Business & Information Systems Engineering

A Novel Business Process Prediction Model Using a Deep Learning Method

--Manuscript Draft--

Manuscript Number:	BUIS-D-17-00238
Full Title:	A Novel Business Process Prediction Model Using a Deep Learning Method
Article Type:	Department Enterprise Modeling & IS
Corresponding Author:	Nijat Mehdiyev, M.Sc. German Research Center for Artifcial Intelligence (DFKI) Saarbruecken, Saarland GERMANY
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	German Research Center for Artifcial Intelligence (DFKI)
Corresponding Author's Secondary Institution:	
First Author:	Nijat Mehdiyev
First Author Secondary Information:	
Order of Authors:	Nijat Mehdiyev
	Joerg Evermann
	Peter Fettke
Order of Authors Secondary Information:	
Funding Information:	Bundesministerium für Bildung und Forschung (01IS12050) — Mr. Nijat Mehdiyev
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 with respect to the execution of business processes. The goal of the approach proposed in this paper is to predict the next business process event from the completed activities of the running process instance, based on the execution log data from previously completed process instances. By predicting business process events, companies can initiate timely interventions to address undesired deviations from the desired workflow. We propose a multi-stage deep learning approach that formulates the next business process event prediction problem as a classification problem. Following a feature pre-processing stage with n-grams and feature hashing, we apply a deep learning model consisting of an unsupervised pre-training component with stacked autoencoders and a supervised fine-tuning component. Experiments conducted on a variety of business process log datasets show that the proposed multi-stage deep learning approach provides promising results. We also compared our results to existing deep recurrent neural networks and conventional classification approaches. Furthermore, we address the identification of the most suitable hyperparameter configuration for the proposed approach, and the handling of the imbalanced nature of business process event datasets.

Powered by Editorial Manager® and ProduXion Manager® from Aries Systems Corporation

Dear Dr. Jelena Zdravkovic,
Dear Dr. Dimitris Karagiannis,
Editors - BISE Department Enterprise Modeling and Enterprise IS,

Thank you very much for the opportunity to fast track our CBI conference paper to the Business & Information Systems (BISE) journal:

Mehdiyev N, Evermann J, Fettke P (2017): A multi-stage deep learning approach for business process event prediction. In: Proceedings-2017 IEEE 19th Conference on Business Informatics, CBI 2017. pp 119–128

This paper was presented at CBI 2017 and received very positive feedback from reviewers, the audience and some colleagues. Thus, we are very excited about extending the paper and having the opportunity to submit this extended version to your BISE department, "Enterprise Modeling and Enterprise IS".

We would like also to express our gratitude to the CBI 2017 Program Chairs - Peri Loucopoulos, Oscar Pastor and Jelena Zdravkovic. Below, we provide a list of changes.

We look forward to hearing from you with the review reports soon.

Sincerely,

Nijat Mehdiyev
Joerg Evermann
Peter Fettke

Description of changes

We not only significantly extended the writing, discussion and presentation in this version of the paper, but, more importantly, undertook a considerable amount of additional research (hyperparameter optimization, data rebalancing, additional evaluation and comparison). We provide details of the changes by section:

1. Introduction

- The introduction section of the paper was significantly expanded. In particular, among other changes, the role and importance of the enterprise information systems for business process prediction problem were emphasized.
- The discussion of our research contribution was considerably expanded and highlighted.

2. Related Work

- Additional recent studies were introduced and discussed in considerable depth.
- The related work were categorized and details of individual papers were provided.
- Current research gaps were identified and the addressed issues in the present paper were presented.

3. Proposed Approach

- The description of stacked autoencoder based deep learning approach was expanded, corrected, fine-tuned and presented in an easier-to-understand way.
- The theoretical arguments for the expected superiority of our proposed model were supported with additional discussion and references to the literature.

- The elements of the unsupervised pretraining and supervised fine-tuning stages of the deep learning approach were described more elaborately and precisely.

4. Evaluation

- Additional classification measures such as micro-averaged F-measure, micro-averaged Matthews Correlation Coefficient (MCC) and area under ROC curve (AUC) are described, discussed, computed, and analyzed in order to discuss diverse aspects of evaluations.
- One additional research question (total of 3) is addressed. This research question focuses on the correct classification of rare events or rare activities. Rare events present challenges for any classifier, but are often of great business relevance as they typically signal process exceptions, error compensations or other critical deviations from the typical process path. In particular, we show that our approach can be combined with a modern form of neural-network based data rebalancing or augmentation to considerably improve its prediction/classification performance for rare, but important, events. Specifically:
    - Radial Basis Function Neural Networks were applied to generate semi-artificial examples of the minority class to balance the classes.
    - A comparison of our proposed approach with balanced and imbalanced data was conducted. ROC Curve Graphs were used to assess the performance of the classifiers.
- The proposed model was additionally compared against conventional, generic (i.e. not process specific) classification approaches such as Random Forests, Naïve Bayes, Support Vector Machines and C4.5 algorithm.
- We implemented a random search based hyperparameter optimization approach to identify the best hyperparameter configuration of the deep learning. A detailed overview was provided in a separate subsection. This has led to considerable improvements in the prediction quality of our approach, further strengthening our results and the contribution of this paper.

Title:

A Novel Business Process Prediction Model Using a Deep Learning Method

Authors:

1. Nijat Mehdiyev (M.Sc.)
Institute for Information Systems (IWi)
German Research Center for Artificial Intelligence (DFKI) and Saarland University
Campus D3.2, 66123
Saarbruecken, Germany
nijat.mehdiyev@iwi.dfki.de

2. Joerg Evermann (Prof. Dr.)
Memorial University of Newfoundland
310 Elizabeth Avenue, NL, A1B 3X5
St. John's, NL, Canada
jevermann@mun.ca

3. Peter Fettke (Prof. Dr.)
Institute for Information Systems (IWi)
German Research Center for Artificial Intelligence (DFKI) and Saarland University
Campus D3.2, 66123
Saarbruecken, Germany
peter.fettke@iwi.dfki.de

Blinded Manuscript

A Novel Business Process Prediction Model Using a Deep Learning Method

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 with respect to the execution of business processes. The goal of the approach proposed in this paper is to predict the next business process event from the completed activities of the running process instance, based on the execution log data from previously completed process instances. By predicting business process events, companies can initiate timely interventions to address undesired deviations from the desired workflow. We propose a multi-stage deep learning approach that formulates the next business process event prediction problem as a classification problem. Following a feature pre-processing stage with n-grams and feature hashing, we apply a deep learning model consisting of an unsupervised pre-training component with stacked autoencoders and a supervised fine-tuning component. Experiments conducted on a variety of business process log datasets show that the proposed multi-stage deep learning approach provides promising results. We also compared our results to existing deep recurrent neural networks and conventional classification approaches. Furthermore, we address the identification of the most suitable hyperparameter configuration for the proposed approach, and the handling of the imbalanced nature of business process event datasets.

Keywords: Process prediction; Deep learning; Feature hashing; N-grams; Stacked autoencoders.

1. Introduction

Since firms adopt similar products and technologies, high-performance business processes are one of the last points of differentiation (Davenport and Harris 2007). Operationalizing predictive analytics by embedding their insights to the enterprise processes can boost business value (LaValle et al. 2011). Various process aware enterprise information systems such as Workflow Management Systems (WMS), Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), Supply Chain Management (SCM), Knowledge Management (KM), Product Management (PM), Incident Management (IM), Case Handling Systems, among others, can log the activities generated during process execution (van der Aalst et al. 2011). Such logs can act as a valuable source for predictive analytics, which improves decision making by providing insights into the process behavior. An effective design and implementation of such predictive approaches are important to ensure that business activities will run in a desired manner by avoiding predicted failures and deviations from the intended process behavior. Detecting process anomalies in real-time, analyzing the behavioral patterns of customers to make tailored offers, risk management by predicting compliance violations, effective resource allocation etc., are some of the use cases that can be addressed by process data driven predictive analytics (Evermann et al. 2017).

Business process prediction deals with predicting a target variable of interest after extracting features from business process log data. Predicting continuous target values, such as remaining process execution time, requires the application of regression algorithms. On the other hand, since predicting the next process events in the running case, the outcome of a process instance, the violation of service level agreements, etc. are problems where the target values are discrete, they are regarded as classification problems.

In this study, we focus on predicting the next business process event, considering the past events of the running process instance, based on execution log data from previously completed process instances. The ability to identify the next events in the running process instance, given the previously performed activities, is a vital issue in process analytics since such analytical information allows the analysts to intervene proactively to prevent undesired behavior. Inspired by recent successful applications of deep learning, we address this problem with a multi-stage deep learning approach. The main contribution of our research is threefold:

1. This study applies, for the first time in the business process management domain, a deep learning approach consisting of an unsupervised pre-training stage with stacked autoencoders, and a supervised fine-tuning stage for the multi-class classification problem.
2. This work improves on prior research by incorporating an extensive data pre-processing stage. We use an n-gram representation and feature hashing approach to build the numerical feature vectors from business process event log data. According to our knowledge, no prior studies have applied feature hashing in this domain.
3. We also address the hyperparameter optimization of our deep learning approach and the complexities related to the imbalanced nature of the business process data with the aim of boosting the precision of the analytics results.

We follow the "exaptation" (extend known solutions to new problems) type of Design Science Research (DSR) knowledge contribution by adopting successful solutions (stacked autoencoders based deep learning, feature hashing) to build innovative predictive analytics models for process data in enterprise information systems (Gregor and Hevner 2013). Integrating data analytics with the enterprise information systems is one of the crucial and emerging trends in the IS research domain (Sun et al. 2015). By incorporating the proposed method into enterprise information systems and decision making mechanisms, it is possible to provide both long-term strategic predictions in batch mode, and to monitor running process instances in real-time. In our approach, the enterprise information systems act as a producer of the inputs for predictive analytics models by providing the required log data, and also as consumers of the knowledge and insights derived from these models.

The remainder of the paper is organized as follows: Section 2 introduces related work on business process prediction. Section 3 provides a broad description of the components of the proposed approach. It discusses the data pre-processing stages, n-gram encoding and feature hashing, and the structure of the deep learning model. Section 4 outlines the experiment settings, the structure of datasets and our empirical results. Section 5 concludes the paper with a discussion and summary.

2. Related Work

A growing body of the literature has recently examined the application of various machine learning approaches in business process management. We provide an overview of these approaches, categorizing them according to the type of the target variable (discrete vs. continuous) they attempt to predict, and discuss the problem types within these categories.

The first category comprises approaches that deal with regression problems by attempting to predict the continuous outputs. Forecasting the remaining processing time of incomplete cases is the most frequently addressed problem in this category. van Dongen et al. (2008) applied

non-parametric regression approaches to compute the remaining cycle time on the data recorded in event logs. Polato et al. (2016) implemented both simple and support vector regression methods to forecast the remaining time of running process instances. Rogge-Solti and Weske (2013) proposed a stochastic Petri net with generally distributed transitions to predict remaining process execution time based on elapsed time since the last observed event. To overcome the shortcomings of conventional regression approaches in predicting remaining time to completion, van der Aalst et al. (2011) presented an annotated transition system that represents an abstraction of the process with time annotations. Folino et al. (2012) introduced a hybrid predictive clustering tree (PCT) and multiple performance annotated Finite State Machine (FSM) models for remaining time prediction. Senderovich et al. (2017) applied linear regression, random forests and XGBoost approaches for remaining time prediction after obtaining the features related from specific process instances and global process models.

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. (Kang et al. 2012 a,b,; Leontjeva et al. 2015; Metzger et al. 2015; Di Francescomarino et al. 2016). The following studies address the next process event prediction that we investigate in this paper. A multi-stage model, which starts by clustering event sequences using the k-mean algorithm combined with sequential alignment, builds individual Markov models of different orders on the obtained clusters (Le et al. 2014). Experiments were conducted on records of processes obtained from a telecommunication company. Another approach, by Le et al. (2017), uses sequential k-nearest neighbor classification and an extension of Markov models to predict the next process steps by considering temporal features. Using the same process log data as Le et al. (2014), they showed the superiority of this model over Markov and Hidden Markov Models (HMM). Unuvar et al. (2016) proposed a decision tree based model to predict the next activity in the running instance of business processes that contain parallel execution paths. Five different models for representing the path attribute of the execution trace were presented and experiments were conducted 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 (Lakshmanan et al. 2015). A simulated dataset was used to verify the prediction accuracy. Somewhat similar to a Markov model, a probabilistic finite automaton (PFA) based on Bayesian regularization by Breuker et al. (2016) uses the Expectation Maximization (EM) approach to estimate the relevant process parameters. The evaluation process was carried out using both a simulated dataset and real-life datasets (the publicly available BPI Challenge 2012 and BPI Challenge 2013 data). Márquez-Chamorro et al. (2017) proposed an evolutionary rule based approach to predict the events of interest after encoding the features using a window technique. The proposed model was evaluated using the BPI Challenge 2013 and health services datasets.

More recent work is moving away from explicit models to deep learning approaches. In the first approach to apply deep-learning, Evermann et al. (2017) applied recurrent neural networks (RNN) with Long Short-Term Memory (LSTM) after transforming the input features using word embeddings. The accuracy improvement potentials 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. Also applying the LSTM approach but only considering the occurrence sequence of the activities and their timestamps, Tax et al.

(2017) 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 also conducted experiments on another publicly available helpdesk dataset from an Italian software company (Tax et al. 2017). Our own earlier, initial study is also based on a deep learning approach (author citation, 2017). However, this paper significantly expands on the earlier paper by improving the hyperparameter optimization, assessing and improving prediction performance on imbalanced datasets (which are typically problematic for classifiers), and additional evaluation and comparison.

One of the main differences between the recent studies by Evermann et al. (2017) and Tax et al. (2017) and our approach lies in the transformation of the sequential process data to the neural network input features (for the predetermined prefix size) that are used to train the models. The majority of existing approaches use the simple index encoding method to build the feature vector from sequence data, but this does not consider the interdependencies among the sequential event data (Leontjeva et al. 2015; Márquez-Chamorro et al. 2017; Senderovich et al. 2017). To tackle this problem, we use an n-gram based encoding schema. Depending on the size of the event space, the n-gram based approach can lead to a very high dimensional feature space. Therefore, we apply a feature hashing technique to obtain a reasonable data size. Another important feature of our study, and one which significantly improves on our approach in (author citation, 2017), is the application of an algorithmic deep learning hyperparameter optimization technique, which has not been used in previous deep learning approaches for business process event prediction. Since the optimal hyperparameter configuration significantly affects the classification results, testing models with only a few hyperparameter combination variations (manual search) is likely to lead to suboptimal results. Finally, almost no study except (Márquez-Chamorro et al. 2017) addresses the classification problem for an imbalanced dataset. Identification of rare events might have important business implications. We address this problem by synthesizing new instances for the minority class using neural networks and thereby balancing the training data set.

3. Proposed Approach

We formulate the prediction of the next business process events as a classification problem. Figure 1 shows an overview of the proposed approach. We apply deep learning algorithms on a feature matrix extracted from various process characteristics such as control flow, data flow, resource, and organizational perspectives, after a thorough data pre-processing stage. The proposed approach starts with the reconstruction of business process events (control flow) obtained from event log data with a sliding window technique and encoded in letters into the n-gram feature representation (Figure 1). Next, a feature hashing algorithm maps the extracted n-grams to hash keys. The hashed feature matrix is then extended by adding the relevant data and resource features. Once the merged feature matrix is available, the proposed deep learning method is applied to predict the next business process events. It consist of two components, an unsupervised layerwise pre-training component that aims to produce higher level feature representations, and a supervised fine-tuning of the whole network for the multiclass classification that adds an output layer on top of the stack.

3.1. 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 sequence of events. Events contain attributes

that describe their characteristics (XES Standard 2016). 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 understood here as predicting the executing activity and lifecycle transition of the next event in the running trace considering the sequence of past events for a predefined prefix length from that particular trace.

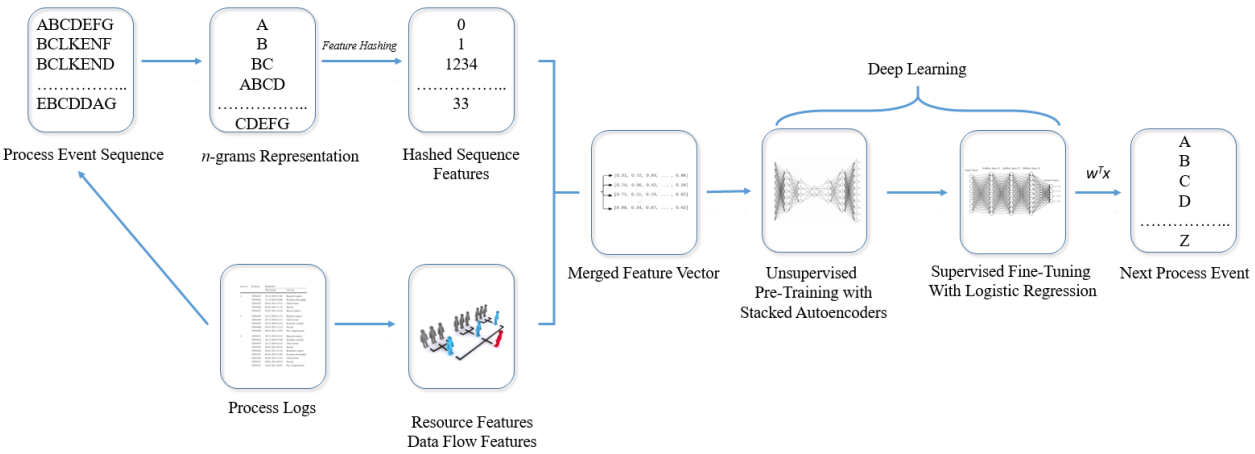


Figure 1 The stages of the proposed approach

3.2. Data Pre-processing

Prior studies rely heavily on assessing the performance of the proposed approaches but, with a few exceptions, pay little attention to data pre-processing. However data preparation comprising various stages such as data cleaning, encoding, dimensionality reduction, feature extraction etc., significantly influences the predictive ability of classifiers.

3.2.1. N-gram encoding

The initial step of our approach is the sequence encoding, which is the conversion of character strings (business process events; specifically the executing activity for each event) into numerical input features. Leontjeva et al. (2015) provided a comparative analysis of various sequence encoding schemas for business process outcome prediction. Choosing an appropriate sequence encoding method is a crucial issue since it significantly influences the accuracy of the machine learning approaches. Process event sequence data contains intrinsic relationships and interdependencies among the individual events. We choose n-gram encoding as a suitable approach for modelling such dependencies due to its ability to consider the relationships between neighboring elements by building all contiguous subsequences (Caragea et al. 2012). We use the combination of n-grams of different sizes 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 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 possible unique n-grams for the event universe 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, GCL, …., DAM} etc. We consider the combination of n-grams of pre-defined sizes. The size of our input feature space e.g. 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 would be:

$$N_{total\_features} = 15+15^2+15^3+15^4+15^5 = 813{,}615$$

Due to its completeness (the alphabet is a-priori known, in our case comprising the set of unique executing activities of the process events), domain independence, efficiency (one pass processing) and simplicity, the n-grams approach has been applied to various problems ranging from protein classification to information retrieval (Tomović et al. 2006). Predictions relying on n-gram event data require no additional preprocessing stages such as sequence alignment. Moreover, the letter n-grams method is also very effective due to its ability to not only encode the letters but also order them automatically. However, as it can be easily inferred from the above example, the major drawback of the n-gram representation is that the size of generated input feature set for classification problems tends to be extremely large: The number of features increases exponentially with the n-gram length that is used. Using all the generated features would overload the prediction system by leading to extremely high computational costs and the sparsity of the input would lead to reduced accuracy. To address this challenge we have adopted a dimensionality reduction technique, feature hashing, to reduce the size of n-gram feature vectors.

3.2.2. Feature Hashing

Feature hashing is an effective dimensionality reduction method that scales up a classification algorithms by mapping the high dimensional input space into a low dimensional space (Weinberger et al. 2009). Feature hashing has already found successful applications in natural language processing (NLP), such as news categorization, spam filtering, sentiment analysis in social networks and different areas of bioinformatics (Forman and Kirshenbaum 2008; Ganchev and Dredze 2008; Caragea et al. 2012; Da Silva et al. 2014). The main idea of feature hashing is to use the hash functions to map n-grams of events to feature vectors which can be passed to the classification approach to train the model. The formal definition is as follows:

> **Definition 2:** *Given a set of hashable features N, which are the n-grams obtained from the business process event sequences, h is the first hash function, h:N → {1,……, m} and ξ is the second hash function, ξ:N → {±1}. The combined feature hashing function* $\Phi^{(h,\xi)}$ *maps the high dimensional input vector of size* ***d*** *into a low-dimensional feature vector* ***m*** *where* ***m*** *<* ***d.*** *The i-th element of the* $\Phi^{(h,\xi)}(x)$ *is given as:* $\Phi_i^{(h,\xi)}(x) = \sum_{j:h(j)=i} \xi(j) x_j$ *where j=0,…, d and i=0,…., m.*

Applying feature hashing not only reduces the training computational costs due to the reduced feature dimensionality but also conserves memory. However, dimensionality reduction via feature hashing can lead to information loss due to hash collisions, i.e. the mapping of many n-grams to the same hash keys. Larger hash tables, implying larger bit sizes of the hash function, can prevent this problem (Weinberger et al. 2009). Bit size determines the numbers of the bits when creating the hash table. The optimal bit size depends on the size of the n-gram vocabulary. A descriptive analysis of the n-grams obtained from the process sequences shows that they

follow Zipf's law (Evermann et al. 2017). This implies that a small proportion of the input features occur with higher frequencies. Hence, the collisions possibly take place for infrequent variables that will incur low information loss (Caragea et al. 2012). The phenomenon can also be observed in protein sequence classification problems (Caragea et al. 2012). As a reasonable trade-off between dimensionality reduction and information loss, we use the 32 bit murmurHash function (Langford et al. 2007) as hash function $h,$. The binary hash function ξ is included to ensure that the hash kernel is unbiased (Weinberger et al. 2009).

3.3. Deep Learning Model

Artificial neural networks (ANN) offer a number of advantages over alternative machine learning approaches for supervised learning tasks, including less need for formal statistical modelling, the capability to detect complex non-linear relationships between predictors and outcomes, the ability to model the interrelationships among the predictor variables, and the availability of a range of training algorithms (Tu 1996). The superior performance of ANN has already been documented in various comparative empirical studies and competitions (Caruana and Niculescu-Mizil 2006; Caruana et al. 2008; Schmidhuber 2015).

The traditional approach to train ANNs, particularly deep neural networks with multiple hidden layers, directly optimizes the loss function through stochastic gradient descent, beginning from randomly initialized weights. However, this results in extremely long training durations and reduces the prediction performance (Vincent et al. 2010). Breakthrough studies beginning in the mid-2000s offered deep learning architectures for training neural networks more effectively (Hinton et al. 2006; Vincent et al. 2008). Deep belief network (DBN), (stacked) autoencoders, denoised (stacked) autoencoders are prominent approaches among deep learning methods. The training process of these deep learning architectures shares commonalities and consists of two stages: (i) unsupervised greedy, layerwise pre-training and (ii) supervised fine-tuning. The main idea of the unsupervised pre-training is to address the need for learning complicated functions that represent high level abstractions by obtaining the network weights through a self-supervised learning that learns the non-linear transformation of the original input. The weights obtained from this stage are then used for training the whole network. The supervised fine-tuning component maps the output data to the pre-trained deep neural network and tries to minimize classification errors with gradient based optimization by adjusting the previously learned weights.

An extensive experimental study showed that the neural networks with unsupervised pre-training component provide better classification results than networks without a pre-training stage because the unsupervised pre-training yields a good initial marginal distribution, captures intrinsic dependencies between variables, outperforms the classical regularization techniques, and acts as a variance reduction technique (Erhan et al. 2010). In this study, we apply stacked autoencoders to extract high level feature representation layerwise in an unsupervised manner. After pre-training with stacked autoencoders, we perform the fine-tuning and relevant classifications using a logistic regression layer after adding an output layer to the obtained stack (see the Figure 2).

3.3.1. Unsupervised Pre-training with Stacked Autoencoders

Autoencoders are the non-linear generalization of the Principal Component Analysis (PCA) that can model non-linear interdependencies among the features of the given dataset (Hinton

and Salakhutdinov 2006). An autoencoder consists of three layers, namely input, hidden and output layers. The hidden layer is referred to as encoding layer while the output layer acts as a decoding layer.

Encoder: The encoder takes the high-dimensional input vector x ∈ [0, 1]^d and maps it to the hidden layer using a non-linear activation function f_θ. Due to its tendency to increase sparsity and reduced tendency of vanishing gradients (Izadyyazdanabadi et al. 2017; Shi and Chu 2017), we adopted the Rectified Linear Unit (ReLu) as an activation function for encoding:

$$h = f_\theta(x) = \text{ReLU}(Wx + b) \quad (1)$$

$\theta = \{W, b\}$ is the parameter set of the encoder where W is a $d' \times d$ weight matrix and b is the bias. $h \in [0, 1]^d$ is the output of the hidden layer representation. **Decoder**: The decoder then maps the hidden layer representation back to the reconstructed vector z ∈ [0, 1]^d through the mapping function $g_{\theta'}$.

$$z = g_{\theta'}(h) = g_{\theta'}(W'h' + b') \quad (2)$$

The main goal of the training is the optimization of parameter sets $\theta = \{W, b\}$ in the encoder and $\theta' = \{W', b'\}$ in the decoder phase respectively, to minimize the reconstruction loss. We used squared error as the reconstruction loss function L:

$$L(x, z) = \|x - z\|^2 = \|x - g(W'(f(Wx + b) + b')\|^2 \quad (3)$$

This optimization problem was solved using the mini batch stochastic gradient descent method.

Stacked autoencoders are a greedy layer-wise approach which conducts multi-phase feature extraction by using the features extracted by one autoencoder, represented by its hidden layer, as input of another, following autoencoder (left side of Figure 2) The stacked autoencoders are trained independently to obtain the initial weights for the next stage, supervised fine-tuning.

3.3.2. Supervised Fine-Tuning

After unsupervised reconstruction based learning of the network weights, logistic regression is applied to fine-tune the weights after mapping the output to class labels (right side of Figure 2). To perform such a training, the decoding parts of the stacked autoencoders are removed and the logistic regression layer is added on top of the trained encoding layers. Since we deal with a multi-class classification problem, the added layer uses Softmax (multinomial logistic regression) units to estimate the probabilities of the classes:

$$P(y = j|x) = \frac{e^{\theta_j}}{\sum_{i=1}^{k} e^{\theta_i}} \quad (4)$$

The probability of the target class y being class j, given the input x, is calculated from the input vector x and a set of weighting vectors w_j, where $\theta_j = w_j^T x$ denotes the inner product of w_j and x.

The combined network is trained by using the usual multi-layer perceptrons to minimize the prediction error. We use stochastic gradient descent (SGD) to minimize the cost 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 (LeCun et al. 2012; Goodfellow et al. 2013).

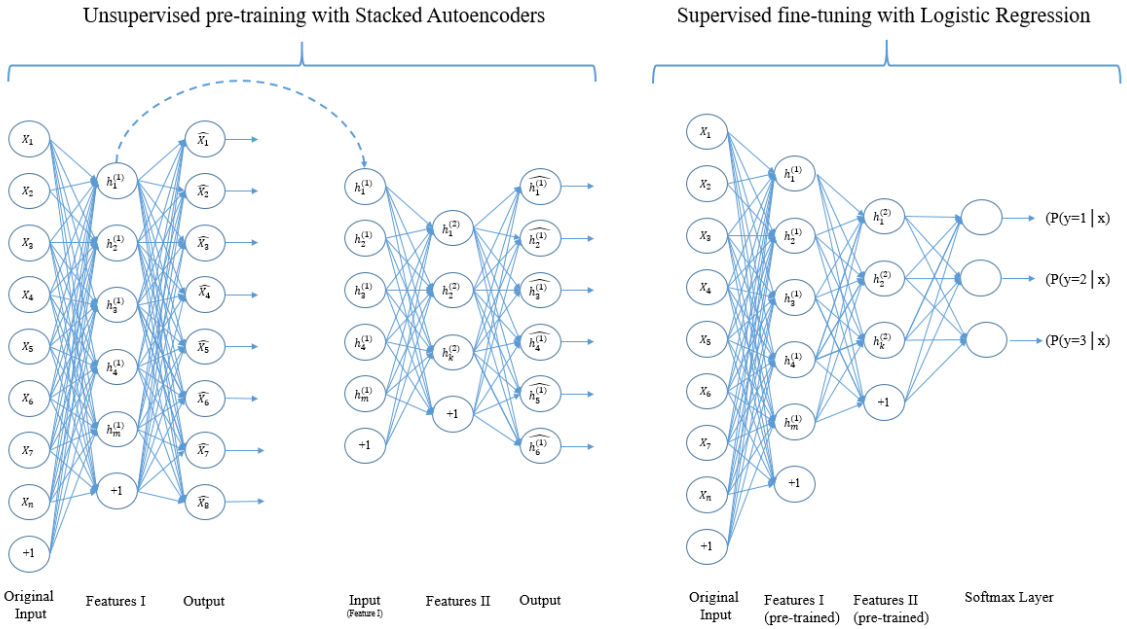


Figure 2 Stacked autoencoders based deep learning. Unsupervised pre-training on the left, supervised fine-tuning on the right.

4. Evaluation

To gauge the effectiveness of the proposed deep learning, we conducted a range of experiments with different datasets, experimental settings and evaluation purposes. In particular, we investigated the following research questions:

- **RQ1:** Does the proposed multi-stage deep learning approach provide superior results in terms of different evaluation measures compared to existing classification approaches?
- **RQ2:** Does the proposed multi-stage deep learning approach outperform the benchmark LSTM based deep learning approaches by Evermann et al. (2017) and Tax et al. (2017) and probabilistic finite automaton (PFA) based on Bayesian regularization by Breuker et al. (2016) for next event prediction?

Business processes often contain rare activities that are not on the typical execution path. Typically, these activities signal process exceptions, process escalation or compensatory tasks. This leads to imbalanced event logs, where some set of events is highly prevalent and another set of events is only sparsely represented. Such imbalanced event logs present a challenge for training many classifiers. However, while rare, these activities are typically highly relevant in a business context, precisely because they signal exceptional process states or execution paths. This in turn implies that it is important for classifiers to correctly classify or predict such important but rare events. Hence, we are interested also in the following research question:

- **RQ3:** Can process prediction with a multi-stage deep learning approach benefit from the application of Radial Basis Function (RBF) neural networks for data balancing that is assumed to help to improve the prediction of rare business process events?

Our 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 (Wickham and Francois 2015). The self-developed Java-based application was used to build the n-grams from the business process event data. The feature hashing approach was carried out using the Microsoft Azure Machine Learning platform which implements the Vowpal Wabbit library (Langford et al. 2007; Barga et al. 2015). Both the pre-trained stacked autoencoders and the supervised deep learning part were created on the H20 open source deep learning platform (Candel et al. 2016). The experimentations for traditional classification techniques were performed using the Weka tool (Hall et al. 2009).

4.1. Datasets

The experiments were conducted using real-life datasets, the BPI Challenge 2012 (van Dongen 2012), BPI Challenge 2013 (W. Steeman 2013), and Helpdesk (Verenich 2016) data. Table 1 provides an overview of the number of unique event types and total number of events in the datasets. The number of unique event types also indicates the number of output classes in our multi-class classification problem.

Table 1 Characteristics of dataset

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 2012** dataset comprises event log data from 262.000 events for 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 Evermann et al. (2017), Breuker et al. (2016), Tax et al. (2017) used only the *completion* events, we filter out the events 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.

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 subsets: The incident management dataset encompasses 7554 cases with 65534 events of 11 unique event types. The open problems dataset contains 819 cases with 2351 events 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 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.

The BPI Challenge datasets not only provide the timestamp of the event occurrence and process trace IDs, but also describe various additional resource and case specific information. This information was also considered in modelling. The BPI Challenge 2012 data provides both organizational information such as the identification number of the resources initiating events, and case specific information such as the amount of the requested loan. The BPI Challenge 2013 datasets contain information about the priority of the problems and incidents, originating 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 the hashed n-gram features. As mentioned above, after generating the feature vectors from the sequence of the activities through n-grams and feature hashing approaches, we append any additional information provided by the log to the feature vector.

4.2. Evaluation Metrics

To evaluate the effectiveness of our deep learning approach and to compare it to alternative classification algorithms, we computed different classification quality metrics such as average accuracy, averaged precision, average recall, average F-measure, and Matthews Correlation Coefficient (MCC) and the area under the ROC curve (AOC) (Table 2) which were adapted to a multi-class classification problem.

Table 2 Evaluation metrics for multi-class classification. l is the number of classes, s_i is the true size of class i (the number of events of class *i*) and $n = \sum_{i=1}^{l} s_i$ is the total size of the dataset.

Metrics	***Formula***
Accuracy	$\frac{1}{n}\sum_{i=1}^{l} s_i \frac{tp_i + fn_i}{tp_i + fn_i + tf_i + fp_i}$
Precision	$\frac{1}{n}\sum_{i=1}^{l} s_i \frac{tp_i}{tp_i + fp_i}$
Recall	$\frac{1}{n}\sum_{i=1}^{l} s_i \frac{tp_i}{tp_i + fn_i}$
F-Measure	$\frac{1}{n}\sum_{i=1}^{l} s_i \frac{precision_i \times recall_i}{precision_i + recall_i}$
MCC	$\frac{1}{n}\sum_{i=1}^{l} s_i \frac{tp_i \times tn_i - fp_i \times fn_i}{\sqrt{(tp_i + fp_i)(tp_i + fn_i)(tn_i + fp_i)(tn_i + fn_i)}}$
AUC	$\frac{1}{n}\sum_{i=1}^{l} s_i \int_0^1 tpr_i \, d(fpr_i)$

In these formulas, tp_i (true positives for class *i*) is the number of events of class *i* that have been classified or predicted as being of class *i*. fp_i (false positives) is the number of events not of class *i* that have been classified (predicted) as being of class *i*. tn_i (true negative) is the number of events not of class *i* that have been classified (predicted) as not of class *i* and finally fn_i (false negatives) is the number of events of class i that have been classified (predicted) as not of class *i*. tpr_i is the true positive rate and fpr_i the false positive rate for class *i*. Accuracy is defined as the proportion of correctly predicted instances of all instances. Precision determines how many activities were correctly classified for a particular class, given all prediction of that class. Recall

is the true positive rate for a particular class. The F-Measure is the harmonic weighted mean of the precision and recall. MCC is referred as the correlation between the actual target values and predicted classifications. AUC is the area under the ROC (receiver operating characteristic) curve. We computed these measures for each individual class and obtained the overall value by summing up their scores, weighted by the true class size.

80% of each dataset was used to train the algorithms and 20% were used for testing purposes. The test results were used to compare the approaches. We used the training data for both unsupervised pre-training and supervised fine tuning of our deep learning model. 10-fold cross validation was used for training the proposed model. For 10-fold cross-validation, the dataset is partitioned into the 10 disjoint subsets. Both training and testing are carried out 10 times. During each iteration, one partitioned subset is used for testing purposes whereas the others serve as input for training the classifier. This procedure is important for identification of best hyperparameter configurations (Vincent et al. 2010). The values of relevant measures are calculated from the test results and reported in Section 4.4 below.

4.3. Hyperparameter Optimization

Sophisticated deep neural networks may have more than fifty hyperparameter (Bergstra et al. 2011). Efficient parameter tuning significantly influences the learning process and prediction outcomes. The main idea behind hyperparameter optimization is the identification of the best model parameter configuration from the given hyperparameter space for obtaining accurate models at a reasonable computational cost. In the traditional approach, manual search, experts define some hyperparameter values for different parameters based on their experience and intuitions (such as number of hidden layers, number of neurons, the learning rate etc.) and try to find the best model in terms of different combination of hyperparameter values by conducting multiple training sessions. However, due to the time consuming nature of this approach, only a few combinations of hyperparameter values can be tested (Bergstra et al. 2011). Furthermore, due to the shortcomings of human reasoning in multi-dimensional spaces, it is challenging to achieve globally optimal outcomes (Witt and Seifert 2017).

The brute force approach, grid search, also referred to as exhaustive search, trains the model for every possible combination of hyperparameter values by following a particular stopping criterion. According to Bergstra and Bengio (2012), grid search identifies better hyperparameter configuration than manual search in the same computational time. A vast majority of the studies from the deep learning domain applies the combined manual and grid search where experts define the set of values for the chosen variables manually and the grid search attempts to find the best configuration by assembling the possible value combinations (Larochelle et al. 2007). Such an exhaustive search may suffer from the curse of the dimensionality since the variety of combinations increases exponentially with the number of the hyperparameter (Bergstra et al. 2011). To tackle this problem, Bergstra and Bengio (2012) proposed a new hyperparameter optimization approach known as random search. The main idea is to pick combinations of hyperparameter values randomly and to train the models in the given constraint (number of models or time). Empirical results show that random search outperforms the brute-force grid search in finding the optimal hyperparameter configuration (Bergstra and Bengio 2012).

Therefore, we adopt the random search hyperparameter optimization approach. We defined the parameter ranges for number of hidden layers [3:10], number of neurons in the hidden layers [10:500], sparse data handling [True, False], initial weight distribution [uniform, normal] for the pre-training component, number of training epochs [10:1000], adaptive learning rate (adaptive learning rate time decay factor=0.99 and adaptive learning rate smoothing factor = 1e-8), (initial) learning rate [0.0001:1], annealing rate [$10:10^6$] when adaptive learning is disabled, etc. for both pre-training component and the whole network. We stopped the search when 200 models for a given dataset are trained. The log-loss was used as the early stopping metric for training. The stopping tolerance was defined as 0.001 and the training process is stopped if relative improvement is below this defined threshold. As an example, Table 3 shows the hyperparameter configuration of our proposed deep learning approach that obtained the best classification accuracy for predicting the next business process event in the BPI_2012_A dataset. We performed the random hyperparameter search for all experiments reported in the present paper.

Table 3 Optimal hyperparameter values for BPI Challenge 2012_A dataset

Parameters (pre-training)	***Values***	***Parameters (whole Network)***	***Values***
Number of Neurons (hidden layers)	425:400:390:300	Number of layers	6 (4 hidden)
Initial Weight Distribution	Normal distribution	Epochs	100
Sparse	True	Adaptive Learning	True
Learn Rate	0.005	Adaptive learning rate smoothing factor	1e-8
Momentum	0.9	Adaptive learning rate time decay factor	0.99
Annealing Rate	10^4	Activation	ReLu
		Activation (classification)	Softmax
		Batch size	20
		classifier L2-penalty	0
		Loss Function	Cross-entropy

4.4. Results

The following subsections provide a detailed discussion of empirical results and address the different research questions.

4.4.1. Comparative Analysis (RQ 1 and RQ 2)

We first compared our approach to conventional (i.e. generic or not-process aware) classification algorithms including support vector machines (SVM), random forests, naïve Bayes, k-nearest-neighbours (kNN) and C4.5 decision trees, which are considered to be some of the most powerful and most widely-used data mining algorithms (Wu et al. 2008). Table 4 presents the obtained results (test results).

The evaluation results in terms of different performance measures show that, with a few exceptions, our proposed deep learning approach outperforms conventional, generic classification techniques. In general, the SVM technique shows a better performance than other methods over all three datasets by getting the closest results to our approach. For the BPI 2013 dataset, all techniques except naïve Bayes perform similarly. However, the performance gaps

between our deep learning approach and the alternative methods are quite large for the BPI 2012 and helpdesk datasets.

Table 4 Results obtained from conventional classification approaches and the proposed deep learning approach (higher numbers are better)

	Accuracy	***Precision***	***Recall***	***F-Score***	***MCC***	***AUC***
BPI 2012_A						
SVM	0.817	**0.856**	0.822	**0.817**	0.748	0.895
RF	0.720	0.714	0.721	0.712	0.566	0.888
Naïve Bayes	0.612	0.633	0.612	0.555	0.485	0.772
C4.5	0.708	0.744	0.709	0.705	0.674	**0.931**
Deep Learning	**0.824**	0.852	**0.824**	**0.817**	**0.751**	0.923
BPI2013_Incidents						
SVM	0.652	0.599	0.653	0.622	0.350	0.730
RF	0.615	0.626	0.616	0.524	0.508	0.895
Naïve Bayes	0.576	0.618	0.577	0.590	0.519	0.879
C4.5	0.659	0.558	0.659	0.582	0.564	**0.900**
Deep Learning	**0.663**	**0.648**	**0.664**	**0.647**	**0.583**	0.862
Helpdesk						
SVM	0.715	0.605	0.716	0.652	0.389	0.725
RF	0.601	0.619	0.601	0.606	0.278	0.688
Naïve Bayes	0.631	**0.634**	0.631	0.622	0.323	0.733
C4.5	0.613	0.534	0.614	0.569	0.214	0.602
Deep Learning	**0.782**	0.632	**0.781**	**0.711**	**0.412**	**0.762**

In summary, to answer **RQ1**, we can observe that our proposed deep learning approach is superior to conventional, generic classification methods.

In order to examine **RQ2**, we compared our results against three recent benchmark approaches for next event prediction. The results for all three BPI 2012 datasets suggest that the proposed model outperforms all three approaches (see Table 5). A bigger difference can be observed for the BPI_2012_W_Completed dataset where our approach achieves an accuracy of 0.831 compared to 0.719 and 0.760 in Breuker et al. (2016) and Tax et al. (2017) respectively. The performance gap compared to Breuker et al. (2016) is greatest for recall (sensitivity). The comparison of our results with Evermann et al. (2017) in terms of precision also shows the superior performance of our proposed approach (0.811 vs. 0.658). Only two other studies used the BPI_2012_A and BPI_2012_O datasets to evaluate their models. Our approach outperforms both of those models in terms of all evaluation measures. The approach by Evermann et al. (2017) performs better for the latter two and achieves results close to ours.

The results for the BPI_2013_Incident dataset are mixed. The approach in Breuker et al. (2016) shows higher predictive performance than ours in terms of accuracy (0.714 vs. 0.663). However, our approach performs significantly better in terms of recall (0.664 vs. 0.377). Precision results obtained in Evermann et al. (2017) are also better than for our approach. However, the experiments conducted on the BPI_2013_Problems dataset suggest that our proposed approach delivers superior results compared to all alternatives.

Finally, only Tax et al. (2017) carried out experiments on the helpdesk data. A closer look to the results shows again the superiority of our proposed model. Our approach performs better than LSTM approach in terms of accuracy (0.782 vs. 0.712).

We also note that, since we use random hyperparameter optimization approach instead of manual search as in our previous study (author citation, 2017), the results presented here are a significant improvements over own earlier work (author citation, 2017).

In summary, to answer **RQ2**, we conclude that our proposed approach outperforms existing deep-learning process prediction approaches for most datasets and on most quality metrics.

Table 5 Comparison against benchmark approaches (higher numbers are better)

	Accuracy	***Precision***	***Recall***
BPI 2012_W			
Breuker et al. (2016)	0.719	-	0.578
Evermann et al. (2017)	-	0.658	-
Tax et al. (2017)	0.760	-	-
Proposed Approach	**0.831**	**0.811**	**0.832**
BPI2012_A			
Breuker et al. (2016)	0.801	-	0.723
Evermann et al. (2017)		0.832	-
Proposed Approach	**0.824**	**0.852**	**0.824**
BPI2012_O			
Breuker et al. (2016)	0.811	-	0.647
Evermann et al. (2017)	-	0.836	
Proposed Approach	**0.821**	**0.847**	**0.822**
BPI2013_Incidents			
Breuker et al. (2016)	**0.714**	-	0.377
Evermann et al. (2017)	-	**0.735**	
Proposed Approach	0.663	0.648	**0.664**
BPI2013_Problems			
Breuker et al. (2016)	**0.690**	-	0.521
Evermann et al. (2017)	-	0.628	
Proposed Approach	0.662	**0.641**	**0.662**
Helpdesk			
Tax et al. (2017)	0.712	-	-
Proposed Approach	**0.782**	0.632	**0.781**

4.4.2. Imbalanced Classification (RQ 3)

In unbalanced datasets some classes are significantly underrepresented compared to others. This reduces the effectiveness of the machine learning techniques, especially for detecting the minority class examples (Wang and Yao 2012). To overcome this, various approaches at the data level (randomly or informatively under/over sampling), algorithm level, cost sensitive learning and boosting methods have been proposed (Sun et al. 2009). Due to their straightforward nature, the resampling approaches are used frequently, but they are unable to increase the information that is required to train the models. Furthermore, undersampling may result in the information loss. To address this issue, the SMOTE (Synthetic Minority Over-sampling Technique) method was proposed. It generates new, non-replicated samples by interpolating neighboring minority class examples, but it also suffers from synthesizing the noisy examples (Huang et al. 2016). Cost sensitive learning techniques are effective approaches to tackle the imbalanced classification problem but require cost information from domain experts. Huang et al. (2016) suggests that applying neural networks to synthesize the samples for minority class is a superior alternative.

In our study, we generate semi-artificial data of the minority class using Radial Basis Function (RBF) neural networks (Robnik-Šikonja 2014). The principle of this approach is to extract Gaussian kernels from the RBF trained with dynamic decay adjustment, and to generate data from each kernel in the required proportions. The details and pseudo-code of the RBF based data generator can be found in (Robnik-Šikonja 2014). This approach was chosen due to its advantages over alternative data generation techniques. Although other data generators consider the relationship between input and target variables, they do not consider dependencies among input variables. Such dependencies are preserved in the RBF based model we have adopted. The RBF approach assumes only the form of the data distribution (Gaussian) but uses extracted distribution parameters to generate data.

The process owners of the "BPI Challenge 2013 Incidents" dataset claim that some employees try to find workarounds which stop the clock from ticking in order to manipulate the total resolution time of an incident. Giving an incident a status of "Wait user" is one of these ways. Although the employees were explicitly requested to avoid using the status of the "Wait user" except for emergency cases, the guideline is sporadically broken. A proactive identification of this misuse thus has a high business relevance. However, the number of occurrences of this minority class is very low compared to the occurrences of events of other classes. To handle this imbalanced classification problem, we formulate the problem as a binary classification problem where the majority class is the set of all other events and the minority class is the "Wait user" event. We then apply our proposed deep learning approach after balancing the classes with the RBF approach. We compare the results against the direct application of our approach to the imbalanced data (without rebalancing). Accuracy is not an appropriate evaluation metric for comparing the classification results in the presence of imbalanced datasets. Even when the classifier detects all majority examples correctly and fails to predict the examples from the minority class, the accuracy will still be high due to prevalence of majority class examples (Han et al. 2005). This would lead to misinterpretation of the model performance. To compare the performance of the models we used the area under the ROC curve (AUC), which is one of the most appropriate measure of the performance for imbalanced data (Bradley 1997). Figure 3 shows ROC curves for the imbalanced data and for the RBF rebalanced data.

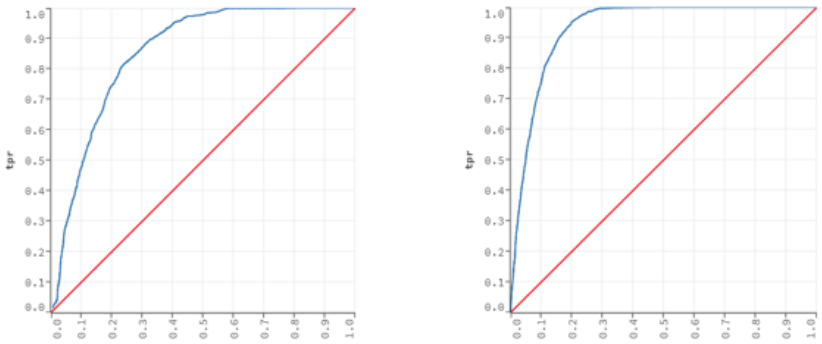


Figure 3 ROC Curves for application to (a) imbalanced and (b) balanced datasets. ROC curves plot the true positive rate (tpr) against the false positive rate (fpr).

The results suggest that balancing the dataset through RBF based data generation affects the effectiveness of the proposed deep learning approach positively by increasing the AUC metric from 0.855 to 0.932.

In summary, to answer **RQ3**, we conclude that RBF based data rebalancing works well in conjunction with our proposed multi-level deep learning prediction approach to improve the prediction of rare, but important, events in a business process.

5. Conclusion

This paper investigated the effectiveness of a stacked autoencoders based deep learning approach for predicting future process events in the running process instance. It is the first application of this approach in the business process prediction domain. To evaluate the predictive performance of our model, we compared it against three recent benchmark approaches, two of which used deep LSTM recurrent neural networks, and conventional classification algorithms. Prior to applying our deep learning model, we used n-gram encoding and feature hashing to build the numerical feature vectors from the categorical process event data by using the sliding window technique. 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 hyperparameter of both data pre-processing techniques and deep neural networks on the prediction results and applied hyperparameter optimization to find the optimal configuration. Finally, we addressed the imbalanced classification problem by employing neural-network based resampling methods.

The successful application of our proposed approach to next event prediction opens up some interesting and important avenues for future research. Our proposed approach, which deals with the next event prediction problem, can also be applied to predicting business process outcomes, such as compliance with service-level agreements, process success or failure or the value of discrete case attributes. Even if there is no crucial need for algorithmic changes, the business process outcome prediction problem requires an intensive feature processing work. Using denoised stacked autoencoders may improve the pre-training results over the ones used here, and is also a subject of future research. Finally, applying the proposed multi-stage deep learning approach for various regression problems, such as time to next event or remaining time to completion, is another interesting future research direction.

References

Author (2017)

Barga R, Fontama V, Tok WH, Cabrera-Cordon L (2015) Predictive analytics with Microsoft Azure machine learning. Apress, 2015

Bergstra J, Bengio Y (2012) Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research 13:281–305.

Bergstra JS, Bardenet R, Bengio Y, Kégl B (2011) Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems. pp 2546–2554

Bradley AP (1997) The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition 30:1145–1159.

Breuker D, Matzner M, Delfmann P, Becker J (2016) Comprehensible Predictive Models for Business Processes. Management Information Systems Quarterly 40:1009–1034.

Candel A, Parmar V, LeDell E, Arora A (2016) Deep learning with h2o. H2O Inc

Caragea C, Silvescu A, Mitra P (2012) Protein sequence classification using feature hashing. Proteome Science 10:1–14.

Caruana R, Karampatziakis N, Yessenalina A (2008) An empirical evaluation of supervised learning in high dimensions. In: Proceedings of the 25th International Conference on Machine learning. ACM, pp 96–103

Caruana R, Niculescu-Mizil A (2006) An empirical comparison of supervised learning algorithms. In: Proceedings of the 23rd international conference on Machine learning. ACM, pp 161–168

Da Silva NFF, Hruschka ER, Hruschka ER (2014) Tweet sentiment analysis with classifier ensembles. Decision Support Systems 66:170–179.

Davenport TH, Harris JG (2007) Competing on analytics : the new science of winning, 1st edn. Harvard Business School Press

Di Francescomarino C, Dumas M, Federici M, et al (2016) Predictive business process monitoring framework with hyperparameter optimization. In: International Conference on Advanced Information Systems Engineering. Springer, pp 361–376

Erhan D, Bengio Y, Courville A, et al (2010) Why Does Unsupervised Pre-training Help Deep Learning? Pierre-Antoine Manzagol. Journal of Machine Learning Research 11:625–660.

Evermann J, Rehse J-R, Fettke P (2017) Predicting process behaviour using deep learning. Decision Support Systems 100:129–140.

Folino F, Guarascio M, Pontieri L (2012) Discovering context-aware models for predicting business process performances. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems." Springer, pp 287–304

Forman G, Kirshenbaum E (2008) Extremely fast text feature extraction for classification and indexing. In: Proceedings of the 17th ACM conference on Information and knowledge management. ACM, pp 1221–1230

Ganchev K, Dredze M (2008) Small statistical models by random feature mixing. In: Proceedings of the ACL08 HLT Workshop on Mobile Language Processing. pp 19–20

Goodfellow IJ, Warde-Farley D, Mirza M, et al (2013) Maxout networks. arXiv preprint arXiv:1302.4389

Gregor S, Hevner AR (2013) Positioning and presenting design science research for maximum impact. Management Information Systems Quarterly 37:337–356.

Hall M, Frank E, Holmes G, et al (2009) The WEKA data mining software: an update. ACM SIGKDD Explorations Newsletter 11:10–18.

Han H, Wang W-Y, Mao B-H (2005) Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. Advances in Intelligent Computing 878–887.

Hinton GE, Osindero S, Teh Y-W (2006) A fast learning algorithm for deep belief nets. Neural Computation 18:1527–1554.

Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. science 313:504–507.

Huang C, Li Y, Change Loy C, Tang X (2016) Learning deep representation for imbalanced classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp 5375–5384

Izadyyazdanabadi M, Belykh E, Mooney M, et al (2017) Convolutional Neural Networks: Ensemble Modeling, Fine-Tuning and Unsupervised Semantic Localization for Intraoperative CLE Images. https://arxiv.org/pdf/1709.03028

Kang B, Kim D, Kang S (2012a) Periodic performance prediction for real- time business process monitoring. Industrial Management & Data Systems 112:4–23.

Kang B, Kim D, Kang S-H (2012b) Real-time business process monitoring method for prediction of abnormal termination using KNNI-based LOF prediction. Expert Systems with Applications 39:6061–6068.

Lakshmanan GT, Shamsi D, Doganata YN, et al (2015) A markov prediction model for data-driven semi-

structured business processes. Knowledge and Information Systems 42:97–126.
Langford J, Li L, Strehl A (2007) Vowpal wabbit online learning project.
Larochelle H, Erhan D, Courville A, et al (2007) An empirical evaluation of deep architectures on problems with many factors of variation. In: Proceedings of the 24th International Conference on Machine learning. ACM, pp 473–480
LaValle S, Lesser E, Shockley R, et al (2011) Big data, analytics and the path from insights to value. MIT Sloan Management Review 52:21.
Le M, Gabrys B, Nauck D (2017) A hybrid model for business process event and outcome prediction. Expert Systems 34:e12079.
Le M, Nauck D, Gabrys B, Martin T (2014) Sequential Clustering for Event Sequences and Its Impact on Next Process Step Prediction. In: International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems. Springer, pp 168–178
LeCun YA, Bottou L, Orr GB, Müller K-R (2012) Efficient backprop. In: Neural networks: Tricks of the trade. Springer, pp 9–48
Leontjeva A, Conforti R, Di Francescomarino C, et al (2015) Complex symbolic sequence encodings for predictive monitoring of business processes. In: International Conference on Business Process Management. Springer, pp 297–313
Márquez-Chamorro AE, Resinas M, Ruiz-Cortés A, Toro M (2017) Run-time prediction of business process indicators using evolutionary decision rules. Expert Systems With Applications 87:1–14.
Metzger A, Leitner P, Ivanovic D, et al (2015) Comparing and Combining Predictive Business Process Monitoring Techniques. IEEE Transactions on Systems, Man, and Cybernetics: Systems 45:276–290.
Polato M, Sperduti A, Burattin A, de Leoni M (2016) Time and Activity Sequence Prediction of Business Process Instances. http://arxiv.org/abs/1602.07566
Robnik-Šikonja M (2014) Data generator based on RBF network. arXiv preprint arXiv:1403.7308
Rogge-Solti A, Weske M (2013) Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In: International Conference on Service-Oriented Computing. Springer, pp 389–403
Schmidhuber J (2015) Deep learning in neural networks: An overview. Neural Networks 61:85–117.
Senderovich A, Di Francescomarino C, Ghidini C, et al (2017) Intra and inter-case features in predictive process monitoring: A tale of two dimensions. In: International Conference on Business Process Management. Springer, pp 306–323
Shi S, Chu X (2017) Speeding up Convolutional Neural Networks By Exploiting the Sparsity of Rectifier Units. https://arxiv.org/pdf/1704.07724
Sun Y, Wong AKC, Kamel MS (2009) Classification of imbalanced data: A review. International Journal of Pattern Recognition and Artificial Intelligence 23:687–719.
Sun Z, Pambel F, Wang F (2015) Incorporating Big Data Analytics into Enterprise Information Systems. In: Information and Communication Technology: Third IFIP TC 5/8 International Conference, ICT-EurAsia 2015, and 9th IFIP WG 8.9 Working Conference, CONFENIS 2015. Springer, pp 300–309
Tax N, Verenich I, La Rosa M, Dumas M (2017) Predictive Business Process Monitoring with LSTM Neural Networks. In: International Conference on Advanced Information Systems Engineering. pp 477–492
Tomović A, Janičić P, Kešelj V (2006) n-Gram-based classification and unsupervised hierarchical clustering of genome sequences. Computer Methods and Programs in Biomedicine 81:137–153.
Tu J V. (1996) Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. Journal of Clinical Epidemiology 49:1225–1231.
Unuvar M, Lakshmanan GT, Doganata YN (2016) Leveraging path information to generate predictions for parallel business processes. Knowledge and Information Systems 47:433–461.
van der Aalst WMP, Schonenberg MH, Song M (2011) Time prediction based on process mining. Information Systems 36:450–475.
van Dongen BF (2012) BPI Challenge 2012. https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f
van Dongen BF, Crooy RA, van der Aalst WMP (2008) Cycle time prediction: When will this case finally be finished? In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems." Springer, pp 319–336
Verenich I (2016) Helpdesk. 10.17632/39bp3vv62t.1, 2016.
Vincent P, Larochelle H, Bengio Y, Manzagol P-A (2008) Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine learning. ACM, pp 1096–1103
Vincent P, Larochelle H, Lajoie I, et al (2010) Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. Journal of Machine Learning Research 11:3371–3408.

W. Steeman (2013) BPI Challenge 2013. https://doi.org/10.4121/uuid:a7ce5c55-03a7-4583-b855-98b86e1a2b07
Wang S, Yao X (2012) Multiclass imbalance problems: Analysis and potential solutions. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 42:1119–1130.
Weinberger K, Dasgupta A, Langford J, et al (2009) Feature hashing for large scale multitask learning. In: Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09. ACM Press, pp 1–8
Wickham H, Francois R (2015) dplyr: A grammar of data manipulation. R package version 04 1:20.
Witt N, Seifert C (2017) Understanding the Influence of Hyperparameters on Text Embeddings for Text Classification Tasks. In: International Conference on Theory and Practice of Digital Libraries. Springer, pp 193–204
Wu X, Kumar V, Quinlan JR, et al (2008) Top 10 algorithms in data mining. Knowledge and Information Systems 14:1–37.
XES Standard (2016) 1849-2016 - IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams. http://www.xes-standard.org/.

Figure 1

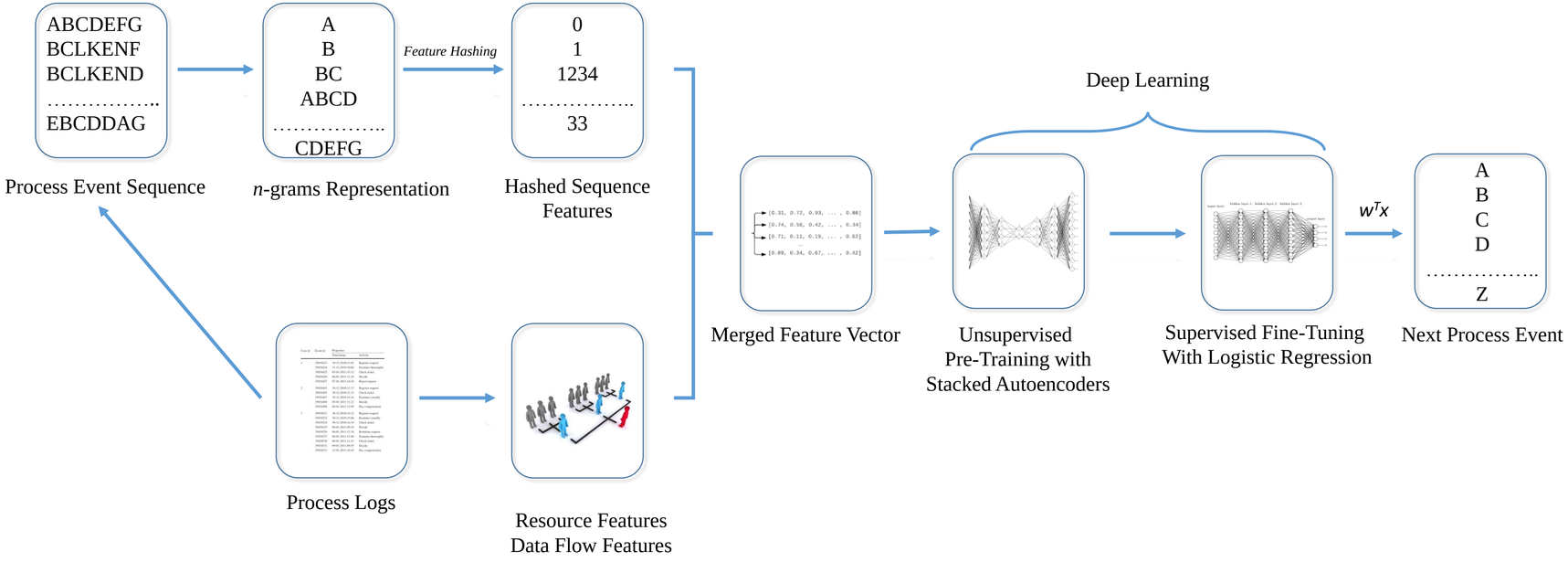


Figure 1 The stages of the proposed approach

Figure 2

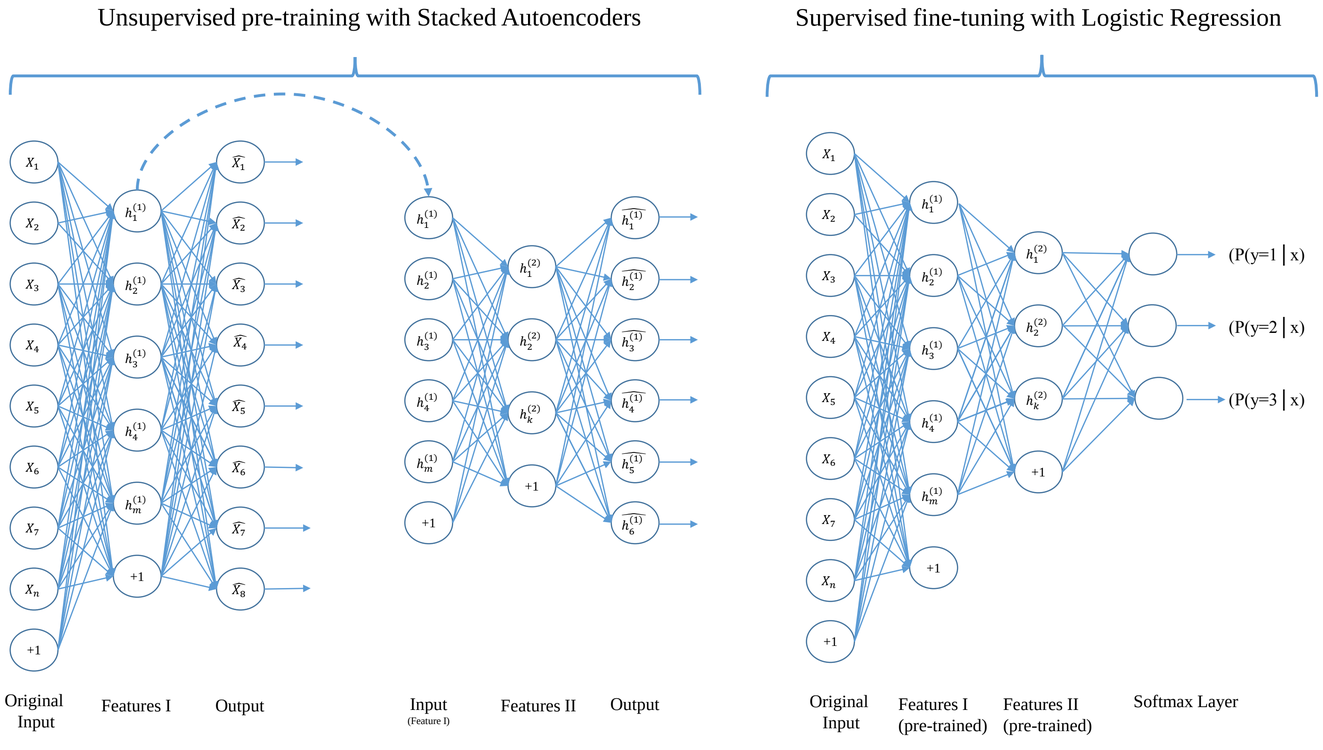
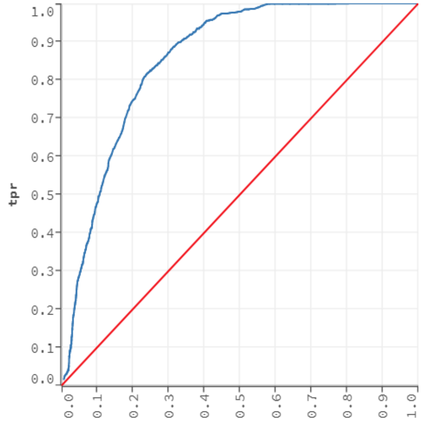


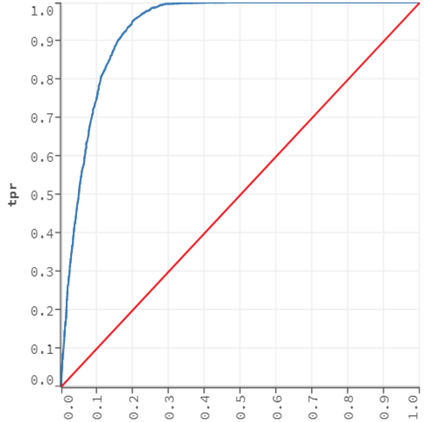
Figure 2 Stacked autoencoders based deep learning. Unsupervised pre-training on the left, supervised fine-tuning on the right.

Figure 3



a. Results for imbalanced dataset

AUC = 0.855



b. Results for balanced dataset

AUC = 0.932

Figure 3 ROC Curves for application to (a) imbalanced and (b) balanced datasets. ROC curves plot the true positive rate (tpr) against the false positive rate (fpr).