Predicting a Next Activity in Event Logs: an Approach based on LSTMs and Gradient Boosting

Maksim Karpov[1 [0000−0003−4503−2682]], Sergey Petrovich[1 [0000−0001−7664−0619]], Artem Streltsov[1 [0000−0001−7889−1630]], and Andrey Ustyuzhanin[1 [0000−0001−7865−2357]]

Faculty of Computer Science, HSE University, Moscow, Russia
{mekarpov, austyuzhanin}@hse.ru, {adstreltsov, sdpetrovich}@edu.hse.ru

Abstract. Predictive process monitoring (PPM) is considered one of the main domains in log analysis along with process mining. Still, it refers to observing future process behaviour, e.g., next activity prediction problem. PPM helps to detect errors in processes in time and prevent critical situations, significantly reducing company expenses and system failures. In all previous research, basic approaches to applying recurrent neural networks were mainly used. In this paper, we explore existing neural network approaches to sequence analysis and apply the best solutions for next activity prediction in an event log. However, neural networks may not always show the best results, so we compare deep learning solutions with gradient boosting models. In addition, we introduce an efficient way of vector representations and advanced methods of feature engineering for logs.

Keywords: Predictive process monitoring · Deep neural networks · Long short term memory · Gradient boosting · Feature engineering

1 Introduction

Nowadays, more and more attention in industry and research is paid to data-driven approaches in decision-making. One such field that helps to extract knowledge from data and apply predictive analytics in business process management systems is predictive process monitoring (PPM).

With the development of data science, companies are increasingly trying to replace "gut feeling" with data-driven results, and with the improvement of data analysis, automated methods are increasingly expanding the understanding of modern business. PPM is one of these tools for data-based analysis. Since nowadays there is a large availability of data logged by information systems supporting the day-to-day operations of companies, and mature and fast approaches are rapidly growing in PPM, these opportunities are becoming more and more practical.

Its instruments may help to predict process behaviour and to identify and address performance and compliance problems. Usually event logs (computer-generated records) are represented as a pair of an activity label and an activity timestamp, and it is crucial for developers or business-owners to understand which event is most likely to happen given the previous event sequence. This may help drastically to reduce costs and find bottlenecks in company processes.

That is why one of the most widespread domain in log analysis is PPM. The most significant tasks are:

- Next activity prediction. In this case the problem is quite similar to multi-class classification, but however the model should pay attention to logs particularities, such as time elapsed between events and their order.
- Next activity timestamp prediction. This task especially needs the application of time-aware models, e. g., LSTMs.
- Remaining cycle time and others. In this research we focus exclusively on next activity prediction task.

Referring to [1] and described more formally, an event log is a sequence of events with time delta between them. Every event log contains at least an activity name, a case id (a particular execution of a process instance) and a timestamp (i.e. time of event occurrence), that can be informative in case of log analysis, and an activity label. Moreover, some events entail other ones, so overall they form a structure where events happen in order, which is also an important feature.

Definition 1 (Event, Trace, Event Log). *An event is a tuple (c, a, ts) where c is the case id, a is the activity (event type) and ts is the timestamp. A trace is a non-empty sequence* $\sigma = \langle e_1, \ldots, e_{|\sigma|} \rangle$ *of events such that* $\forall\, i, j \in 1, \ldots, |\sigma|$ $e_i.c = e_j.c$ *and* $e_i.ts \leq e_j.ts$*, for* $1 \leq i < j \leq |\sigma|$*. An event log* $\mathcal{L}$ *is a set* $\sigma_1, \ldots, \sigma_{|\mathcal{L}|}$ *of traces. A trace can also be considered as a sequence of vectors, in which a vector contains all or a part of the information relating to an event, e.g. an event's activity. Formally,* $\sigma = \langle x_{(1)}, x_{(2)}, \ldots, x_{(t)} \rangle$*, where* $x_{(i)} \in R^{n \times 1}$ *is a vector, and the superscript indicates the time-order upon which the events happened, n is the number of features derived for each event.*

As previously mentioned, PPM as a domain of log analysis is aimed for some process predictions (e.g. next event prediction). In that way, the main goal is the most precise forecast of future events and its occurrence timestamps.

An event description is a string that contains at least a timestamp and a label. A very similar area of machine learning is natural language processing (NLP), where models learn to analyse human language, which is represented as a sequence of words or letters. The main difference of PPM from any other NLP task is the fact that an individual event can be treated as a separate word in a whole sentence (log), hence it needs a more thorough approach to events preprocessing and embedding.

Definition 2 (Prefix and label). *Given a trace* $\sigma = \langle e_1, \ldots, e_k, \ldots, e_{|\sigma|} \rangle$*, a prefix of length k, that is a non-empty sequence, is defined as* $f_p^{(k)}(\sigma) =$

$\langle e_1, \ldots, e_k \rangle$, with $0 < k < |\sigma_c|$ and a label (i.e. next activity) for a prefix of length k is defined as $f^{(k)}_{l,ts}(\sigma) = \langle e_{k+1} \rangle$. This can be generalized for an input trace representing a sequence of vectors. For example, the tuple of all possible prefixes and the tuple of all possible labels for $\sigma = \langle x_{(1)}, x_{(2)}, x_{(3)} \rangle$ are $\langle \langle x_{(1)} \rangle, \langle x_{(1)}, x_{(2)} \rangle \rangle$ and $\langle x_{(2)}, x_{(3)} \rangle$.

In this way, formally, the next activity prediction problem can be reformulated as predicting the event vector by the given non-empty prefix of the trace. Having an opportunity to predict the next activity, large companies in the industry are able to detect anomalies in their processes in real time. In addition, if companies can model processes in such a way, they can find the patterns and shortcomings of the current workflow organization and quickly eliminate or update them.

2 Related work

In the past, predictive business process monitoring approaches rarely took advantage of deep learning. However, some deep neural networks (DNN) architectures, such as RNNs and LSTMs, can be applied relatively straightforwardly since logs of software companies, banks and other institutions can be presented as a sequence of events. Nonetheless, in recent years several attempts were made to leverage these models in PPM.

In [10] there is proposed LSTM-based DNN architecture to predict the completion time of running process. The authors converted activity attributes using one-hot encoding and considered additional real-valued or categorical context attributes.

[14] mostly concentrated on explainability rather than on predictive quality. In their attempt to predict the next activity label they used Bi-LSTM and layer-wise relevance propagation (LRP) to assign a relevance value to each event to estimate its contribution to the answer.

DNN architecture by [13] is based on LSTM cells and is used for next activity and timestamp prediction. They proposed a multitask learning approach and like in LSTM networks for data-aware remaining time prediction of business process instance they used one-hot encoding for activity label. Nevertheless, the authors did not take into consideration any additional data attributes.

[6] introduced approaches for learning low dimensional representations of activities, traces, logs and models. They used methods similar to those proposed by [9].

[11] introduced another DNN architecture based on LSTM. They integrate the elapsed time between consecutive events to adjust the cell memory to predict the next activity label and its timestamp.

Models such as Generative Adversarial Nets (GAN) have been gaining popularity in recent years and have also found application in PPM. In works by [17] the authors went further and predict the rest of the sequence by the given trace prefix. Thus, they solve sequence-to-sequence task and use GAN with LSTM encoder-decoder architecture.

Despite the popularity of deep learning, some methods of classical machine learning can compete with neural networks in quality. [3] proposed a solution based on gradient boosting algorithm. In their work they used XGBoost for PPM tasks.

We propose a modified approach for next activity prediction, using embedding techniques, custom loss functions and different recurrent neural networks, along with applying another gradient boosting method.

3 Activity prediction framework

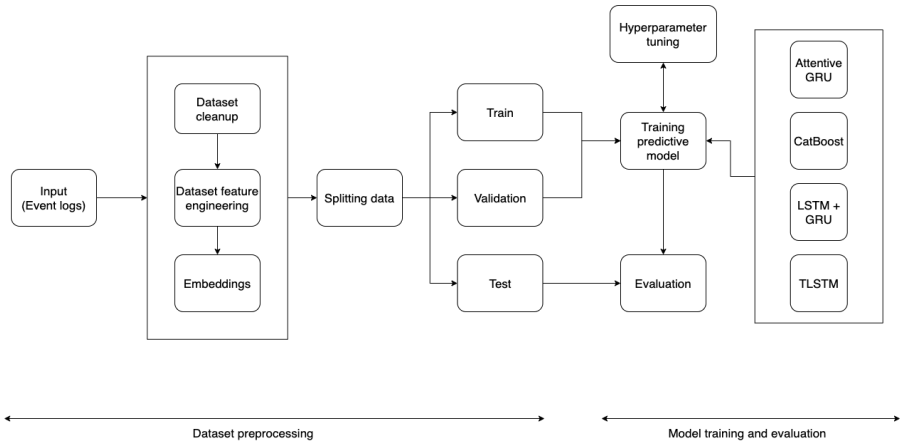

Fig. 1. Predictive framework.

3.1 Dataset preprocessing

It is crucial to apply some embeddings from NLP tasks to represent an event as a numeric vector, as only this data format is supported by predictive models:

- [9] proposed a Word2Vec embedding. Based on this work, [6] introduced an approach for embedding actions, traces, whole logs and models.
- Nevertheless, sometimes we need to encode not only an action, but a whole trace, similar to Doc2Vec, proposed by [7]. For that case [6] introduced a Trace2Vec architecture.
- However, it is worth experimenting with other embedding models, e.g. [12]. Based on co-occurrence probabilities, that approach might be also helpful while finding a proper action representation.

The problem of embedding an activity itself is obvious: training only the event identifier as a word, we lose information about time when the certain event happened.

Hence we need an approach to embed not only the raw activity, but also consider in context that the same event can occur at different times.

We propose an approach to binarize the timestamp into categories depending on time of the day (e.g "noon", "night", "evening") as concatenating to the event identifier and compare results with "raw" activity embedding. This way we enlarge the activity "corpus" and suppose it could make the activity embeddings more informative.

Further, it is reasonable to embed not only each activity separately, but also represent the whole trace or a sub-trace as a numeric vector, using, for example, Doc2Vec by [7]. However, embedding the whole trace may lead to target leakage. Therefore it becomes crucial to embed not the whole trace but its prefix: for each activity we represent the sub-trace from the start to the current event.

Data cleanup. Another important part of dataset preprocessing is its cleanup from outliers.

As the problem is not stated as anomaly detection, we consider excluding some activities like in [13] and [11]. We propose to drop such traces that contain "rare activities".

We consider an event rare if it occurs less than one percent across the entire log.

For Helpdesk, we drop some additional traces, that do not contain certain activities, that are supposed to be important and to be present in each non-outlying case.

Using timestamp. We propose to add some features based on the timestamp to make the model better comprehend the business process model.

- Time of day. In business process models, it is important to take into account what time of day an event occurs. This is important to understand, since many events occur during or outside the working day. We binarize timestamp into six possible variants of time of day:
  - Late Night
  - Early Morning
  - Morning
  - Noon
  - Evening
  - Night
- Timestamp split. We split timestamp into hour, day of week, month and year. And further, each of those features is encoded with One-Hot to make the model select weights for each entity value separately.
- Previous activity. For each event we keep the one that occurred in previous step.
- Timestamp deltas. The timestamp itself is not so informative, thus we consider using deltas between events instead.

1 With previous activity in the case.
2 With the first activity in the case. That helps to estimate the overall duration of the case.

– "Extended activity". We concatenate activity and the time of the day and encode such new activity with One-Hot.
– Ratio between the activity number in the case (after sorting the case log by timestamp) and the average case log length. According to [13] and [11], they use activity number in case as a feature. However, it is worth taking into account the average length of cases to estimate how long the trace continues relative to the other traces present in the event log.

Final datasets and their descriptions are presented in the Table 1.

Table 1. Embedding datasets description

Slug	Description
more_categorical	Used more categorical features (hour, day, month, etc...). Used as a base for other variants.
glove_raw	Using glove as embeddings for activity
glove_extended	Glove to embed activities + Doc2Vec to embed sub-traces
no_drop_cases	Do not drop cases with rare activities
raw_activity	Use embedding only on activity, do not append part of day
trace2vec	Word2Vec + Doc2Vec

3.2 Predictive model

One of the key aspects of the whole model is its architecture. As mentioned above, previously the problem of next activity prediction was solved by a statistical model or classical machine learning approaches. In this paper we try some neural network architectures.

Since the input data for model is presented in the form of sequences, it seems appropriate to use tools from related fields for sequence analysis. Nowadays the most popular area in sequence processing is NLP. Traditionally, NLP problems are solved using RNN models, especially "vanilla" LSTM and GRU cells. In this way, the main direction of this research is the implementation of different recurrent neural network architectures and the development of its various combinations that have not been used before in PPM tasks.

Focal loss. In this paper we chose to train our next activity prediction deep learning models using **Focal loss**, which was developed by [19]. Common methods to learn classification neural networks use standard cross-entropy loss (CE):

$$CE(p_t) = -\log p_t \tag{1}$$

where p_t is the predicted probability of the true label for t-th sample of training dataset. However, this loss function faces several problems when dealing with unbalanced datasets. This is due to the fact that it may be more profitable and easier for the model to increase the probability on already well-predicted major class samples than to try to better the learning probabilities for minor classes. This leads to an unsatisfactory quality of the model on the whole dataset. In tasks of PPM datasets are often unbalanced.

The idea of **Focal loss** is to add extra factor $(1 - p_t)^\gamma$, that is why this loss is defined as:

$$FL(p_t) = -(1 - p_t)^\gamma \log p_t \tag{2}$$

where $\gamma \geq 0$ is a scalar hyperparameter. The authors highlight two properties of the focal loss. First, when an example is misclassified and p_t is small, the modulating factor is near 1 and the loss is unaffected. As $p_t \to 1$, the factor goes to 0 and the loss for well-classified examples is down-weighted. Second, the focusing parameter γ smoothly adjusts the rate at which easy examples are down-weighted. When $\gamma = 0$, FL is equivalent to CE, and as γ is increased the effect of the modulating factor is likewise increased.

LSTM and GRU. The first model to solve the next activity prediction problem consists of RNN neural networks, specifically LSTM and GRU. The model consists of two parallel neural networks: GRU-network and LSTM-network. At each step they concatenate hidden states and pass the resulting vector through several fully connected layers. Final predictions are obtained using a softmax function. This architecture looks quite promising for the baseline, as it allows the LSTM network and the GRU network to complement each other. It is supposed to have a significant advantage over models that use only one type of network.

Exponential T-LSTM. Another approach is the T–LSTM model by [11]. As was mentioned before, the key idea of T–LSTM is that the importance of the previous event in next activity prediction decreases with the growth of the elapsed time. Thus, to take into account the influence of time, the authors add an additional multiplier in the LSTM cell, which is computed by some decay function. In the original paper they use the logarithmic function:

$$decay(x) = \frac{1}{\log(x + e)} \tag{3}$$

In our research we introduced **Pytorch** implementation of T-LSTM model with exponential decay function, which improved target test scores:

$$decay(x) = exp(-\alpha \cdot x) \tag{4}$$

It is also worth noting, that the proposed function has an additional scalar parameter α. Varying this parameter gives additional flexibility in optimization,

so this alpha parameter was also involved in the process of hyperparameter tuning.

Attentive GRU. Despite widespread classical solutions for sequence analysis, it seems promising to use more advanced and modern techniques. For instance, it may be effective to implement an attention mechanism. This can solve the main problem of recurrent neural networks - vanishing gradient problem. It should also be noted that LSTM and GRU networks partially cope with that problem, but instead they cannot take into account parts of the sequence which are quite far from each other because the information carried by the earlier parts of the sequence is forgotten. Traces in event logs are normally long, that is why LSTM or GRU networks may not work accurate in these cases.

The last model is inspired by [5]. Our model has the same architecture, but the GRU cell replaces LSTM.

CatBoost. CatBoost[1] is a machine learning algorithm that uses gradient boosting on decision trees. It is available as an open source library, developed by the Russian software company Yandex. Nowadays, it is considered one of the most powerful gradient boosting algorithms. Therefore, it is promising to try this model in PPM task.

The main problem that occurs is that gradient boosting is an algorithm of classical machine learning, so it requires tabular data. It means that every data object must be represented as a vector of fixed dimension. In this paper, as such a vector representation, we used the concatenation of the embeddings of an event feature representation and its next activity as a target. The problem of not choosing a method of a sliding window is a fact that datasets may have traces of size 2, and the constraint of fixed dimension forces us to take the minimum of lengths.

In this paper, we use fine-tuned CatBoost along with different log embeddings as our approach to solve PPM problem.

4 Evaluation

4.1 Datasets

Most of the previous works apply developed methods to two main publicly available datasets: **Helpdesk**[2] and **BPI12W**[3].

- **Helpdesk** dataset is event log concerning the ticketing management process of the Helpdesk of an Italian software company.
- **BPI12W** dataset is event log of a loan application process of a large German financial institution.

[1] https://catboost.ai
[2] https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb
[3] https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f

Despite the prevalence of these datasets, they are quite trivial and far from real life. There are less than 10 different activities in each dataset, which makes traces too short. Therefore it does not allow any model to fully identify underlying patterns in the data. Nevertheless, many earlier authors measured their results using **Helpdesk** and **BPI12W**, so we are forced to use it for testing in order to be able to compare the quality.

Nevertheless, the latest research has moved away from the classic **Helpdesk** and **BPI12W**. In recent works the quality of proposed methods are estimated using more complex and modern **BPI12**[4] and **BPI17**[5] datasets.

These two datasets are newer versions of **BPI** datasets. There are about 30 different activities with very unbalanced distribution. The size of the datasets is much larger, which is an important aspect for training deep neural networks and the average length of traces is significantly longer, especially in **BPI17**. This makes data more realistic.

4.2 Experimental results

In this section we are going to compare the results of the previously described models.

Table 2. Test results comparison for T-LSTM models with different decay functions.

Original TLSTM test results				
Dataset	Helpdesk	BPI12W	BPI12	BPI17
Accuracy	0.724	0.778	-	-
Exponential TLSTM test results				
Dataset	Helpdesk	BPI12W	BPI12	BPI17
Accuracy	0.775	0.787	0.849	0.879

In Table 2 there are represented the results of time-aware LSTMs for next activity prediction. A small change of the decay function led to a significant improvement in quality.

Regarding the attention models, this architecture did not produce any strong quality boost either LSTM-based and GRU-based. The results are shhown in Table 3. his situation can be explained by the assumption that in the explored datasets the next event depends only on several previous ones, so it is impractical to take into account the entire prefix as we do in attention mechanism.

The best quality was achieved using various types and modifications of recurrent neural networks, specifically LSTM + GRU model. This architecture is enough complex to be able to extract underlying patterns in the data. Thus, this model was mainly used when testing our own embeddings and new features. The achieved results are represented in Table 4.

[4] https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f
[5] https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb

Table 3. Test results comparison for different recurrent neural networks with attention layer.

Original LSTM with attention test results				
Dataset	Helpdesk	BPI12W	BPI12	BPI17
Accuracy	0.833	0.723	0.816	-
GRU with attention test results				
Dataset	Helpdesk	BPI12W	BPI12	BPI17
Accuracy	0.735	0.773	0.849	0.878

Source code is available at our GitHub repository[6].

Table 4. Test accuracy with standard deviation for neural network model using different embedding techniques.

Embeddings type	Helpdesk	BPI12W	BPI12	BPI17
pure_traces	0.747 ± 0.004	0.774 ± 0.004	0.840 ± 0.010	0.874 ± 0.001
glove_extended	0.833 ± 0.005	0.788 ± 0.003	0.844 ± 0.009	0.892 ± 0.009
glove_raw	0.844 ± 0.001	0.781 ± 0.014	0.844 ± 0.005	**0.903 ± 0.001**
more_categorical	**0.846 ± 0.001**	0.773 ± 0.013	0.853 ± 0.001	0.901 ± 0.001
no_drop_cases	-	-	0.843 ± 0.002	0.873 ± 0.006
raw_activity	**0.846 ± 0.001**	**0.796 ± 0.003**	**0.854 ± 0.002**	0.894 ± 0.006
trace2vec	0.844 ± 0.001	0.779 ± 0.011	0.853 ± 0.004	0.887 ± 0.002

In this paper we also used a gradient boosting model for activity prediction. Unexpectedly, the CatBoost model showed the best quality for three out of the four datasets and surpassed deep learning approaches. This, in turn, confirms the hypothesis that the last few events have the greatest influence in the presented log datasets.

Table 5. Test accuracy for CatBoost model using different embedding techniques.

Embeddings type	Helpdesk	BPI12W	BPI12	BPI17
pure_traces	-	-	-	-
glove_extended	0.866	**0.8**	0.863	0.85
glove_raw	**0.868**	**0.8**	0.864	**0.86**
more_categorical	**0.868**	**0.8**	0.86	0.846
no_drop_cases	-	**0.8**	0.85	0.855
raw_activity	0.866	**0.8**	**0.865**	-
trace2vec	**0.868**	**0.8**	0.864	0.849

[6] https://github.com/serp404/ML4PPM

5 Acknowledgments

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 20-37-90136. The reported study was funded by RFBR, project number 20-37-90136.

6 Conclusion

In this paper we propose an approach to predict next activity label given a trace prefix. The approach consists of several log embedding techniques, using modified predictive models, including different RNN variations and gradient boosting. Our results show that applying deep learning approaches along with hyperparameters tuning and embeddings lead to higher accuracy even on complex datasets compared to baseline methods. Finally, we reach better quality on some event logs using CatBoost as a predictive model, having lower training complexity.

As for future work, we will investigate the applicability of our approaches for other PPM tasks. Especially we are going to experiment with next timestamp prediction task.

References

1. Wil van der Aalst. 2016. Process Mining. Data Science in Action. DOI 10.1007/978-3-662-49851-4
2. Chiara Di Francescomarino, Chiara Ghidini, Fabrizio Maria Maggi, Fredrik Milani. 2018. Predictive Process Monitoring Methods: Which One Suits Me Best? arXiv:1804.02422 [cs.AI]
3. Irene Teinemaa, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi. 2018. Outcome-Oriented Predictive Process Monitoring: Review and Benchmark arXiv:1707.06766 [cs.AI]
4. Stefan Falkner, Aaron Klein, Frank Hutter. 2018. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. ICML2018.arXiv:1807.01774 [cs.LG]
5. Abdulrahman Jalayer, Mohsen Kahani, Amin Beheshti, Asef Pourmasoumi and Hamid Reza Motahari-Nezhad. 2020. Attention Mechanism in Predictive Business Process Monitoring. IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC), Eindhoven, Netherlands, 2020, pp. 181-186. DOI: 10.1109/EDOC49727.2020.00030
6. Pieter De Koninck, Seppe vanden Broucke, Jochen De Weerdt. 2018. Act2vec, trace2vec, log2vec, and model2vec: Representation Learning for Business Processes. Lecture Notes in Computer Science (LNCS, volume 11080). DOI 10.1007/978-3-319-98648-7_18
7. Quoc V. Le, Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. arXiv:1405.4053 [cs.CL]
8. Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, Ameet Talwalkar. 2016. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. Journal of Machine Learning Research 18 (2018) 1-52. arXiv:1603.06560 [cs.LG]

9. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs.CL]
10. Nicolò Navarin, Beatrice Vincenzi, Mirko Polato, Alessandro Sperduti. 2017. LSTM Networks for Data-Aware Remaining Time Prediction of Business Process Instances. IEEE Symposium on Deep Learning (IEEE DL'17) @ SSCI. arXiv:1711.03822v1 [cs.LG]
11. An Nguyen, Srijeet Chatterjee, Sven Weinzierl, Leo Schwinn, Martin Matzner, Bjoern Eskofier. 2020. Time Matters: Time-Aware LSTMs for Predictive Business Process Monitoring. 1st International Workshop on Leveraging Machine Learning in Process Mining (ML4PM). arXiv:2010.00889v3 [cs.LG]
12. Jeffrey Pennington, Richard Socher, Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. DOI 10.3115/v1/D14-1162
13. Niek Tax, Ilya Verenich, Marcello La Rosa, Marlon Dumas. 2017. Predictive Business Process Monitoring with LSTM Neural Networks. Lecture Notes in Computer Science, 10253 (2017) 477-492. arXiv:1612.02130v2 [stat.AP]
14. Sven Weinzierl, Sandra Zilker, Jens Brunk, Kate Revoredo, Martin Matzner, Jörg Becker. 2020. XNAP: Making LSTM-based Next Activity Predictions Explainable by Using LRP. arXiv:2008.07993v3 [cs.AI]
15. Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. arXiv:1607.04606 [cs.CL]
16. Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. arXiv:1409.0473 [cs.CL]
17. Farbod Taymouri, Marcello La Rosa, Sarah Erfani, Zahra Dasht Bozorgi, Ilya Verenich. 2020. Predictive Business Process Monitoring via Generative Adversarial Nets: The Case of Next Event Prediction. arXiv:2003.11268 [cs.LG]
18. Farbod Taymouri, Marcello La Rosa, Sarah M. Erfani. 2021. A Deep Adversarial Model for Suffix and Remaining Time Prediction of Event Sequences. arXiv:2102.07298 [cs.LG]
19. Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár. 2017. Focal Loss for Dense Object Detection. arXiv:1708.02002 [cs.CV]
20. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. 2017. Attention Is All You Need. arXiv:1706.03762 [cs.CL]
21. Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]
22. Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv:2010.11929 [cs.CV]