VaryMinions: Leveraging RNNs to Identify Variants in Event Logs

Sophie Fortz
sophie.fortz@unamur.be
PReCISE, NaDI, Faculty of Computer Science, University of Namur
Namur, Belgium

Paul Temple
paul.temple@unamur.be
PReCISE, NaDI, Faculty of Computer Science, University of Namur
Namur, Belgium

Xavier Devroey
x.d.m.devroey@tudelft.nl
Delft University of Technology
Delft, Netherlands

Patrick Heymans
patrick.heymans@unamur.be
PReCISE, NaDI, Faculty of Computer Science, University of Namur
Namur, Belgium

Gilles Perrouin
gilles.perrouin@unamur.be
PReCISE, NaDI, Faculty of Computer Science, University of Namur
Namur, Belgium

ABSTRACT

Business processes have to manage *variability* in their execution, *e.g.,* to deliver the correct building permit in different municipalities. This variability is visible in event logs, where sequences of events are shared by the core process (building permit authorisation) but may also be specific to each municipality. To rationalise resources (*e.g.,* derive a configurable business process capturing all municipalities' permit variants) or to debug anomalous behaviour, it is mandatory to identify to which variant a given trace belongs. This paper supports this task by training Long Short Term Memory (LSTMs) and Gated Recurrent Units (GRUs) algorithms on two datasets: a configurable municipality and a travel expenses workflow. We demonstrate that variability can be identified accurately (> 87%) and discuss the challenges of learning highly entangled variants.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Software and its engineering** → **Software reverse engineering; Software product lines**.

KEYWORDS

Configurable processes, Recurrent Neural Networks, Variability Mining

ACM Reference Format:
Sophie Fortz, Paul Temple, Xavier Devroey, Patrick Heymans, and Gilles Perrouin. 2021. VaryMinions: Leveraging RNNs to Identify Variants in Event Logs. In *Proceedings of the 5th International Workshop on Machine Learning Techniques for Software Quality Evolution (MaLTESQuE '21), August 23, 2021, Athens, Greece.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3472674.3473980

MaLTESQuE '21, August 23, 2021, Athens, Greece

1 INTRODUCTION

Business processes capture the activities of every profit or non-profit, public or private organisation, coordinating humans and software to collectively deliver value. As organisations evolve, new needs appear: *e.g.,* covering electric scooters for an insurance company or handling a change in the law about reimbursing travel expenses at the university. These needs lead to the emergence of *process variants*, differing in their control flow or performance while having commonalities with the original processes. We consider process executions stored in event logs, where an *event trace* (or trace) is an ordered sequence of events. To debug an anomalous process execution or to explore process refactoring opportunities, it is necessary to identify which variant(s) may have produced a given trace. Existing *variant analysis* [42] techniques do not answer this question but rather cover the inverse operation: *i.e.,* focusing on the differences between identified variants. In this paper, we train Recurrent Neural Networks (RNNs) [37] with different hyperparameters (loss and activation functions among others) to predict the candidate variant(s) that could produce a given event trace.

We provide the following contributions: (i) a first experiment of the usage of Long Short Term Memory (LSTMs) [22] and Gated Recurrent Units (GRUs) [8], two RNN architectures, on two datasets (municipality management and travel expenses) showing that we can identify the variant(s) producing an event trace with a high accuracy (> 87%) and that there is no clear dominance of one network architecture; (ii) a characterisation of the learning difficulty based on behaviour shared amongst event traces; (iii) an implementation of our approach exploiting the Tensorflow [12] and Keras [9] frameworks, available in the replication package with the full results [16].

Section 2 introduces process mining, RNNs, and related work. Section 3 presents the datasets, the experimental setup and results. Section 4 discusses certain factors influencing our experiments such as hyper-parameter variability and alternate labelling of variants. Section 5 wraps up the paper.

2 BACKGROUND AND RELATED WORK

Process Variants. Nowadays, many organisations work with multiple (business) processes in parallel that can highly depend on environmental and human factors. For instance, a business process

can be influenced by regional laws, available resources, the size of the organisation, *etc.* Most of them share common behaviour meaning that for one general business process, one can define several process variants. These similar processes can be grouped in process lines or process families and then modelled using different formalisms [36]. Analysing the specificities and commonalities of these process variants allow scale economies and help practitioners to improve the general business process, define new variants or maintain the existing ones [42]. For maintenance purposes, behaviour can be examined to identify related problems. This behaviour may concern only a fraction of all the variants and knowing them is crucial. However, event logs do not usually *contain information about the specific variant (or set of variants) which (could have) produced the event traces.* This can prevent practitioners to reproduce the *exact context of the problem that has to be fixed.* This forms the problem statement we are tackling in this paper.

LSTMs and GRUs. Recurrent neural networks (RNN) can predict the next event in a trace or the final execution state [14, 41]. The problem is when data sequences are long, traditional RNNs may face the so-called *vanishing or exploding gradient* problem [21]: weights from the first layers may rarely be adjusted since, during learning, the back-propagation mechanism re-injects prediction errors backwardly in the network starting from its output layer so that it can ultimately provide the right outcome. Conversely, the gradient can grow exponentially, yielding intractable computations. Two RNN architectures are widely used when dealing with long sequences and long-term dependencies: Long-Short Term Memory (LSTM) [22] and Gated Recurrent Unit (GRU) [8]. LSTMs and GRUs were designed to alleviate these gradient issues [10] by using gates to keep specific long-term information in memory. They proved their efficiency to deal with text classification [25, 29] for instance. We consider traces as text, *i.e.,* an ordered sequence of symbols that follows a given grammar. This motivates the use of LSTMs and GRUs for our purpose. Furthermore, these models have shown encouraging results for activity prediction and classification over process logs [20, 41].

Machine Learning for Process Monitoring and Mining. Machine learning has been notably used in business process monitoring. For example, deep learning [33, 34] and auto-encoders [32] are used to detect anomalies inside a process. In our case, we focus on the classification task. Bobek *et al.* [6] offer recommendations to configure variability-aware business processes at design time with Bayesian Networks. Clustering techniques have also been used [11, 30, 46] to perform classification tasks in an unsupervised way, *i.e.,* without knowing the classes to learn. Song *et al.* use dimensionality reduction techniques to improve trace clustering [39]. In our context, we want to specify the variants (*i.e.,* the classes) to learn. Finally, Hinkka *et al.* [20] aim at categorising traces into classes, thanks to LSTMs and GRUs. However, their approach differ on several points: (i) they define artificial classes, and (ii) they focus on binary classification.

Engineering Configurable Processes. When trying to (retro)-engineer configurable processes or even perform maintenance and/or evolution, some of the reported techniques rely on grammar-based or evolutionary algorithms, while others are machine learning (ML) oriented. The latter mostly consider tasks like clustering traces (*e.g.,* [39]) or predicting the next events based on past observations (*e.g.,* [41]). However, few techniques allow to retrieve a complete configurable process from event logs. Some approaches use genetic algorithms [7, 26], but they are limited to a small number of variants. Another option is to use (configurable) process fragments to re-build the configurable model [4]. Sikal *et al.* propose a pattern for variability discovery during process mining, but this approach is only methodological at this stage [38].

Machine Learning for Variability-intensive Systems. While there is a growing interest to employ ML techniques for VIS engineering [35], to the best of our knowledge, classification of variants from behavioural traces using ML techniques has not been studied. ML approaches have been used to support performance prediction and optimisation, *e.g.,* [1, 5, 24] or to improve the search for good and acceptable configurations *e.g.,* [31, 43]. If some of these works also target classification tasks, they consider configurations as the main entry point of their approaches and do not take the behaviour of the studied systems into account. ML also support defect prediction [2, 40]. In particular, Strüder *et al.* demonstrated that artificial neural networks were suitable for that task [40]. While ML can support VIS engineering, the converse, *i.e.,* applying variability-aware techniques to neural networks is also true. For example, Ghamizi *et al.* developed a framework to explore variability amongst different neural networks architectures and automated search-based techniques to find the optimal one for a given task [17, 18]. We built our work on the analogy between behavioural execution traces and text to choose RNNs as a relevant way to deal with temporal sequences. Natural Language Processing (NLP) is also increasingly used to extract variants from text specifications. Li *et al.* conducted a systematic literature review of this topic [27]. None of the reviewed works used RNNs but other classification models (decision trees, association rules, *etc.*). Recently, Arganese *et al.* investigated ambiguity in natural requirements as variability points [3] but here also, reasoning is more at the syntactical level (words) rather than on the analysis of complete sequences.

3 EVALUATION

Our evaluation addresses the following research questions: **RQ1** *How accurately can we identify process variants based on their traces?*, studying the applicability of our approach; and **RQ2** *What is the performance of LSTMs versus that of GRUs for process traces classification?*, identifying the most suited family of classifiers.

3.1 Datasets

We use two datasets: the 2015 and 2020 editions of the Business Process Intelligence Challenge (BPIC). Each one contains event logs, describing different executions of configurable processes: *BPIC15* (DS1) [45] represents building permit applications in five municipalities, each one corresponding to a process variant; and *BPIC20* (DS2) [44] gathers data from the travel reimbursement process at the Eindhoven University of Technology (TU/e), where variants correspond to different kind of documents to be managed.

Preprocessing. The original datasets contain only valid and complete traces, together with other information. We prune the logs to keep only the process variant id, the trace id and the sequence of events. To cope with different trace lengths, we apply padding

Table 1: Overview of the preprocessed datasets used in our experiments. Class-specific metrics (cols 3–5) represent: (i) the number of traces per class, (ii) the percentage of traces assigned specifically to this variant in the dataset, and (iii) the percentage of traces shared by this variant and at least an other one.

Dataset	Class Id	# Traces	% Variant-specific behaviour	% Shared behaviour
DS1 (BPIC15) 5,542 traces	**Munic. 1**	1170	99.658	0.342
	Munic. 2	828	99.638	0.362
	Munic. 3	1350	99.778	0.222
	Munic. 4	1049	99.905	0.095
	Munic. 5	1153	99.827	0.173
DS2 (BPIC20) 2,074 traces	**Int'l Decl.**	753	30.013	69.987
	Dom. Decl.	99	100	0.0
	Permit Req.	1478	64.344	35.656
	Prepaid	202	90.099	9.901
	Req. For Pay.	89	77.528	22.472

(*i.e.,* filling traces with other meaningless events and using a mask to know where the processing should stop). Trace duplicates are removed and since multiple variants can produce the same trace, we encode the variants into a binary vector (where the size matches the number of variants) that serve as a label. A value of one at the *i*-th index of the vector denotes that we observed at least once the trace associated to variant *i*. Traces that are associated to all variants have thus a vector full of ones. In the end, each trace is associated with one or more variants (*i.e.,* classes) and we expect the RNN models to learn these associations to predict the variant(s) for an unlabelled trace.

Description As described in Table 1, DS1 contains 5, 542 traces after preprocesssing. The five process variants are fairly equally represented since they contain 1, 108 traces on average, with a minimum of 828 and a maximum of 1, 350. DS1 is therefore well balanced. DS2 contains 2, 074 traces after preprocessing, with 5 process variants. The least and most represented process variants contain respectively 89 and 1, 478 traces, with an average of 415 traces per variant. Therefore, the dataset is imbalanced, suggesting it is more difficult to learn accurately.

To better characterise the learning complexity, Table 1 shows the number of traces per class and the overlap (*i.e.,* percentage of variant-specific and shared behaviour) between classes. The number of traces provides a first indication of the learning difficulty: more traces generally yield a more accurate network once trained. DS1 contains equally represented classes with limited overlaps ($< 0.5\%$ in the last column), while DS2 is less balanced in how classes are represented and how they are interleaved, denoting a shared behaviour between multiple variants. In particular, for DS2, there is a big overlap between the *International Declaration* and the *Permit Request* variants, and between the *Prepaid Travel Cost* and the *Request For Payment* variants, while the *Domestic Declaration* variant is completely separated.

3.2 RNN Configurations

Hyper-parameters values are described hereafter and in Table 2.

Classifiers. We chose to use GRU and LSTM as the two RNN models to learn how to map traces to their corresponding process variants. Since the traces are short compared to text documents, we decided to use networks with only one hidden layer, avoiding potential overfitting which may emerge from more complex structures (*e.g.,* auto-encoder) while offering satisfactory prediction abilities.

The network begins with an Embedding layer whose purpose is to transform individual traces' events into tensors which are the primitive objects manipulated by RNNs. The output of the Embedding layer is a set of tensors with weights that represent numerically (and not alphabetically) events in traces. We consider that all events are meaningful, as they are part of actual process executions. We thus decided to keep them all in our representation. Then, the Embedding layer is linked to the recurrent layer (*i.e.,*, our hidden layer) in turn connected to the output layer of the RNN (*i.e.,* a Dense layer with as many neurons as classes to predict). Regarding the hidden layer, we used bidirectional recurrent networks [37] exploiting past and future information by reading forward and backward traces in two independent passes. Such units have been demonstrated to perform well for natural language processing applications, such as attributing phonemes to spoken sequences [37]. In our case, traces are fully available at training time and reading forward and backward can help in grasping long term relations between events.

Units. We vary the number of units in the hidden layer. We have considered the following numbers of units: 5, 10 and 30. These numbers are commensurate with the maximum trace length.

Training set, batch size and epochs. We have additionally set the following parameters: (i) the percentage of the datasets used for training is set to 66% which is a common value in the ML community, the remaining traces are used in the test set to assess generalisation performances of the trained models; (ii) we set the batch size to 128, which is adapted to the dataset size; (iii) We set the number of epochs to 20 to avoid overfitting. In our preliminary evaluations (between 10 and 50 epochs), a plateau was reached after approximately 15 epochs. We finally set the number of epochs to 20, to allow for small increases in accuracy.

Activation functions. We experimented with different activation functions. These are used at the level of units to provide an output value for each neuron. For the hidden layer, we have used a Rectified Linear Unit (ReLU) function that alleviates the vanishing gradient problem. On the output layer, we have experienced with sigmoid and hyperbolic tangent (tanh) activation functions.

Loss functions. Loss functions are used during training to optimise the weights of the networks by back-propagating errors. We have used three loss functions already available, namely Binary Cross-Entropy (with and without logits) (respectively Bin-CE and Bin-CE logits) and the Mean Squared Error (MSE). Logit is defined as the inverse function of the sigmoid. These functions are used in a variety of contexts[1]. We also used two others that we implemented ourselves: a variant of Jaccard distance [23] (Weight_Jaccard), and the Manhattan distance between two vectors. Because we might have to assign a single trace to different process variants, we want to compute element-wise differences between two vectors to ultimately define a distance score. The Manhattan distance (sometimes called L1 norm) computes the sum of absolute differences between

[1]Categorical Cross-Entropy prevents from associating multiple process variants to one trace as it is usually used in conjunction with the softmax activation function.

Table 2: Hyper-parameters settings

Hyper-parameter	Considered values
Type of Classifier	GRU, LSTM
# Units	5, 10, 30
% Training Set	66%
Batch Size	128
# Epochs	20
Activation Function	sigmoid, tanh
Loss Function	Bin-CE, Bin-CE logits, MSE, Weight_Jaccard, Manhattan

each element of the two vectors (*i.e.,* in this case the process variants). The Jaccard distance assesses how equal elements of two vectors are over their size. We have implemented a variant of the Jaccard distance to cope with floating-point values generated by the network. The Jaccard distance has been employed to evaluate trace dissimilarity in variability-intensive systems (*e.g.,* [13]). Further discussions about the use and characteristics of these loss functions are provided in Section 4.1.

3.3 Evaluation Setup

We evaluated 2 $models \times 3$ $\#units \times 1$ $\%training\ set \times 1$ $batch\ size \times 1$ $\#\ epochs \times 2$ $activation\ funtions \times 5$ $loss\ functions = 60$ different parameterisations of RNN. The time needed for a single execution varies between 20 seconds and 2 minutes depending on the dataset. We did not address class imbalance issues since even minority classes were successfully classified by our models. We conducted our experiments on three different laptops with similar hardware (core i7 CPUs, 16GB of RAM). In total, we ran our 60 different network configurations with 10 repetitions (to mitigate random aspects) on the two different datasets resulting in $60 \times 10 \times 2 = 1,200$ runs and more than 22 hours of execution. All our scripts are written in Python 3, with the Keras and Tensorflow frameworks for deep learning. Our replication package is openly available [16].

3.4 Evaluation Results

Due to space constraints, Tables 3a and 3b only report the classification performances of the 5 best and 5 worst configurations for both datasets. The full results are available in our replication package. We observe that for DS1 and DS2 at least one of the networks can achieve an accuracy of 88% (for DS1) and 87% (for DS2) with a small standard deviation. Among the top 5 parameterisations, two out of five use LSTM and three use GRU. The best results are achieved with 30 units. Three pairs of loss and activation functions seem to stand out: MSE with tanh, MSE with sigmoid and binary cross-entropy (without logits values) with sigmoid.

Answer to RQ1 (accuracy): we were able to train RNNs providing an accuracy above 87% for DS1 and DS2. The following pairs of loss and activation functions stand out: MSE with tanh, MSE with sigmoid and binary cross-entropy combined with the sigmoid.

Regarding the five worst-performing parameterisations, we find similar configurations for both datasets. For example, five configurations (two for DS1 and three for DS2) show that the combination of the sigmoid activation function and the binary cross-entropy with logits is not a good fit. This is consistent with Keras documentation,[2] logits should not be used when dealing with probability distributions (*e.g.,*, sigmoid function). The Manhattan distance does not provide good results with sigmoid (in two configurations for DS1 and one for DS2). The number of units play a role in the performance of the RNN. When we analyse all the results, a tanh activation function combined with a low number of units generally leads to lower accuracy. Similarly to the best configurations, using GRU or LSTM does not seem to influence the results.

Table 3: Averaged accuracy and standard deviations over 10 runs. Each line corresponds to a parameterisation of a RNN.

(a) Results for dataset BPIC15.

Model	Loss	Activ.	Units	Mean	Sd
LSTM	MSE	tanh	30	0.88249	0.00662
GRU	Bin-CE	sigmoid	30	0.87788	0.01041
GRU	MSE	tanh	30	0.87066	0.01096
GRU	MSE	sigmoid	30	0.86642	0.01323
LSTM	MSE	sigmoid	30	0.86265	0.01275
		...			
LSTM	Manhattan	sigmoid	30	0.21862	0.0169
LSTM	Bin-CE logits	sigmoid	30	0.21804	0.01106
GRU	Bin-CE logits	sigmoid	30	0.21565	0.01526
GRU	Manhattan	sigmoid	30	0.21464	0.01558
LSTM	Manhattan	tanh	5	0.20308	0.03185

(b) Results for dataset BPIC20

Model	Loss	Activ.	Units	Mean	Sd
GRU	Bin-CE	sigmoid	30	0.87323	0.05122
GRU	MSE	sigmoid	30	0.83938	0.08053
LSTM	MSE	tanh	30	0.8296	0.05432
GRU	MSE	tanh	30	0.81034	0.0615
LSTM	MSE	sigmoid	30	0.7983	0.09539
		...			
GRU	Bin-CE logits	sigmoid	30	0.46501	0.00728
LSTM	Bin-CE logits	sigmoid	10	0.46133	0.01754
LSTM	Weight_Jaccard	tanh	10	0.43669	0.03958
LSTM	Manhattan	sigmoid	30	0.2772	0.19316
LSTM	Bin-CE logits	sigmoid	30	0.27224	0.18167

Answer to RQ2 (classifiers): In the top combinations of both DS1 and DS2, performance of the LSTM and GRU varies significantly (*e.g.*, from 79% to 88% for GRU) and are mixed, with no absolute winner. Therefore, we cannot conclude on the prevalence of GRUs over LSTMs for these two datasets.

4 DISCUSSION AND FUTURE WORK

4.1 Hyper-parameter Variability

The use of RNNs in this context requires to carefully dimension the network and consider many configurations that can influence the classification performances. We present them below.

Loss functions. We use the mean squared error which is used when tackling a regression problem. In this case, the output of the network is not a class label anymore but a real value and the goal of the regression is to minimise the difference between the expected values and the predicted ones. Using such kind of formulation can be surprising as we initially tackle a classification problem (*i.e.,* does a trace belong or not to one or more variants?) but the output

[2] https://keras.io/api/losses/probabilistic_losses/#binarycrossentropy-function

of our network are probabilities of belonging to variants. In this sense, we can argue that the tackled problem is slightly turned into a regression problem. This would explain the surprisingly good results that we obtained with this loss function.

The choice of the loss function is therefore extremely difficult since multiple aspects must be taken into account: the formalisation of the problem (*e.g.,* single or multi-label, regression or classification) or the way to compute errors. Even when trying to choose carefully the loss function according to these points, our results indicate that the MSE does surprisingly well while the initial setting suggests that it is not an appropriate measure. Given the importance of a loss function on the observed performance, experimenting with further loss functions appears promising. For example, the focal loss [28], which penalises more misclassified instances than well-classified ones, is a perspective that we aim to follow.

Complexity of the neural networks We argued that learning trace-to-variant mapping was feasible due to the number of traces w.r.t. the limited number of process variants. Generally, the challenge lies in the fact that having temporal sequences forces dependencies between elements that are usually learned separately. Therefore, we observed that network architectures with more units provided better performance. Besides, we suppose that deeper RNNs (*i.e.,* increasing the number of hidden layers) may have a positive impact. Adding more layers increases the complexity of the model (as well as required resources needed for training), but allows for a more accurate mapping between traces and variants. Yet, the risk of overfitting must not be neglected. In the future, we will also consider architectures such as auto-encoders to produce a compact intern representation of traces, that could be more efficient in discriminating them according to the process variants. Similarly to other application domains (*e.g.,* image or speech processing), learning more compact representations could rely on new feature descriptors instead of only considering events of a trace.

4.2 Variant-based vs. Option-based Labelling

Our results indicate that applying classification techniques on a variant-based approach (*i.e.,* identify the variants producing a specific trace) using Natural Language Processing (NLP) [3, 27] is promising. However, it has a major drawback: a variant-based approach requires enumerating all the variants to produce the training dataset. If in our evaluation, the number of variants was limited, the well-known combinatorial explosion problem may prevent us to apply these techniques to larger configurable processes like, for instance, continuous integration workflows with hundreds of options, leading to an intractable number of possible variants.

One future possibility to address this limitation is to work on data representation. Indeed, a variant is formed by a combination of (Boolean) options, corresponding to a *configuration* of the system. If variants cannot be enumerated, these options can. In this case, we need a new representation which can depict the three states of each option: activated, deactivated or undetermined (*i.e.,* the presence of the option is not relevant for the current context). The neural network will learn a partial configuration allowing for a more fine-grained mapping. This would be useful to locate with more precision a combination of options yielding a given anomalous event trace. Such learned models may be used in fault localisation and repair techniques [15].

4.3 Threats to Validity

Internal validity. We selected LTSMs and GRUs both for their ability to deal with temporal sequences and to evade the vanishing gradient issue. Due to the lack of computing resources, we could not apply automated tuning techniques; however, we evaluated 60 combinations of RNNs for both datasets. Since it is impossible to provide exhaustive coverage of the hyperparameter space, we may have missed some relevant parameterisations.

External Validity. Our initial results may not generalise to all configurable systems as we only considered process variants. We chose datasets having very different characteristics and based our perspectives on more general variability-aware consideration which we think provide useful insights outside the process modelling community. In the future, we would like to extend our work to different kinds of execution logs, issued from variability-intensive open-source software such as JHipster [19].

5 CONCLUSION

In this paper, we used LSTMs and GRUs to map event traces to their corresponding process variants, a prerequisite to perform processing debugging and refactoring activities. Our experiments [16] demonstrated that it is possible to learn mappings with an accuracy greater than 87% on two typical business process logs. Our future plans include the design of dedicated loss functions, the exploration of different neural architectures, and the adaptation of the approach to other application domains, such as variability-intensive systems.

ACKNOWLEDGEMENTS

Sophie Fortz is supported by the FNRS via a FRIA grant. Paul Temple is supported by the EOS VeriLearn project (FNRS Grant O05518F-RG03). Gilles Perrouin is an FNRS Research Associate. Xavier Devroey is supported by the Vici "TestShift" project (No. VI.C.182.032) from the Dutch Science Foundation NWO.

REFERENCES

[1] Juliana Alves Pereira, Mathieu Acher, Hugo Martin, and Jean-Marc Jézéquel. 2020. Sampling Effect on Performance Prediction of Configurable Systems: A Case Study. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering*. ACM, 277–288. https://doi.org/10.1145/3358960.3379137

[2] Benoit Amand, Maxime Cordy, Patrick Heymans, Mathieu Acher, Paul Temple, and Jean-Marc Jézéquel. 2019. Towards learning-aided configuration in 3D printing: Feasibility study and application to defect prediction. In *Proceedings of the 13th International Workshop on Variability Modelling of Software-Intensive Systems*. ACM, 1–9. https://doi.org/10.1145/3302333.3302338

[3] Eleonora Arganese, Alessandro Fantechi, Stefania Gnesi, and Laura Semini. 2020. Nuts and Bolts of Extracting Variability Models from Natural Language Requirements Documents. In *Integrating Research and Practice in Software Engineering*. Springer, 125–143. https://doi.org/10.1007/978-3-030-26574-8_10

[4] Nour Assy, Nguyen Ngoc Chan, and Walid Gaaloul. 2015. An automated approach for assisting the design of configurable process models. *IEEE transactions on services computing* 8, 6 (2015), 874–888. https://doi.org/10.1109/TSC.2015.2477815

[5] Davide Bacciu, Stefania Gnesi, and Laura Semini. 2015. Using a Machine Learning Approach to Implement and Evaluate Product Line Features. In *Proceedings 11th International Workshop on Automated Specification and Verification of Web Systems, WWV 2015, Oslo, Norway, 23rd June 2015 (EPTCS)*, Maurice H. ter Beek and Alberto Lluch-Lafuente (Eds.), Vol. 188. EPTCS, 75–83. https://doi.org/10.4204/EPTCS.188.8

[6] Szymon Bobek, Mateusz Baran, Krzysztof Kluza, and Grzegorz J Nalepa. 2013. Application of Bayesian Networks to Recommendations in Business Process Modeling.. In *AIBP@ AI* IA*. Springer, 41–50.

[7] Joos CAM Buijs, Boudewijn F van Dongen, and Wil MP van der Aalst. 2013. Mining configurable process models from collections of event logs. In *Business process management*. Springer, 33–48. https://doi.org/10.1007/978-3-642-40176-3_5

[8] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 1724–1734. https://doi.org/10.3115/v1/D14-1179

[9] François Chollet et al. 2015. Keras. https://keras.io.

[10] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

[11] Jochen De Weerdt, Seppe KLM vanden Broucke, Jan Vanthienen, and Bart Baesens. 2012. Leveraging process discovery with trace clustering and text mining for intelligent analysis of incident management processes. In *IEEE Congress on Evolutionary Computation*. IEEE, 1–8. https://doi.org/10.1109/CEC.2012.6256459

[12] TensorFlow Developers. 2021. *TensorFlow*. https://doi.org/10.5281/zenodo.4758419

[13] Xavier Devroey, Gilles Perrouin, Axel Legay, Pierre-Yves Schobbens, and Patrick Heymans. 2016. Search-based similarity-driven behavioural SPL testing. In *Proceedings of the Tenth International Workshop on Variability Modelling of Software-intensive Systems*. 89–96.

[14] Joerg Evermann, Jana-Rebecca Rehse, and Peter Fettke. 2017. Predicting process behaviour using deep learning. *Decision Support Systems* 100 (2017), 129–140. https://doi.org/10.1016/j.dss.2017.04.003

[15] Dirk Fahland and Wil M.P. van der Aalst. 2015. Model repair — aligning process models to reality. *Information Systems* 47 (2015), 220–243. https://doi.org/10.1016/j.is.2013.12.007

[16] Sophie Fortz, Paul Temple, Xavier Devroey, Patrick Heymans, and Gilles Perrouin. 2021. VaryMinions. https://doi.org/10.5281/zenodo.5083334

[17] Salah Ghamizi, Maxime Cordy, Mike Papadakis, and Yves Le Traon. 2019. Automated search for configurations of convolutional neural network architectures. In *Proceedings of the 23rd International Systems and Software Product Line Conference-Volume A*. ACM, 119–130. https://doi.org/10.1145/3336294.3336306

[18] Salah Ghamizi, Maxime Cordy, Mike Papadakis, and Yves Le Traon. 2020. FeatureNET: diversity-driven generation of deep learning models. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*. ACM, 41–44. https://doi.org/10.1145/3377812.3382153

[19] Axel Halin, Alexandre Nuttinck, Mathieu Acher, Xavier Devroey, Gilles Perrouin, and Benoit Baudry. 2019. Test them all, is it worth it? Assessing configuration sampling on the JHipster Web development stack. *Empirical Software Engineering* 24, 2 (2019), 674–717. https://doi.org/10.1007/s10664-018-9635-4

[20] Markku Hinkka, Teemu Lehto, Keijo Heljanko, and Alexander Jung. 2018. Classifying process instances using recurrent neural networks. In *International Conference on Business Process Management*. Springer, 313–324. https://doi.org/10.1007/978-3-030-11641-5_25

[21] Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6, 02 (1998), 107–116. https://doi.org/10.1142/S0218488598000094

[22] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[23] Paul Jaccard. 1901. Etude comparative de la distribution florale dans une portion des Alpes et des JurA. *Bulletin de la Société Vaudoise des Sciences Naturelles* 37 (1901).

[24] Christian Kaltenecker, Alexander Grebhahn, Norbert Siegmund, and Sven Apel. 2020. The interplay of sampling and machine learning for software performance prediction. *IEEE Software* 37, 4 (2020), 58–66. https://doi.org/10.1109/MS.2020.2987024

[25] Kamran Kowsari, Donald E Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S Gerber, and Laura E Barnes. 2017. Hdltex: Hierarchical deep learning for text classification. In *16th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 364–371. https://doi.org/10.1109/ICMLA.2017.0-134

[26] Marcello La Rosa and Marlon Dumas. 2008. Configurable process models: how to adopt standard practices in your how way? *BPTrends Newsletter* (2008).

[27] Yang Li, Sandro Schulze, and Gunter Saake. 2017. Reverse engineering variability from natural language documents: A systematic literature review. In *Proceedings of the 21st International Systems and Software Product Line Conference-Volume A*. ACM, 133–142. https://doi.org/10.1145/3106195.3106207

[28] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2020. Focal Loss for Dense Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 2 (2020), 318–327. https://doi.org/10.1109/TPAMI.2018.2858826

[29] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent Neural Network for Text Classification with Multi-Task Learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16)*. AAAI Press, 2873–2879. https://doi.org/10.5555/3060832.3061023

[30] Ronny S Mans, MH Schonenberg, Minseok Song, Wil MP van der Aalst, and Piet JM Bakker. 2008. Application of process mining in healthcare–a case study in a dutch hospital. In *International joint conference on biomedical engineering systems and technologies*. Springer, 425–438. https://doi.org/10.1007/978-3-540-92219-3_32

[31] Vivek Nair, Tim Menzies, Norbert Siegmund, and Sven Apel. 2017. Using bad learners to find good configurations. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, Paderborn, Germany, September 4-8, 2017*, Eric Bodden, Wilhelm Schäfer, Arie van Deursen, and Andrea Zisman (Eds.). ACM, 257–267. https://doi.org/10.1145/3106237.3106238

[32] Hoang Thi Cam Nguyen, Suhwan Lee, Jongchan Kim, Jonghyeon Ko, and Marco Comuzzi. 2019. Autoencoders for improving quality of process event logs. *Expert Systems with Applications* 131 (2019), 132–147. https://doi.org/10.1016/j.eswa.2019.04.052

[33] Timo Nolle, Alexander Seeliger, and Max Mühlhäuser. 2018. BINet: multivariate business process anomaly detection using deep learning. In *International Conference on Business Process Management*. Springer, 271–287. https://doi.org/10.1007/978-3-319-98648-7_16

[34] Timo Nolle, Alexander Seeliger, Nils Thoma, and Max Mühlhäuser. 2020. DeepAlign: Alignment-Based Process Anomaly Correction Using Recurrent Neural Networks. In *International Conference on Advanced Information Systems Engineering*. Springer, 319–333. https://doi.org/10.1007/978-3-030-49435-3_20

[35] Juliana Alves Pereira, Hugo Martin, Paul Temple, and Mathieu Acher. 2020. Machine Learning and Configurable Systems: A Gentle Introduction. In *Proceedings of the 24th ACM Conference on Systems and Software Product Line: Volume A (SPLC '20)*. ACM, Article 40, 1 pages. https://doi.org/10.1145/3382025.3414976

[36] Marcello La Rosa, Wil MP Van Der Aalst, Marlon Dumas, and Fredrik P Milani. 2017. Business process variability modeling: A survey. *Comput. Surveys* 50, 1 (2017), 1–45. https://doi.org/10.1145/3041957

[37] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing* 45, 11 (1997), 2673–2681. https://doi.org/10.1109/78.650093

[38] Rabab Sikal, Hanae Sbai, and Laila Kjiri. 2018. Configurable process mining: variability Discovery Approach. In *IEEE 5th International Congress on Information Science and Technology (CiSt)*. IEEE, 137–142. https://doi.org/10.1109/CIST.2018.8596526

[39] Minseok Song, H Yang, Seyed Hossein Siadat, and Mykola Pechenizkiy. 2013. A comparative study of dimensionality reduction techniques to enhance trace clustering performances. *Expert Systems with Applications* 40, 9 (2013), 3722 – 3737. https://doi.org/10.1016/j.eswa.2012.12.078

[40] Stefan Strüder, Mukelabai Mukelabai, Daniel Strüber, and Thorsten Berger. 2020. Feature-oriented defect prediction. In *Proceedings of the 24th ACM Conference on Systems and Software Product Line: Volume A*. ACM, 1–12. https://doi.org/10.1145/3382025.3414960

[41] Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. 2017. Predictive Business Process Monitoring with LSTM Neural Networks. In *Advanced Information Systems Engineering*. Springer, 477–492. https://doi.org/10.1007/978-3-319-59536-8_30

[42] Farbod Taymouri, Marcello La Rosa, Marlon Dumas, and Fabrizio Maria Maggi. 2021. Business process variant analysis: Survey and classification. *Knowledge-Based Systems* 211 (2021), 106557. https://doi.org/10.1016/j.knosys.2020.106557

[43] Paul Temple, Mathieu Acher, Gilles Perrouin, Battista Biggio, Jean-Marc Jézéquel, and Fabio Roli. 2019. Towards quality assurance of software product lines with adversarial configurations. In *Proceedings of the 23rd International Systems and Software Product Line Conference-Volume A*. ACM, 277–288. https://doi.org/10.1145/3336294.3336309

[44] Boudewijn van Dongen. 2020. BPI Challenge 2020. https://doi.org/10.4121/uuid:52fb97d4-4588-43c9-9d04-3604d4613b51

[45] B.F. (Boudewijn) van Dongen. 2015. BPI Challenge 2015. https://doi.org/10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1

[46] Ángel Jesús Varela-Vaca, José A Galindo, Belén Ramos-Gutiérrez, María Teresa Gómez-López, and David Benavides. 2019. Process mining to unleash variability management: discovering configuration workflows using logs. In *Proceedings of the 23rd International Systems and Software Product Line Conference-Volume A*. ACM, 265–276. https://doi.org/10.1145/3336294.3336303