLu (4) and Maxout (5):

$$f(\alpha) = (e^{\alpha} - e^{-\alpha}) / (e^{\alpha} + e^{-\alpha}) \tag{3}$$

$$f(\alpha) = \max(0, \alpha) \tag{4}$$

$$f(\alpha_1, \alpha_2) = \max(\alpha_1, \alpha_2) \tag{5}$$

b) Regularization

Prior research suggests that the dropout approach applied to deep multilayer perceptrons can significantly improve learning and prevent overfitting [36]. Hence, we employ dropout on our activation function.

c) Loss Function

We chose the Cross-Entropy as the loss function, L, since we deal with the classification problem:

$$L = -\sum (\ln (o_y^{(j)}) * t_y^{(j)} + \ln(1 - o_y^{(j)}) * (1 - t_y^{(j)})) \tag{6}$$

where $t^{(j)}$ is the predicted activity, $o^{(j)}$ is the actual output activity, *j* is the training example and *y* stands for the output unit.

d) Other Parameters

We use stochastic gradient descent (SGD) to minimize the loss function because it is a memory efficient and fast approach. A lock-free methodology was adopted to parallelize the SGD where the multiple cores contribute to gradient updates [37], [38]. Momentum terms change the adopted backpropagation by enabling prior iterations to alter the latest version of it. As a technique for accelerating the training process by speeding up the gradient descent, momentum helps to avoid local minima. Furthermore, the H2O platform implements ADADELTA as an adaptive learning method, which uses only first order information for dynamical adaptation and requires lower computational efforts compared to other alternatives [39]. The major drawbacks of prior learning methods such as continual decay of learning rates and the efforts for manual determination of global learning are eliminated through the ADADELTA approach. The values of the momentum, adaptive learning and other advance optimization parameters used in our experiments are summarized in Table I.

TABLE I. OTHER PARAMETERS

Parameter	***Value***
Optimizer	Nesterov Accelerated Gradient
Mini Batch Size	10
Learning rate	0.005
Adaptive Learning Approach	ADADELTA
Adaptive learning rate smoothing factor	1e-8
Adaptive learning rate time decay factor	0.99
Epochs	100
Initial Weight Distribution	Normal distribution
Beginning Momentum	0
Momentum Ramp	1e6
Input Dropout Ratio	0
Hidden Dropout Ratios	0.5 for each hidden layer

IV. Evaluation

To ensure a robust comparison with state-of-the-art methods [17], [18], [19] we performed our experiments on the same datasets by using all multi-class classification evaluation metrics adopted by these studies. Furthermore, we evaluated our approach in different settings by changing the parameters of our feature hashing, our autoencoders, and our multilayer deep feedforward neural network classifier.

The experiments were performed using an Intel Core i7-5500U 2.0 GHz processor with 16 GB RAM. For initial data preprocessing we used the data manipulation package *dplyr,* available in the RStudio software, which is an integrated development environment for R [40]. The feature hashing approach was carried out using the Microsoft Azure Machine Learning platform which implements the Vowpal Wabbit library [24], [41]. Stacked autoencoders and deep feedforward neural networks were created on the H20 open source deep learning platform [35].

We used 10-fold cross validation for training and testing data. This approach provides more reliable results than adopting the holdout technique. In 10-fold cross-validation the data are partitioned into the 10 disjoint subsets. Both training and testing are carried out 10 times. During these iterations one partitioned subset is used for testing purposes whereas the others serve as input for training the classifier. The mean values of relevant measures are calculated from the test results obtained from these repetitions.

The following subsections provide a broader overview to diverse aspects of demonstration and evaluation.

A. Datasets

The experimental process was carried out on the following various real-life datasets: The BPI Challenge 2012 [42], BPI Challenge 2013 [43], [44], [45] and Helpdesk [46] to evaluate the performance of the proposed multi-stage deep learning approach. Table II provides an overview of the number of unique event types and total number of events in all datasets. The number of unique event types also indicates the number of the output classes in our multi-class classification problem.

The **BPI Challenge 2012** dataset comprises event log data from 262.000 events in 13.087 cases obtained from a Dutch Financial Institute. The activities related to a loan application process are categorized into three sub-processes: processes related to the application (A), the work items belonging to applications (W) and the state of offer (O). Events for the A and O sub-processes contain only the completion lifecycle transition, while the W process includes the *scheduled*, *started* and *completed* lifecycle transitions Since all approaches presented in [17], [18], [19] used only the *completed* activities of work item processes, we filter out the activities with the lifecycle transitions *started* and *scheduled* from this sub process. In summary, similar to the previous papers we evaluate our approach on three datasets from BPI Challenge 2012: BPI_2012_A, BPI_2012_O and BPI_2012_W_Completed.

TABLE II. Characteristics of Datasets

Datasets	*# of unique event types*	*# of events*
BPI_2012_W_Completed	6	72.413
BPI_2012_A	10	60.849
BPI_2012_O	7	31.244
BPI_2013_Incidents	13	65.533
BPI_2013_Problems	7	9.011
Helpdesk	9	13.711

The **BPI Challenge 2013** dataset contains log data obtained from an incident and problem management system of Volvo IT in Belgium. This dataset has three sub-datasets: The incident management dataset encompasses 7554 cases with 65534 events of 11 unique event types. The open problems dataset contains 819 cases with 2351 event of 5 unique event types and the closed problems dataset comprises 1487 cases with 6660 events of 7 unique event types. We also merged both open and closed problems dataset in order to create a final dataset identical to that in other studies. After combining both problem datasets we obtained 9011 process events.

The **Helpdesk** dataset comprises event data from a ticketing management system designed for the help desk of an Italian software company. The event log contains 3804 cases with 13710 events.

Moreover, the log data from all three datasets not only provides the timestamp of the event occurrence and process trace IDs, but also describes various additional resource and case specific information. This information was also considered in modelling. The BPI Challenge 2012 data provides both process specific information such as the identification number of the resources responsible for processes, and case specific information such as the amount of the requested loan. The BPI Challenge 2013 datasets also contain information about the priority of the problems and incidents, involved functional divisions and organizational lines, related products, process owners' countries and names. Only the Helpdesk dataset provides neither case nor resource specific information, therefore we use only hashed features for modelling.

As mentioned above, after generating the feature vectors from the sequence of the activities through n-grams and feature hashing approaches, we append the provided additional information to the feature vector. Due to the ability of the deep feedforward neural networks to handle the categorical input variables no special transformation is required for training the classifier.

B. *Evaluation Metrics*

We calculate accuracy, weighted precision and recall using the formulas tailored to multi-class classification problems. Accuracy is defined as the proportion of correctly predicted instances of all instances. Precision determines how many activities were correctly classified for the particular class given all prediction of that class whereas the recall is the true positive rate for that particular class (Table III). We adopt the one-versus-all approach for computing the precision and recall values for each individual class and obtain the overall value by summing up their weighted scores.

$$\text{Accuracy} = (tp+tn)/(tp+tn+fp+fn) \tag{7}$$

$$\text{Precision}_w=\sum c_i /N * tp_i /(tp_i+fp_i) \tag{8}$$

$$\text{Recall}_w= \sum c_i /N * tp_i /(tp_i+fn_i) \tag{9}$$

C. *Evaluation Results*

The results for the proposed multi-stage deep learning approach and state-of-the-art approaches are presented in Table IV. We used four different variations of our model where we adjusted the (i) feature hashing properties such as bitsize and length of n-grams and (ii) the properties of deep neural networks by using different non-linear activation functions and changing the number of the neurons in the hidden layer. By doing so we aim to answer RQ 2, whether various hyperparameter adjustments lead to better results.

To address RQ 1, we compared the results of our model against state-of-the-art approaches which used the same datasets. The results for all three BPI 2012 datasets suggest that the proposed approach outperforms all three prior, state-of-the-art, approaches. A bigger difference can be observed for the BPI_2012_W_Completed dataset where our approach achieves an accuracy rate of 0.827 compared to 0.719 and 0.760 in [17] and [19] respectively. The performance gap is much greater than [17] in terms of recall (sensitivity). The comparison of the results with [18] in terms of precision, reveals the superior performance of the proposed approach (0.806 vs. 0.658) as well. Only two authors used the BPI_2012_A and BPI_2012_O datasets to evaluate their models. Our approach outperforms both models in terms of all evaluation measures. The approach by [18] performs better for the latter two and achieves close results as in ours.

TABLE III. SYMBOLS USED IN EQUATIONS 7-9

Symbol	*Meaning*
tp	Events with the positive event type classified correctly
fp	Events with negative event types classified as positive
tn	Events with nevative event types classified correctly
fn	Events with positive event type classified wrongly

For BPI_2013_Incident dataset the results are mixed. The approach by [17] shows higher predictive performance than ours in terms of accuracy (0.714 vs. 0.655). However, our approach performs significantly better in terms of the recall (0.654 vs. 0.377). Precision results obtained by [18] are better than our approach. However, the experiments conducted on the BPI_2013_Problems dataset suggest that the proposed approach in this study delivers superior results compared to all alternatives. Finally, only [19] carried out experiments on the Helpdesk data. A closer look to the empirical results reveals again the superiority of our proposed model. Our proposed approach performs better than LSTM approach in terms of accuracy (0.779 vs. 0.712). In addressing the RQ 1, we can conclude that, based on the empirical results and with a few exceptions (in terms of some accuracy measures), our approach can beat the benchmark results.

We have also investigated the effect of bitsize dimension in the feature hashing step on the accuracy of the prediction results. As mentioned above, the adverse effects of the collision can be mitigated by increasing the bitsize in the hash table. However, the empirical results show that adjusting the bitsize does not influence the prediction results after some threshold, which was 10 in our case. This can be explained by the fact that the n-grams obtained from the process sequences follow Zipf's law, which states that a small proportion of the input features occur with higher frequencies. This implies that the collisions possibly take place for infrequent variables. The phenomenon can also be observed in protein sequence classification problems [20].

Since the majority of process traces in the BPI_2012 and Helpdesk datasets consist of less than 6 activities, the maximum length of n-grams was defined as 5. The BPI_2013 datasets were an exception. In both datasets we could build 10-grams to gauge the ability of the proposed approach in predicting the eleventh event in the running session. The results suggest that increasing the size of the n-grams does not improve the prediction capability of the model. However, using bigger n-grams leads to computational costs.

From our results we can also conclude that using the ReLu as a non-linear activation function leads to better predictions results. Moreover, we have observed that increasing the number of neurons in the hidden layer does not change the results but makes the algorithm more expensive in terms of computational cost. To answer RQ 2, we suggest that changing the activation function leads to an improvement of prediction results, whereas larger n-grams, larger bitsize and more neurons in the hidden layer do not affect the results in terms of accuracy.

TABLE IV. EVALUATION RESULTS

Dataset	Approach	Feature Hashing		Deep Neural Network Classifier		Results		
		n-grams	Bitsize	Hidden Layers	Activation Function	Accuracy	Precision	Recall
BPI_2012_W_Completed	Proposed Approach	5	10	3 (100,100,100)	Rectifier	**0.827**	**0.806**	**0.828**
		5	10	3 (100,100,100)	Tanh	0.822	0.799	0.822
		5	10	3 (100,100,100)	Maxout	0.824	0.802	0.825
		4	20	3 (200,200,200)	Rectifier	0.825	0.804	0.825
	Other Approaches	[17]				**0.719**	**-**	**0.578**
		[19]				**0.760**	**-**	**-**
		[18]				**-**	**0.658**	**-**
BPI_2012_A	Proposed Approach	5	10	3 (100,100,100)	Rectifier	0.819	0.843	0.819
		5	10	3 (100,100,100)	Tanh	0.816	0.830	0.817
		5	10	3 (100,100,100)	Maxout	0.791	0.798	0.792
		4	20	3 (200,200,200)	Rectifier	0.805	0.809	0.805
	Other Approaches	[17]				**0.801**	**-**	**0.723**
		[19]				**-**	**-**	**-**
		[18]				**-**	**0.832**	**-**
BPI_2012_O	Proposed Approach	5	10	3 (100,100,100)	Rectifier	**0.817**	**0.836**	**0.818**
		5	10	3 (100,100,100)	Tanh	0.811	0.823	0.811
		5	10	3 (100,100,100)	Maxout	0.809	0.819	0.809
		4	20	3 (200,200,200)	Rectifier	0.814	0.821	0.813
	Other Approaches	[17]				**0.811**	-	**0.647**
		[19]				-	-	-
		[18]				-	**0.836**	-
BPI_2013_Incidents	Proposed Approach	5	10	3 (100,100,100)	Rectifier	**0.655**	**0.649**	**0.654**
		5	10	3 (100,100,100)	Tanh	0.632	0.610	0.632
		5	10	3 (100,100,100)	Maxout	0.627	0.601	0.627
		10	20	3 (200,200,200)	Rectifier	0.633	0.614	0.633
	Other Approaches	[17]				**0.714**	**-**	**0.377**
		[19]				**-**	**-**	**-**
		[18]				**-**	**0.735**	**-**
BPI_2013_Problems	Proposed Approach	5	10	3 (100,100,100)	Rectifier	**0.654**	**0.635**	**0.653**
		5	10	3 (100,100,100)	Tanh	0.643	0.621	0.642
		5	10	3 (100,100,100)	Maxout	0.642	0.623	0.641
		10	20	3 (200,200,200)	Rectifier	0.649	0.634	0.649
	Other Approaches	[17]				**0.690**	-	**0.521**
		[19]				-	-	-
		[18]					**0.628**	
Helpdesk	Proposed Approach	2	10	3 (100,100,100)	Rectifier	**0.779**	0.628	**0.779**
		2	10	3 (100,100,100)	Tanh	0.777	**0.631**	0.777
		2	10	3 (100,100,100)	Maxout	0.777	0.627	0.778
		2	20	3 (200,200,200)	Rectifier	0.776	0.629	0.777
	Other Approaches	[17]				-	-	-
		[19]				**0.712**	-	-
		[18]				-	-	-

V. Conclusion

The paper investigated the effectiveness of the multi-stage deep learning approach consisting of deep stacked autoencoders and feedforward neural networks for predicting future process events in the running instance. It is the first use of this approach in the business process prediction domain. To evaluate the predictive capability of our model, we compared it against three recent approaches, two of which used deep LSTM recurrent neural networks. Furthermore, before applying our approach, we used the n-gram approach and the feature hashing technique to build the numerical feature vector from the categorical process event data. The overall objective was to examine the feasibility and impact of applying the proposed approach to process prediction. The experimental results suggest that the proposed model can achieve good results in terms of different classification evaluation measures and outperforms the state-of-the-art approaches in the majority of experiments for predicting the next process events. We have also investigated and discussed the impact of adjusting the parameters of both data pre-processing techniques and deep neural networks on the prediction results.

However, we did not concentrate on adjusting the advanced optimization parameters of deep neural networks which is a topic for future research. In our paper, we focused on the batch (offline) analysis of the process log data. In our future work, we intend to apply the proposed approach for next event prediction by connecting directly to the PAIS which provides the log data. By doing so, we can test the ability of our approach to make "on-the-fly" predictions.

The proposed approach deals with the next event prediction problem which can be extended to forecasting the business process outcomes. Even if there is no crucial need for algorithmic changes, the business process outcome prediction problem requires intensive feature processing work. Applying the proposed multi-stage deep learning approach for various regression problems would also be an interesting future research direction.

Acknowledgment

This research was funded in part by the German Federal Ministry of Education and Research under grant number 01IS12050 (project PRODIGY).

References

[1] W. M. P. van der Aalst, M. H. Schonenberg, and M. Song, "Time prediction based on process mining," Information Systems, vol. 36, pp. 450-475, 4/2011.

[2] W. M. P. van der Aalst, M. Pesic, and M. Song, "Beyond Process Mining: From the Past to Present and Future," 22nd International Conference on Advanced Information Systems Engineering:, CAiSE 2010, Hammamet, Tunisia, June 7-9, 2010, pp. 38-52.

[3] A. Rogge-Solti and M. Weske, "Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays," International Conference on Service-Oriented Computing, 2013, pp. 389-403.

[4] F. Folino, M. Guarascio, and L. Pontieri, "Discovering Context-Aware Models for Predicting Business Process Performances," On the Move to Meaningful Internet Systems: OTM 2012: Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012, Rome, Italy, September 10-14, 2012. Proceedings, Part I, 2012, pp. 287-304.

[5] M. Ceci, P. F. Lanotte, F. Fumarola, D. P. Cavallo, and D. Malerba, "Completion Time and Next Activity Prediction of Processes Using Sequential Pattern Mining," Discovery Science: 17th International Conference, DS 2014, Bled, Slovenia, October 8-10, 2014. 2014, pp. 49-61.

[6] M. Polato, A. Sperduti, A. Burattin, and M. de Leoni, "Time and activity sequence prediction of business process instances," arXiv preprint arXiv:1602.07566, 2016.

[7] A. Bolt and M. Sepúlveda, "Process remaining time prediction using query catalogs," International Conference on Business Process Management, 2013, pp. 54-65.

[8] B. Kang, D. Kim, and S.-H. Kang, "Real-time business process monitoring method for prediction of abnormal termination using KNNI-based LOF prediction," Expert Systems with Applications, vol. 39, pp. 6061-6068, 2012.

[9] B. Kang, D. Kim, and S.-H. Kang, "Periodic performance prediction for real-time business process monitoring," Industrial Management & Data Systems, vol. 112, pp. 4-23, 2012.

[10] A. Leontjeva, R. Conforti, C. Di Francescomarino, M. Dumas, and F. M. Maggi, "Complex symbolic sequence encodings for predictive monitoring of business processes," International Conference on Business Process Management, 2015, pp. 297-313.

[11] C. Di Francescomarino, M. Dumas, M. Federici, C. Ghidini, F. M. Maggi, and W. Rizzi, "Predictive business process monitoring framework with hyperparameter optimization," International Conference on Advanced Information Systems Engineering, 2016, pp. 361-376.

[12] A. Metzger, P. Leitner, D. Ivanović, E. Schmieders, R. Franklin, M. Carro, et al., "Comparing and combining predictive business process monitoring techniques," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 45, pp. 276-290, 2015.

[13] M. Le, D. Nauck, B. Gabrys, and T. Martin, "Sequential Clustering for Event Sequences and Its Impact on Next Process Step Prediction," International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, 2014, pp. 168-178.

[14] M. Le, B. Gabrys, and D. Nauck, "A hybrid model for business process event and outcome prediction," Expert Systems, 2014.

[15] M. Unuvar, G. T. Lakshmanan, and Y. N. Doganata, "Leveraging path information to generate predictions for parallel business processes," Knowledge and Information Systems, vol. 47, pp. 433-461, 2016.

[16] G. T. Lakshmanan, D. Shamsi, Y. N. Doganata, M. Unuvar, and R. Khalaf, "A markov prediction model for data-driven semi-structured business processes," Knowledge and Information Systems, vol. 42, pp. 97-126, 2015.

[17] D. Breuker, M. Matzner, P. Delfmann, and J. Becker, "Comprehensible predictive models for business processes," MIS Quarterly, vol. 40, pp. 1009-1034, 2016.

[18] J. Evermann, J.-R. Rehse, and P. Fettke, "Predicting process behaviour using deep learning," Decision Support Systems, 2017., in press

[19] N. Tax, I. Verenich, M. La Rosa, and M. Dumas, "Predictive business process monitoring with LSTM neural networks," 29th International Conference on Advanced Information Systems Engineering, 2016, in press.

[20] C. Caragea, A. Silvescu, and P. Mitra, "Protein sequence classification using feature hashing," Proteome Science, vol. 10, pp. 14, 2012.

[21] A. Tomović, P. Janičić, and V. Kešelj, "n-Gram-based classification and unsupervised hierarchical clustering of genome sequences," Computer methods and programs in biomedicine, vol. 81, pp. 137-153, 2006.

[22] N. Mehdiyev, J. Krumeich, D. Werth, and P. Loos, "Determination of Event Patterns for Complex Event Processing Using Fuzzy

Unordered Rule Induction Algorithm with Multi-objective Evolutionary Feature Subset Selection," 49th Hawaii International Conference on System Sciences (HICSS), 2016, pp. 1719-1728.

[23] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in Proceedings of the 26th Annual International Conference on Machine Learning, 2009, pp. 1113-1120.

[24] J. Langford, L. Li, and A. Strehl, "Vowpal wabbit online learning project," Technical report, http://hunch.net, 2007.

[25] K. Ganchev and M. Dredze, "Small statistical models by random feature mixing," Proceedings of the ACL08 HLT Workshop on Mobile Language Processing, 2008, pp. 19-20.

[26] N. F. Da Silva, E. R. Hruschka, and E. R. Hruschka, "Tweet sentiment analysis with classifier ensembles," Decision Support Systems, vol. 66, pp. 170-179, 2014.

[27] G. Forman and E. Kirshenbaum, "Extremely fast text feature extraction for classification and indexing," Proceedings of the 17th ACM conference on Information and knowledge management, 2008, pp. 1221-1230.

[28] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science, vol. 313, pp. 504-507, 2006.

[29] J. V. Tu, "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes," Journal of Clinical Epidemiology, vol. 49, pp. 1225-1231, 1996.

[30] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," Proceedings of the 23rd international conference on Machine learning, 2006, pp. 161-168.

[31] R. Caruana, N. Karampatziakis, and A. Yessenalina, "An empirical evaluation of supervised learning in high dimensions," International conference on Machine learning, 2008, pp. 96-103.

[32] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Networks, vol. 61, pp. 85-117, 1/ 2015.

[33] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," Neural Computation, vol. 18, pp. 1527-1554, 2006.

[34] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," International Conference on Machine Learning (3), vol. 28, pp. 1139-1147, 2013.

[35] A. Candel, V. Parmar, E. LeDell, and A. Arora, "Deep Learning with H2O," H2O. ai Inc., 2016.

[36] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio, "Maxout Networks," International Conference on Machine Learning (3), vol. 28, pp. 1319-1327, 2013.

[37] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in Neural networks: Tricks of the trade, ed: Springer, 2012, pp. 9-48.

[38] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," Advances in Neural Information Processing Systems, 2011, pp. 693-701.

[39] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," arXiv preprint arXiv:1212.5701, 2012.

[40] H. Wickham and R. Francois, "dplyr: A grammar of data manipulation," R package version 0.4, vol. 1, p. 20, 2015.

[41] R. Barga, V. Fontama, W. H. Tok, and L. Cabrera-Cordon, Predictive analytics with Microsoft Azure machine learning: Springer, 2015.

[42] B. F. van Dongen, "BPI Challenge 2012. Eindhoven University of Technology. Dataset. http://dx.doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f,".

[43] W. Steeman, "BPI Challenge 2013, incidents. Ghent University. Dataset. http://dx.doi.org/10.4121/uuid:500573e6-accc-4b0c-9576-aa5468b10cee,".

[44] W. Steeman, "BPI Challenge 2013, open problems. Ghent University. Dataset. http://dx.doi.org/10.4121/uuid:3537c19d-6c64-4b1d-815d-915ab0e479da,".

[45] W. Steeman, "BPI Challenge 2013, closed problems. Ghent University. Dataset. http://dx.doi.org/10.4121/uuid:c2c3b154-ab26-4b31-a0e8-8f2350ddac11,".

[46] I. Verenich, "Helpdesk," 10.17632/39bp3vv62t.1, 2016.