# Nirdizati 2.0: New Features and Redesigned Backend

Williams Rizzi[1,2], Luca Simonetto[2], Chiara Di Francescomarino[2], Chiara Ghidini[2], Tõnis Kasekamp[3], and Fabrizio Maria Maggi[3]

[1] Free University of Bozen-Bolzano, Bozen, Italy
[2] Fondazione Bruno Kessler, Trento, Italy
{wrizzi, dfmchiara, ghidini}@fbk.eu
[3] University of Tartu, Tartu, Estonia
toniskasekamp@gmail.com, f.m.maggi@ut.ee

**Abstract.** Nirdizati is a dedicated tool for Predictive Process Monitoring, a field of Process Mining that aims at predicting how an ongoing execution of a business process will develop in the future using past process executions recorded in event logs. Nirdizati is a web application supporting users in building, comparing, and analyzing predictive models that can then be used to perform predictions on the future of an ongoing case. By providing a rich set of different state-of-the-art approaches, Nirdizati offers BPM researchers and practitioners a useful and flexible instrument for investigating and comparing Predictive Process Monitoring techniques. In this paper, we present a Nirdizati version with a redesigned backend, which improves its modularity and scalability, and with new features, which further enrich its capability to support researchers and practitioners to deal with different monitoring tasks.

**Keywords:** Predictive Process Monitoring · Process Mining · Machine Learning.

## 1 Introduction

Nirdizati [10] is an open-source, web-based Predictive Process Monitoring [13] tool supporting a wide range of state-of-the-art approaches and providing researchers and practitioners with a highly flexible instrument for the construction, comparison, analysis, and selection of predictive models. The previous version of Nirdizati supported different feature encodings and a variety of learning algorithms thus enabling the user to select the best predictive model providing accurate outcome-based (e.g., whether an ongoing trace will produce a certain outcome eventually in the future) and numerical (e.g., the remaining time of an ongoing trace) predictions related to a currently running process execution.

In this demo paper, we introduce a new version of Nirdizati that has been enhanced with: a completely redesigned backend to allow faster processing and better usage of resources; new algorithms to build the predictive models; and new encoding techniques to feed the algorithms with a richer set of

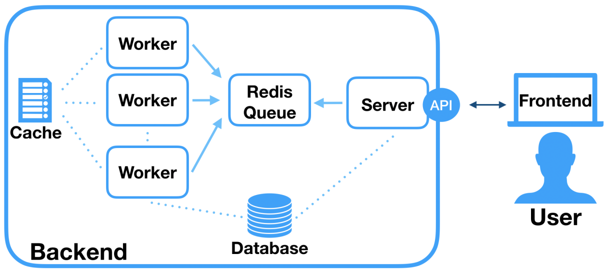

**Fig. 1.** `Nirdizati` architecture. The system follows a client-server architecture. The server exposes APIs to communicate with the client. When the server receives a new job, it puts it in a Redis Queue. When a new job is available in the Redis Queue, a worker takes care of the job. Workers, whose number can vary according to the needs, share access to the Redis Queue, to the cache, and to the database.

features. The backend redesign has introduced a new cache system in the pre-processing steps, and a database to store the status of the `Nirdizati` components and to track the evolution of the system as a whole. The new features include: (i) incremental learning algorithms, which are able to update the predictive models at runtime by leveraging new cases, as soon as they complete; (ii) time series prediction algorithms, which enable the prediction of the sequence of future activities; and (iii) intercase feature encoding, for taking into account not only the current ongoing case when making predictions on the current case, but also other concurrent ongoing cases. The current version of the tool is online at `http://research.nirdizati.org/`, while a tutorial and a video are available at `https://drive.google.com/drive/folders/1sG69DabBhmuPsWAOPA4yNLjFbEFVknYO`.

## 2 Redesigned Backend

By design, `Nirdizati`[4] supports the multiprocessing capabilities of nowadays processors, using a master-slave setting (see Fig. 1). In this setting, the master - the *server* in Fig. 1 - creates and orchestrates the jobs, and the slaves - the *workers* in Fig. 1 - take care of performing the jobs. When using such a setting for comparing different algorithms on the same task, most of the processing time is taken for the pre-processing phases that are repeated over and over for each different configuration. To overcome this time and resource consumption, `Nirdizati` has now been equipped with a *cache system* that avoids pre-processing the data multiple times when evaluating multiple approaches on a shared task and dataset.

When using caching in the `Nirdizati` architecture, however, two or more slaves performing the same pre-processing step concurrently can generate duplicate cache files. A *database system* able to track in a granular way the status of

[4] Source code available at `https://github.com/nirdizati-research/`

each object of the tool, i.e., the set of elements used in a given task, has been introduced. For instance, the object representing the task of splitting a log into a training and a validation set is composed of a foreign key to the object representing the log, the training and the validation set sizes, the type of ordering used for the split, the name of the split, and a unique key. Storing each object of the tool in a table of the database allows, on the one hand, an easy retrieval of the required information and, on the other hand, keeping in memory only the database keys instead of the objects themselves, resulting in a lighter footprint on the memory. From the database it is possible to retrieve the current jobs and their status, the available cached information, the previously trained models and the corresponding accuracy metrics. By relying on a stateful architecture and capitalizing the work carried out, `Nirdizati` is able to optimally exploit multiprocessing.

## 3 New Features

Two new families of algorithms have been added to the tool, namely, *incremental learning* and *time series prediction* algorithms. Furthermore, a new family of *intercase encodings* has been made available in `Nirdizati`.

While the previous version of `Nirdizati` mainly focused on predictive models for outcome-based and time-related predictions, this version of the tool also supports the creation and comparison of models predicting sequences of next activities. To this aim, the tool has been equipped with *time series prediction* based on two recurrent neural network algorithms: (i) Long Short Term Memory Networks (LSTM), and (ii) Gated Recurrent Unit (GRU), which have been proven to give competitive results in Predictive Process Monitoring [16].

`Nirdizati` has also been enriched with *incremental algorithms* for carrying out classification tasks. In particular, (i) Multinomial Naïve Bayes, (ii) Stochastic Gradient Descent (SGD) Classifier, (iii) Perceptron, and (iv) Neural Networks have been introduced. We decided to include Multinomial Naïve Bayes, since it is a fairly common algorithm that is rather simple, yet generally working pretty well on datasets with different characteristics [14]. SGD Classifier and Neural Networks showed to be rather effective techniques in different fields, while Perceptron is a particular case of SGD [19].

Finally, in order to define *intercase encodings*, the following intercase features can now be used: (i) executed events per day, (ii) resources used per day, and (iii) new cases per day. Eliciting and leveraging such features has shown to be rather effective in improving the accuracy of the trained predictive models [15].

Extensive improvements on the usability of the tool have been carried out:

- *Log feature visualization.* Once the user has uploaded an event log, the *log feature visualization* functionality allows accessing information regarding the log characteristics (e.g., the number of events and resources per day).
- *Dataset splitting, encoding and labeling.* These functionalities enable the user to *split* an event log into training and validation set by selecting the preferred case order and sizes of the sets; *label* the cases, choosing among a predefined

set of labeling criteria, and check the distribution of labels in the log; and select the preferred *encoding(s)* to be used.

- *Hyperparameter optimization.* Hyperparameter optimization approaches support users in tuning the hyperparameters to be used for training the predictive models [3,4]. Two possible hyperparameter optimization strategies are available in `Nirdizati`, i.e., (i) Tree of Parzen Estimators (TPE), and (ii) Random Search algorithm.
- *Result visualization.* The results can be filtered and visualized using various interactive plotting strategies. The results page contains four elements: (i) the configuration table, (ii) the results table, (iii) the prefix plot, and (iv) the bubble chart plots. The configuration table summarizes, for each predictive model, the configuration that originated the model, while the results table reports the accuracy of the model (in terms of well-known accuracy metrics, e.g., F-score and AUC) in tabular format. The results are also provided in the prefix plot, which shows the trend of each metric for different prefix lengths, and in the bubble chart plots, which enable the comparison of the different predictive models with respect to various perspectives (e.g., the comparison of different encodings or classification algorithms).

## 4 Concluding Remarks

This paper presented the latest advancements of `Nirdizati`. Specifically, its backend has been redesigned to provide a reliable and efficient framework that allows an easy comparison of different state-of-the-art Predictive Process Monitoring approaches. Moreover, new features have been added, so that `Nirdizati` is now able to run the most recent state-of-the-art techniques, such as incremental learning algorithms, time series prediction algorithms, and intercase feature encodings.

The recent stream of publications in the Predictive Process Monitoring field [2, 5, 12, 13, 15, 17] shows the need for tools able to support researchers and users in analyzing, comparing and selecting the techniques that are the most suitable for them. This need is also reflected in the growth of Predictive Process Monitoring plug-ins in well known process mining tools such as ProM [6] and Apromore [11, 18]. `Nirdizati` is a completely dedicated tool for running a very rich set of Predictive Process Monitoring techniques and its latest advancements make it even more robust, scalable and usable.

We assess the current Technology Readiness Level of `Nirdizati` to be 5. This release offers indeed a well-defined structure of the software and code documentation;[5] moreover, it is equipped with a very large test suite, and a *Continuous Integration* deployment pipeline. The tool has been extensively used and its features exercised on both simulated and real data, in the medical [1], administrative [7] and financial domain [8, 9]. We believe that all these reasons make `Nirdizati` a mature and useful instrument for the BPM community.

[5] `https://nirdizati-research.readthedocs.io/`

## References

1. 3TU Data Center: BPI Challenge 2011 Event Log (2011), http://dx.doi.org/10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54
2. Di Francescomarino, C., Dumas, M., Maggi, F.M., Teinemaa, I.: Clustering-based predictive process monitoring. IEEE Transactions on Services Computing pp. 1–1 (2018). https://doi.org/10.1109/TSC.2016.2645153
3. Di Francescomarino, C., Dumas, M., Federici, M., Ghidini, C., Maggi, F.M., Rizzi, W.: Predictive business process monitoring framework with hyperparameter optimization. In: CAiSE 2016. pp. 361–376 (2016)
4. Di Francescomarino, C., Dumas, M., Federici, M., Ghidini, C., Maggi, F.M., Rizzi, W., Simonetto, L.: Genetic algorithms for hyperparameter optimization in predictive business process monitoring. Inf. Syst. **74**(Part), 67–83 (2018)
5. Di Francescomarino, C., Ghidini, C., Maggi, F.M., Milani, F.: Predictive process monitoring methods: Which one suits me best? In: BPM 2018. pp. 462–479 (2018)
6. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM framework: A new era in process mining tool support. In: ICATPN 2005. pp. 444–454 (2005)
7. van Dongen B. F.: BPI Challenge 2012 (2012), http://dx.doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f
8. van Dongen B. F.: BPI Challenge 2015 (2015), http://dx.doi.org/10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1
9. van Dongen B. F.: BPI Challenge 2017 (2017), https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f
10. Jorbina, K., Rozumnyi, A., Verenich, I., Di Francescomarino, C., Dumas, M., Ghidini, C., Maggi, F.M., La Rosa, M., Raboczi, S.: Nirdizati: A web-based tool for predictive process monitoring. In: BPM Demo Track and BPM Dissertation Award, BPM 2017 (2017)
11. La Rosa, M., Reijers, H.A., van der Aalst, W.M.P., Dijkman, R.M., Mendling, J., Dumas, M., García-Bañuelos, L.: APROMORE: an advanced process model repository. Expert Syst. Appl. **38**(6), 7029–7040 (2011)
12. Leontjeva, A., Conforti, R., Di Francescomarino, C., Dumas, M., Maggi, F.M.: Complex symbolic sequence encodings for predictive monitoring of business processes. In: BPM 2015. pp. 297–313 (2015)
13. Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: CAiSE 2014. pp. 457–472 (2014)
14. Maisenbacher, M., Weidlich, M.: Handling concept drift in predictive process monitoring. In: IEEE SCC 2017. pp. 1–8 (2017)
15. Senderovich, A., Di Francescomarino, C., Ghidini, C., Jorbina, K., Maggi, F.M.: Intra and inter-case features in predictive process monitoring: A tale of two dimensions. In: BPM 2017, Proceedings. pp. 306–323 (2017)
16. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with LSTM neural networks. In: CAiSE 2017. pp. 477–492 (2017)
17. Teinemaa, I., Dumas, M., Maggi, F.M., Di Francescomarino, C.: Predictive business process monitoring with structured and unstructured data. In: BPM 2016. pp. 401–417 (2016)
18. Verenich, I., Mõskovski, S., Raboczi, S., Dumas, M., La Rosa, M., Maggi, F.M.: Predictive process monitoring in Apromore. In: Information Systems in the Big Data Era - CAiSE Forum 2018, Proceedings. pp. 244–253 (2018)
19. Zhang, T.: Solving large scale linear prediction problems using stochastic gradient descent algorithms. In: ICML (2004)