Assessment of "Prediction-based Resource Allocation using LSTM and minimum cost and maximum flow algorithm"

by Gyunam Park and Minseok Song at ICPM 2019

Term paper - So.Se. 2023

Machine Learning Applications in Process Mining

Date: 28.07.2023

Sourabh Satish Zanwar

sourabh.zanwar@rwth-aachen.de

Abstract. Predictive business process monitoring is an approach that analyzes completed cases of a business process to predict the behavior of running instances. Recent research in this field aims to increase the efficiency and productivity of a business process by anticipating potential issues during its execution. However, these studies often fail to suggest concrete actions to improve the process, leaving it to the user's subjective judgment. This paper introduces a novel method that connects the outcomes of predictive business process monitoring to actual process improvements. The proposed method optimizes resource allocation in a non-clairvoyant online environment with limited information by exploiting predictions. The approach integrates offline prediction model construction, which uses LSTM to forecast the processing time and the next activity of an ongoing instance, with online resource allocation based on the minimum cost and maximum flow algorithm. To evaluate the proposed method, the authors conducted experiments with an artificial event log and a real-life event log from a global financial organization. We futher investigate this technique by including the exploration of different models, such as bi-directional LSTM models, GRUs and CNNs. From the experiment we find that the CNN based prediction model has the best performance in terms of weighted completion time, this is because of better generalization of the CNN model as opposed to overfitting in other RNNs.

1 Introduction

1.1 Goal of the reference paper

The aim of this work reference to original paper [22] is to solve a non-clairvoyant online-over-time problem [23] with the two phase method integrating predictions with resource allocation optimization and evaluate this proposed method for effectiveness and efficiency on both artificial and real-life event log data.

1.2 Goal of the current paper

The primary objective of this paper is to replicate the method proposed in the original research paper. However, it is important to address a significant limitation of the method, which relates to the performance of the prediction model. To gain deeper insights into this limitation, our study aims to explore the effectiveness of various predictive models, including Bi-directional LSTM, Gated Recurrent Units (GRUs), and Convolutional Neural Networks (CNNs). These models will be applied to predict both the next activity and the processing time in conjunction with resource allocation based on the Minimum Cost Maximum Flow (MCMF) and Network Simplex method.
By employing multiple predictive models, we seek to comprehensively evaluate their performance and determine which approach yields the most accurate and reliable predictions. The utilization of BiLSTM, GRUs, and CNNs allows for the exploration of different architectures that capture temporal and spatial patterns in the data. Moreover, coupling these prediction models with resource allocation techniques such as MCMF and Network Simplex enhances the overall decision-making process and ensures an optimized allocation of resources.
Through our experimental investigations, we aim to shed light on the strengths and weaknesses of each predictive model and its impact on the resource allocation process. By considering a range of models and methodologies, we can provide valuable insights into the predictive capabilities and efficiency of different approaches for resource allocation in real-world scenarios.

1.3 Structure of the current paper

The paper consists of seven sections, not including the Bibliography. In the first section, the authors introduce the problem and provide an overview of their work in this area. The second section defines the preliminary concepts and includes the problem statement. The third section discusses the baseline approach and provides insights into the problem. The authors use a running example throughout the paper to illustrate their approach. The fourth section outlines the two-phase method introduced by the authors and provides details on the construction of the models. This section also includes subsections for Time and Next Activity Prediction and Resource Scheduling. In the fifth section, the authors evaluate their approach using both artificial and real event logs. The sixth section discusses related work, specifically predictive business monitoring

processes and job shop scheduling. Finally, the paper concludes with a discussion of limitations and future research directions.

1.4 Summary of the current paper

The paper discusses the use of process mining techniques to extract insights from event logs in Process-Aware Information Systems (PAISs) [30]. It specifically focuses on predictive business process monitoring, which provides predictions on the future status of ongoing cases of a business process. The authors suggest that the prediction results can be transformed into concrete improvement actions to improve business processes, specifically in the area of resource allocation. Resource allocation is a challenging problem in business process management, and the authors propose a two-phase method to optimize resource allocation based on predictions to efficiently solve the non-clairvoyant online-over-time problem. This problem occurs when decision-makers do not know the data concerning a job when the job is presented to them. The proposed method integrates predictions with resource allocation optimization to minimize the total weighted completion time. The authors evaluate their proposed method on both an artificial and a real-life event log to demonstrate its effectiveness and efficiency.

1.5 Applications

In order to assess the effectiveness of the proposed method, we conducted a comprehensive evaluation using a real-life example from the Business Processing Intelligence Challenge 2012 (BPIC'12) dataset [2]. The dataset pertains to the processing of personal loan or overdraft applications at a global financing organization over a duration of six months. By employing this real-life scenario, we aimed to replicate the complexities and intricacies of resource allocation in a practical setting. The BPIC'12 dataset provides valuable insights into the challenges and dynamics involved in processing loan and overdraft applications, including the various activities, resources, and temporal dependencies within the process. By utilizing this rich dataset, our evaluation encompasses a realistic representation of resource allocation scenarios, allowing for a more comprehensive analysis and accurate assessment of the proposed method's performance. Moreover, the utilization of a real-life example enhances the practicality and applicability of our findings. By considering a genuine organizational context and industry-specific processes, we can draw meaningful conclusions about the effectiveness and potential benefits of the proposed method in real-world scenarios. This real-life evaluation serves as a robust validation of the method's utility and efficacy, providing valuable insights for organizations seeking to optimize resource allocation in similar domains.

Notation	Description
I, R, A	Set of instances, resources, and activities
WI	Set of work items
wi_{ik}	kth work item of instance I_i
$p_{i,k,j}$	Processing time of work item $wi_{i,k}$ by resource r_j
ri_i	Remaining time for I_i
rr_j	Remaining time for R_j to be ready
S_i	Start time of I_i
F_i	Finish time of I_i
C_i	Completion time of I_i $(F_i - S_i)$
w_i	weight of I_i

Table 1. Notations and Symbols used throughout the paper

2 Background

2.1 Resource Allocation

Definitions
For an event universe $\mathcal{E}$, where each event has the features activity, resource and timestamp, following are the definitions of the preliminaries required for the remainder of the paper.
Trace: A trace $\sigma = e_1, e_2, e_3, ..., e_n \in \varepsilon^*$ such that for $1 \leq i < j \leq |\sigma| : e_i \neq e_j$

Event Log: An event log is $\mathcal{L}$ is a multi-set of traces such that each event appears at most once in the entire log.

Prefix, Suffix: A prefix of length $k(0 < k < n)$ of a trace $\sigma = \langle e_1, e_2, ..., e_n \rangle$ is $h_k(\sigma) = \langle e_1, e_2, ..., e_k \rangle$ and its suffix is $t_k(\sigma) = \langle e_{k+1}, e_{k+2}, ..., e_n \rangle$. For instance, for $\sigma_i = \langle e_1, e_2, e_3, e_4, e_5 \rangle, h_3(\sigma_i) = \langle e_1, e_2, e_3 \rangle$ and $t_3(\sigma_i) = \langle e_4, e_5 \rangle$.

Event Representation Function: A function $\pi_{\mathcal{A}} \in \mathcal{E} \rightarrow \mathcal{A}$ assigns process activities to each event, and a function $\pi_{\mathcal{R}} \in \mathcal{E} \rightarrow \mathcal{R}$ assigns each event to a resource. A function $\pi_{\mathcal{T}} \in \mathcal{T} \rightarrow \mathcal{A}$ assigns processing times to events.
Throughout the paper, various notations are utilized to represent different concepts and variables. These notations are listed in Table 1 for easy reference and comprehension.

2.2 Problem Statement

Following section contains the problem statement of the work

Non-clairvoyant Online Job Shop Scheduling:
For a set of instances I, we have a set of tasks that need to be done in a particular order for a set of jobs. The non-clairvoyant online job shop scheduling

problem tries to schedule all the tasks to be done while minimizing the $\sum w_i C_i$, where w_i is the weight of instance i (I_i), and C_i is the difference betweeen the finish time, F_i and start time S_i for that instance. In our current context, we can establish a set of assumptions that form the foundation of our understanding. These assumptions are as follows:

1. We possess knowledge solely regarding the weight of each instance.
2. Our awareness of the next operation is solely acquired once the ongoing operation of an instance concludes.
3. Each operation necessitates particular resources, and it is imperative to note that only one operation can be performed on an instance at any given time.
4. Once initiated, an operation cannot be halted until its completion.

By acknowledging these assumptions, we establish a framework upon which our subsequent analysis and decision-making processes will rely.

2.3 Machine Learning Models

Long Short Term Memory Models (LSTM): LSTM[12] is a type of recurrent neural network (RNN) architecture designed to address the vanishing gradient problem by incorporating memory cells and gating mechanisms. It is effective in capturing long-range dependencies in sequential data. LSTM models can effectively capture the dependencies and patterns in a sequence of previous activities, enabling accurate predictions of the next activity. It requires a reasonable amount of training time to learn the intricate temporal dynamics of the event log.

Bi-directional LSTM: Bi-directional LSTM[27] is an extension of LSTM that processes input sequences in both forward and backward directions. By considering the past and future context simultaneously, it enables the model to capture dependencies in both directions and improve its understanding of the input sequence. By considering both the past and future context of previous activities, bi-directional LSTM models can make more informed predictions of the next activity and the time required for it. However, due to the increased complexity of processing sequences in both directions, training bi-directional LSTM models may take longer compared to traditional LSTM models.

Gated Recurrent Units: GRU [4] is another variant of the RNN architecture that addresses the vanishing gradient problem. It incorporates simplified gating mechanisms by combining the forget and input gates, making it computationally efficient while still effective for sequence modeling tasks. GRU models, with their simplified gating mechanisms, can quickly learn and capture the dependencies in the event log for predicting the next activity and estimating the time required. They generally require less training time compared to LSTM models, making

them a suitable choice for scenarios where computational efficiency is crucial.

Convolutional Neural Networks: CNN [16] is a type of neural network commonly used for analyzing visual data. It leverages convolutional layers to automatically learn hierarchical representations from the input data, enabling it to capture spatial dependencies and detect patterns. Although not the most ideal, CNNs can also be well-suited for tasks involving sequential data, such as event logs, as they can effectively learn patterns and dependencies between previous activities to predict the next activity and estimate the time required.

2.4 Optimization Algorithm

Minimum Cost Maximum Flow (MCMF) Algorithm: The Minimum Cost Maximum Flow (MCMF) algorithm [10] is an optimization algorithm used to find the most efficient flow of resources through a network, while minimizing the associated costs. It combines the concepts of maximum flow and minimum cost to solve resource allocation problems. In a network, resources are represented by nodes, and the flow of resources is represented by directed edges between the nodes. Each edge has a capacity that indicates the maximum amount of flow it can accommodate. The objective of the MCMF algorithm is to determine the maximum flow that can be achieved from a source node to a sink node while minimizing the total cost of the flow. To solve this problem, the MCMF algorithm follows an iterative process. It starts by initializing the flow values to zero for all edges in the network. Then, it repeatedly searches for an augmenting path, which is a path from the source to the sink that has available capacity for additional flow. In each iteration, the algorithm searches for the augmenting path with the minimum total cost. The cost of each path is determined by the costs associated with the edges and the amount of flow that can be added. Once an augmenting path is found, the algorithm increases the flow along that path, adjusting the flow values and updating the costs accordingly. The process continues until no more augmenting paths can be found, indicating that the maximum flow has been achieved. At this point, the algorithm outputs the maximum flow and the corresponding minimum cost. It is commonly used in transportation and logistics, network optimization, and other domains where efficient resource allocation is crucial.

3 Methodology

3.1 Running Example

To enhance our understanding of the task at hand, let's explore a practical example that will help illustrate the different methods involved. In this example, we consider a series of instances, each comprising a work item and a corresponding resource node that the work item requires.

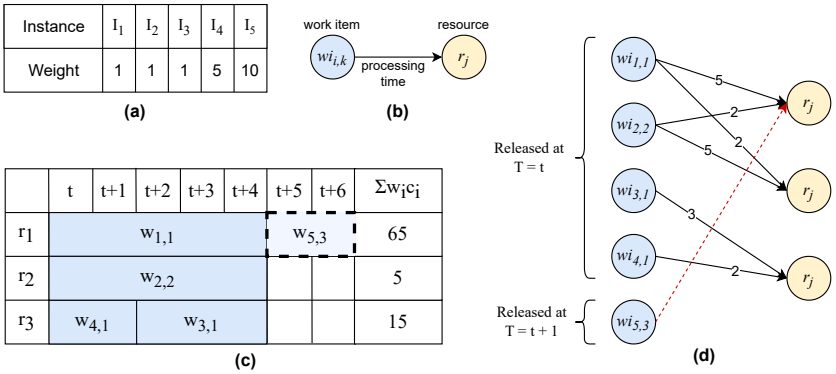


Fig. 1. Running example, initial setting and baseline allocation

Let's denote each work item as $wi_{i,k}$, where i represents the instance I_i, and k denotes the kth work item within that instance. Furthermore, these work items require specific resources, denoted as r_j. The relationship between a work item $wi_{i,k}$ and a resource r_j is represented by an arc, indicating that the work item can be processed by the resource. Additionally, the label assigned to this arc represents the processing time required to complete the work item. Each instance I_i carries a weight that is associated with it. A visual representation of this concept can be found in Figure 1(b). The weights of various instances are listed in Figure 1(a).

By examining this running example, we can gain a clearer understanding of how different methods and strategies come into play when assigning resources and optimizing the overall process. In Figure 1(d) we can see the initial setting of the resource allocation at time $T = t$.

3.2 Baseline Method

In this section, we introduce a baseline solution called WeightGreedy [15] to tackle this challenge. The basic principle behind WeightGreedy is to assign each work item to an available resource in a "first come, first served" manner. If there are conflicting demands for the same resource, the work item with a higher weight is given priority. In cases where multiple work items have the same weight, a random selection is made to break the tie.
Let's consider an example to illustrate the resource scheduling under the WeightGreedy approach. Suppose there are four work items released at a specific time. Two of these work items, $wi_{1,1}$ and $wi_{2,2}$, require the same resource, let's say r_1, and they both have a weight of 1. In this case, we randomly assign $wi_{1,1}$ to r_1. Then, we move on to the next available resource, r_2, and assign $wi_{2,2}$ to it. Now, let's consider another resource, r_3, which has two demands. One work item, $wi_{3,1}$, has a weight of 1, and another work item, $wi_{4,1}$, has a weight of 5.

Since $wi_{4,1}$ has a higher weight, we allocate r_3 to it. If at a later time, a new work item, $wi_{5,3}$, is released, we need to check if any resources are available. If no resources are available, $wi_{5,3}$ remains in the waiting list until a resource becomes free.
The WeightGreedy approach continues to assign work items to available resources based on their weights and resource availability. The completion times for each resource are calculated based on the assigned work items. It is important to note that the WeightGreedy approach serves as a baseline for resource allocation. However, due to the limited information about the process, this solution may not always yield the optimal assignment. Further exploration and refinement of resource allocation strategies are necessary to achieve more efficient and effective results. An resource allocation done by the baseline can be seen in Figure 1(c).

3.3 Proposed Methods

Proposed method is a two phased method. There are two parts in the first phase, one with prediction of next activity and then the second part is prediction of required time for the activity [31][24]. These are used for getting information on next activity and the amount of time it will require. In the second phase we use this predicted information, i.e. the next activity and the time required for the activity for the allocation of resources using Min cost max flow algorithm.

Phase 1
Model Construction: Within this paper, we adopt an approach, as proposed in the literature [29], to tackle the resource allocation problem. Our chosen methodology revolves around the utilization of Long Short-Term Memory (LSTM) neural networks, a powerful and sophisticated deep learning technique. By employing LSTM networks, we aim to leverage their inherent ability to capture temporal dependencies and patterns in the data.
In this step, our objective is to develop and train two distinct functions that play a pivotal role in our predictive framework. The first function, denoted as f_a, focuses on the prediction of the next activity in the resource allocation process. Concurrently, the second function, referred to as f_t, is designed specifically to estimate the required time for processing the identified activity. In the proposed framework, both functions f_a and f_t operate on input matrices of dimensions $M \times N$, where M represents the number of past events considered, and N is defined as the sum of the cardinalities of the unique activity set $\mathcal{A}$ and the unique resource set $\mathcal{R}$, derived from the event log. The event log serves as the primary source for obtaining the target outputs of these functions, namely the next activity and the corresponding processing time. The output of f_t is a numerical value representing the estimated processing time, while the output of f_a is a vector of dimensions $|\mathcal{A}|$, which signifies a one-hot encoded representation of the predicted next activity. Through this approach, the framework aims to leverage the rich information embedded within the event log to facilitate accurate predictions of the next activity and its associated processing time.

Let's consider an example to illustrate the concepts. Suppose we have a model that we train using a sequence of events, denoted as $\sigma 1 = \{e_1, e_2, e_3, e_4\}$. We transform the first two events of $\sigma 1$, which are e_1 and e_2, into one-hot-encoded vectors. To do this, we assign a unique binary representation to each activity and resource in the system. In this case, let's assume we have four activities (a_1, a_2, a_3, a_4) and three resources (r_1, r_2, r_3). Therefore, the length of the one-hot-encoded vector for each event is set to 7, representing the total number of unique activities and resources combined.
For the event e_1, we determine its corresponding one-hot-encoded vector by mapping its activity to a_1, and its resource to r_2. Consequently, the one-hot-encoded vector for e_1 becomes $[1, 0, 0, 0, 0, 1, 0]$. Similarly, we calculate the one-hot-encoded vector for e_2 by mapping its activity to $a2$, and its resource to $r3$. Thus, the one-hot-encoded vector for e_2 is $[0, 1, 0, 0, 0, 0, 1]$ and for e_3 the activity to a_3 and the resource to $r3$, with one-hot-encoded vector for $e3$ as $[0, 0, 1, 0, 0, 0, 1]$. Moving on to the target outputs, our first target output corresponds to the processing time of e_3. In this example, let's assume the processing time of e_3 is 4. Therefore, our first target output becomes 4. The second target output represents the activity of $e4$, which is a_3. We transform this activity label to a one-hot-encoded vector, resulting in $[0, 0, 1, 0]$ as the target of f_a.
To summarize, in this revised example, we train our model using the sequence of events $\sigma 1 = \{e_1, e_2, e_3, e_4\}$. We convert the first three events, e_1, e_2, e_3, into one-hot-encoded vectors based on their respective activities and resources. The target outputs consist of the processing time of $e_3(4)$ and the one-hot-encoded vector representing the activity of $e_4(a3)$. Figure 2 shows the process of getting the predictions from the pre-processed data.
The two LSTM models we train for f_a and f_t are trained using Mean Absolute Error loss, Adam optimizer for processing time and Cross-entropy loss and Adam optimizer for next activity prediction. There were two LSTM layers with 100 hidden units in each layer with dropout and batch normalization as regularization strategies to avoid over-fitting. All the weights were initialized using Xavier initialization, the learning rate was 0.001 and the model was trained for a maximum of 100 epochs with early stopping strategy based on the validation loss.

Next activity and time prediction: Using the prediction model developed in the previous step, we make predictions about the processing time and next activity of ongoing instances based on the event stream, which provides information about running instances. Whenever a work item in an instance is initialized, we perform two consecutive predictions for that instance. The first prediction determines the next activity of the instance. Using this prediction, we create artificial events by combining the predicted next activity with available resources. Each artificial event, along with historical events, is then used as input for the second prediction, which aims to predict the processing time of the (predicted) artificial event. We repeat this second prediction for each artificial event. The prediction results are used to improve the network, and they are updated only

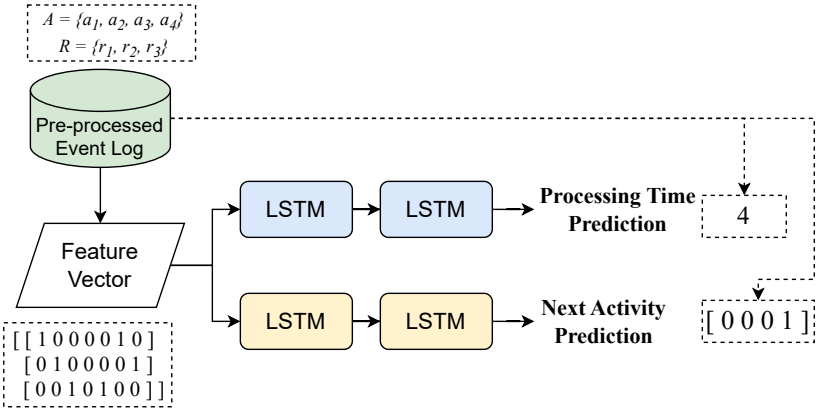


Fig. 2. Phase 1: Prediction of next activity and processing time

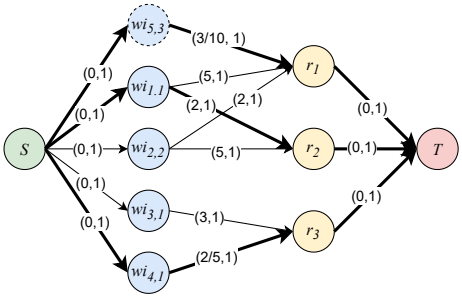
when the running instance is ready for the next operation and the previous prediction result for the next activity differs from the actual one.

Phase 2

Resource allocation: In this step, we optimize the resource allocation using Minimum Cost Maximum Flow algorithm. We construct a bi-partite graph where the left-nodes are work items and right nodes are the resources. We use the results from the next activity and processing time prediction to include the current as well as future activities. We then add edges from source node s to each work item node $wi_{i,k}$ and from each resource r_j node to the sink node t. Next, we add edges from all work items $wi_{i,k}$ to resources r_j, these edges also have the $(cost, capacity)$ as their labels. The cost is obtained by the formula $c = (p_{i,k,j} + max(ri_i, rr_j, 0))/w_i$, where $p_{i,k,j}$ is the processing time for work item $wi_{i,k}$ by resource r_j. To generate an optimal matching between work items and resources, known as a pseudo-assignment, we utilize the minimum cost maximum flow algorithm. Specifically, we employ the network simplex method [5] as our chosen algorithm for this task. The Simplex network method is an algorithm used to solve problems involving finding the best solution among many possibilities. It works by starting with an initial solution and then repeatedly improving it until the best solution is found. It does this by moving from one possible solution to another, making small adjustments each time. By iteratively making these adjustments, the Simplex method efficiently explores different options and ultimately finds the optimal solution. We can see the steps in Algorithm 1. In Figure 3(a), we can seee the application of the suggested resource allocation and in Figure 3(b) we can see the result obtained from it. From this allocation we aim to minimize the weighted sum of costs, $\sum w_i c_i$.

Algorithm 1 Resource Scheduling algorithm

Input: $\hat{WI}, \hat{R}$
Output: Psuedo-Assignment $\hat{M}$
Produce source node s, sink node t
for node $wi_{i,k} \in \hat{WI}$ **do**
  add edge $(s, wi_{i,k}, (0,1))$
end for
for node $r_j \in \hat{R}$ **do**
  add edge $(r_j, t, (0,1))$
end for
for node $wi_{i,k} \in \hat{WI}$ **do**
  for node $r_j \in \hat{R}$ **do**
    $c \leftarrow (p_{i,k,j} + max(ri_i, rr_j, 0))/w_i$
    add edge $(wi_{i,k}, r_j, (c,1))$
  end for
end for
$M \leftarrow MinCostMaxFlow(s,t)$
return M



	t	t+1	t+2	t+3	t+4	Σwᵢcᵢ
r1		w5,3		w2,2		25
r2	w1,1					5
r3	w4,1	w3,1				15

(b)

Fig. 3. Phase 2: Application of suggested resource allocation on running example

4 Data

4.1 Description

The BPIC'12 dataset [2] contained a total of 262,200 events within 13,087 cases, tracking the procedure from application submission to conclusion, such as approval, cancellation, and rejection. The process was classified into three types, and the third type, which contained events executed manually, was investigated in this study. Each case in the dataset had a case attribute, AMOUNT REQ, representing the amount requested in the application. To create equally spaced buckets based on the amount requested, the cases were divided into 10 groups, with the lowest amount in the first bucket and the highest amount in the tenth bucket. Each case was given a weight according to the bucket label it belonged to.

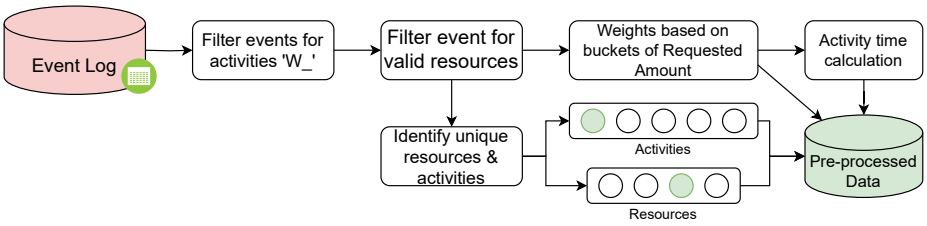


Fig. 4. Pre-processing and Filtering steps for the BPIC'12 Event log

To use the dataset as an input for phase 1 of the method, the events before 10/3/2012 were filtered and used to build an event log. The instances and resources present on 10/3/2012 in the dataset were extracted and used as components of the event stream for phase 2. For these instances, only the operations that took place on 10/3/2012 were considered to produce work items. The information on the activities and resources that the instances had experienced before 10/3/2012 was also used to make predictions. In total, 110 instances were considered, and each instance had conducted an average of three activities on the specified date.

4.2 Pre-processing

Before diving into the core analysis and decision-making processes, it is important to prepare and structure the data appropriately. This involves a series of steps to refine and organize the data for further analysis.
The following paragraphs outline the key pre-processing steps: Firstly, we filter out activities that are not carried out manually. We keep only the activities with names starting with 'W_' as these are the manual activities in the system. Next, we remove traces that do not have valid resource allocations (indicated by the value '0'). This ensures that the dataset only includes traces with meaningful resource allocations, providing a more accurate representation of the process.
Furthermore, we identify all the unique resources and activities present in the dataset. These entities are then one hot encoded into binary vectors, allowing for efficient representation and use in subsequent analysis and modeling. Additionally, we divide the requested amount for each case into ten buckets, which represent the weight assigned to each case. This categorization helps us consider the different weights associated with cases, providing a more nuanced understanding of the process dynamics.Lastly, we determine the time required for each activity by calculating the difference between the START and COMPLETE lifecycle transitions. This time metric provides insights into the duration of each activity, contributing to our understanding of the temporal aspect of the process.
Figure 4 shows the involved pre-processing steps. By following these pre-processing steps, we refine, structure, and enhance the data, creating a solid foundation for comprehensive analysis and modeling tasks.

5 Experiments

5.1 Proposed Methods in Original Paper

In this section we discuss the experiments suggested in the original paper, that we have replicated.

LSTM + MCMF: In the proposed method, a unidirectional LSTM model was employed for predicting the next activity and processing time, in conjunction with the Minimum Cost Maximum Flow (MCMF) algorithm for resource allocation. The LSTM models were trained to forecast the subsequent activity and estimate the processing time based on the current state of the case. The model architecture closely follows that outlined in the original paper, utilizing two LSTM layers, each consisting of 100 hidden units. During training, the LSTM models were subjected to a maximum of 100 epochs with early stopping implemented to prevent overfitting. For the next activity prediction task, the Mean Cross-entropy loss function was utilized, whereas the Absolute Error loss function was employed for processing time prediction. Both models were optimized using the Adam optimizer, a popular choice for training neural networks due to its adaptive learning rate. By leveraging the power of the LSTM model and carefully selecting appropriate loss functions and optimizer, we aimed to optimize the accuracy of next activity and processing time predictions. These trained models were then seamlessly integrated with the MCMF algorithm to enable efficient and informed resource allocation decisions

5.2 Additions to the Proposed Method

In this setion we discuss about the contribution of this term paper to the already suggested method for prediction and allocation of resources. From the limitations of the original paper a major concern is the performance of prediction model. For this we propose three additional models for the prediction of next activity and processing time.

BiLSTM + MCMF: Using a Bi-directional LSTM, as opposed to a single-direction LSTM, offers the advantage of leveraging information from both past and future contexts. By processing the input sequence in both directions, the model gains a more complete understanding of the temporal relationships within the data. This bidirectional approach allows the LSTM to capture dependencies that may exist both before and after a given point in the sequence. Consequently, the model becomes more effective in capturing long-range dependencies and making accurate predictions, leading to improved performance in tasks involving sequential data analysis. We use the same model architecture and replace the uni-directional LSTM with a Bi-directional one.
As the bi-directional LSTM is a more complex architecture than a normal LSTM, the training time and inference time are expected to be longer. Also, because of the more complex structure, we may be facing overfitting on the training data.

GRUs + MCMF: Using a Gated Recurrent Unit (GRU) instead of a Long Short-Term Memory (LSTM) offers several advantages in sequence modeling tasks. The GRU is a simplified variant of the LSTM, designed to streamline the learning process and reduce computational complexity. It achieves this by combining the forget and input gates of the LSTM into a single "update gate." This simplification allows for faster training and inference times. Additionally, the GRU has fewer parameters than the LSTM, making it more memory-efficient and less prone to overfitting. This characteristic is especially valuable when working with limited data or computational resources. Moreover, the GRU's architecture promotes a more streamlined flow of information within the network, enabling it to learn and adapt quickly to new patterns in the data. This makes the GRU particularly suitable for tasks where real-time or online learning is crucial. Overall, the GRU provides a balance between simplicity and performance, making it a preferred choice in many applications. These qualities make it a valuable alternative to the LSTM in sequence modeling tasks. Because of the simple architecture of the GRUs, we could train the models much faster as compared to LSTMs. As we may encounter overfitting in BiLSTM models we could also face under-fitting in GRU models. We use a similar architecture to LSTM and BiLSTM, in this model we have $13,669$ trainable parameters.

CNNs + MCMF: Using a Convolutional Neural Network (CNN) for predicting the next activity and processing time based on previous activities in a sequential task offers several benefits. CNNs are particularly effective in capturing spatial and temporal patterns in data. In the context of predicting activities, a CNN can learn to identify patterns and dependencies among different activities, even when they occur at different positions or time steps within the sequence. These networks are known for their ability to automatically extract relevant features from input data. In the case of activity prediction, a CNN can learn to recognize and extract meaningful patterns or combinations of activities that are indicative of the subsequent activity and its associated processing time. The hierarchical structure of CNNs allows them to capture both local and global dependencies within the sequence. This is particularly useful for activity prediction, as local dependencies might exist between neighboring activities, while global dependencies can arise from the overall context of the sequence. Furthermore, CNNs are robust to variations and noise in the input data, making them suitable for real-world scenarios where the observed sequences may contain inconsistencies or incomplete information. The convolutional layers in a CNN can effectively handle such variations by identifying relevant features regardless of their precise location in the sequence. We use 2 CNN layers each followed by a MaxPooling Layer, we have a total of $8,389$ trainable parameters in this model.

The final architecurte of out next activity and time prediction and resource

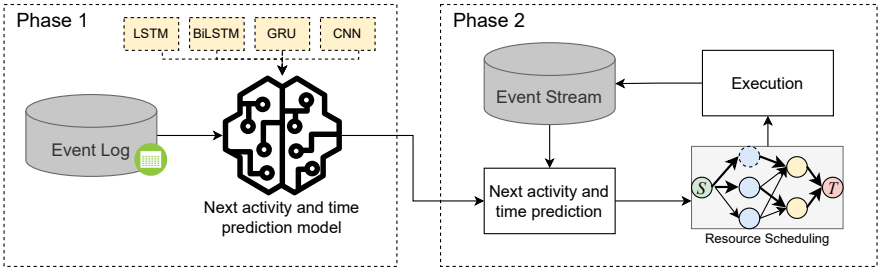


Fig. 5. Phase 1 and Phase 2: Next activity and time prediction and resource allocation

allocation can be seen in Figure 5. The code used for the experiments has been saved to a github repository[1], the code is based on the original implementation[2].

6 Results

In order to evaluate the effectiveness of our approach, we conducted simulations on real-life data from the BPIC'2012 dataset, specifically focusing on the resource allocation process on 10/3/2012. The experimental results are summarized in Table 2.

When comparing the baseline approach with our suggested approach that combines LSTM and MCMF, we observed significant differences in the weighted completion times. The baseline approach yielded a total weighted completion time of 2695, whereas our LSTM + MCMF approach resulted in a weighted completion time of 1823, representing an increase of approximately 43% compared to the baseline. To explore alternative model architectures, we experimented with other models instead of LSTM. Surprisingly, we found that simpler models exhibited better performance. For instance, the GRU + MCMF approach achieved a completion time of 1658, which is 62% less than the baseline approach. Even more impressively, the CNN + MCMF approach demonstrated an outstanding weighted completion time of 807, which is approximately 2.3 times faster than the baseline.

Surprisingly, contrary to previous research findings, our experimental results showcased that the Convolutional Neural Network (CNN) model outperformed the Recurrent Neural Network (RNN) models in our resource allocation task. Traditionally, RNN models such as LSTM and GRU have been widely favored for sequence prediction tasks due to their ability to capture temporal dependencies. However, our findings revealed that the CNN + MCMF approach achieved superior performance compared to the LSTM and GRU models. This unexpected outcome suggests that the inherent strengths of CNNs in capturing spatial patterns and extracting meaningful features from matrix-like data were particularly

[1] `https://github.com/sourabh-zanwar/seminar-mlapm-ss2023`
[2] `https://github.com/gyunamister/prediction_based_resource_allocation`

	Method	Weighted completion	Computation Time	Prediction Time
Suggested in original paper	**Baseline**	2695	60	56
	LSTM + MCMF	1823	3151	3145
Additional Prediction Models	**BiLSTM + MCMF**	1928	3194	3189
	GRU + MCMF	1658	3266	3261
	CNN + MCMF	**807**	3645	3639

Table 2. Experimental Results on the BPIC'12 Event Log

well-suited for our resource allocation scenario. These findings challenge conventional assumptions and highlight the importance of exploring various model architectures to identify the most suitable approach for a specific task.
The observed differences in computation times primarily stem from the prediction of the next activity and processing time. As shown in Table 2, a significant portion of the computation time is dedicated to the prediction phase. This is because our approach involves assigning the most efficient resources to work items and reserving resources for instances with higher weights. Due to the numerous arcs between instances and resources, each work item presents several resource options, which necessitates extensive computations for accurate parameter predictions.
Overall, our findings highlight the impact of different model architectures on the efficiency and accuracy of resource allocation. While more complex models such as Bi-directional LSTM showed improvement compared to the baseline, they were outperformed by simpler architectures like GRU and CNN. These results emphasize the significance of selecting appropriate model architectures that strike a balance between prediction power and computational efficiency, ultimately leading to more efficient resource allocation.

7 Comparison with related research

In this paper, the task is mainly assigning the resources in a efficient way based on not only current activities but also activities in the future.

7.1 Predictive Modeling

Prediction-based resource allocation has garnered significant attention in the research community, with numerous attempts made to optimize resource allocation through predictions. While our approach focuses on predicting the next activity and processing time, it is worth noting that prediction tasks in resource allocation extend beyond single activity forecasting. Researchers have explored predictions involving sequences of activities [6][28], remaining time estimation[13][21], case outcomes[19][32], and even required resources [17][3]. These diverse prediction features enable a more comprehensive understanding of the process dynamics and facilitate more informed resource allocation decisions.

In terms of model architectures, researchers have employed various techniques to tackle prediction-based resource allocation[25]. Traditional feed-forward neural networks have been utilized, leveraging their ability to capture complex patterns in the data. Recurrent neural networks (RNNs) such as Long Short-Term Memory (LSTM) [9][21] and Gated Recurrent Unit (GRU) [11] networks have also proven effective in handling sequential data and capturing dependencies over time. Additionally, autoencoder networks have been employed [18], enabling the extraction of latent representations for prediction tasks. Notably, convolutional neural networks (CNNs) have demonstrated efficiency in handling matrix-like data structures, making them suitable for prediction-based resource allocation scenarios. The application of CNNs allows for effective feature extraction from resource allocation data, enhancing prediction accuracy and aiding in informed resource allocation decision-making [7][1].

7.2 Resource allocation and scheduling

In the second phase of our method, we employed the Minimum Cost Maximum Flow (MCMF) algorithm for resource allocation, which optimizes the allocation of resources to tasks based on cost considerations. However, there are other algorithms available for solving similar resource allocation problems. One notable algorithm is the Hungarian algorithm, proposed by Kuhn in 1955, which efficiently solves assignment problems by assigning resources to tasks to optimize a given objective [14]. Additionally, the dynamic Hungarian algorithm, introduced by Mills in 2007, addresses scenarios where the costs of execution may change dynamically [20].

Another algorithm, Ant Colony Optimization (ACO)[8], draws inspiration from the foraging behavior of ants in finding optimal paths. ACO algorithm can be adapted for resource allocation problems by representing resources as pheromones and allowing agents to navigate the solution space using these pheromone trails. This approach facilitates the identification of resource allocations that lead to efficient solutions.

Furthermore, Rugwiro et al. explored the combination of ACO and deep reinforcement learning for task scheduling and resource allocation, demonstrating the potential of integrating different techniques for improved resource allocation outcomes [26]. Their work highlights the benefits of leveraging ACO in conjunction with deep reinforcement learning to optimize resource allocation processes. These algorithms, including the Hungarian algorithm, dynamic Hungarian algorithm, and ACO, offer diverse approaches to resource allocation, catering to different problem contexts and optimization objectives. Researchers have also explored the integration of these algorithms with advanced techniques such as deep reinforcement learning, showcasing the potential for hybrid approaches to enhance resource allocation performance in various domains.

8 Summary

In conclusion, the paper proposed a novel two-phase method for improving business processes using predictive business process monitoring. The method addressed the problem of online resource allocation in non-clairvoyant online environments by predicting future behaviors of instances and resources in the process. The paper demonstrated the effectiveness and efficiency of the proposed method on both an artificial log and a log from BPIC'12. However, the method has limitations such as heavy reliance on the performance of the prediction model and relatively high computation time. To address these limitations, the paper suggests future work to extend the two-phase method to minimize potential risks in the business process and to adopt advanced dispatching techniques. Overall, the paper presents a significant contribution to the field of predictive business process monitoring and suggests exciting avenues for future research. With further research in this area to try to combat a limitation, in regards to the term paper, I examined using other model architectures, namely Bi-directional LSTMs, Gated Recurrent Units, and CNN for prediction of next activity and processing time. On evaluation of the performance it was found that using CNN architecture had the best performance in case of this data.

9 Limitations

While the proposed method shows promise for improving business processes through predictive monitoring, it does have a few limitations that should be considered. Firstly, the success of the method is highly dependent on the accuracy of the prediction model. In case of this data, the CNN has shown better performance than other model structures, however, this is not always the case [25]. Having a model that learns the nuances of a event log and generalises it well still remains a challenge. Secondly, the computation time required for this method is very high compared to the baseline approach. The execution of predictions for each newly-assigned instance increases the search space for solving a network problem, leading to a longer processing time. For the research one of the limiting factors could be availability of the data that could be used to train and test the prediction models as well as have labeled optimal resource allocation.

References

1. Abdulrhman Al-Jebrni, Hongming Cai, and Lihong Jiang. Predicting the next process event using convolutional neural networks. In *2018 IEEE International Conference on Progress in Informatics and Computing (PIC)*, pages 332–338. IEEE, 2018.
2. R. P. Jagadeesh Chandra Bose and Wil M. P. van der Aalst. Process mining applied to the bpi challenge 2012: Divide and conquer while discerning resources, 2013.

3. Manuel Camargo, Marlon Dumas, and Oscar González-Rojas. Learning accurate lstm models of business processes. In *Business Process Management: 17th International Conference, BPM 2019, Vienna, Austria, September 1–6, 2019, Proceedings 17*, pages 286–302. Springer, 2019.
4. Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.
5. W. H. Cunningham. A network simplex method, Dec 1976.
6. Chiara Di Francescomarino, Chiara Ghidini, Fabrizio Maria Maggi, Giulio Petrucci, and Anton Yeshchenko. An eye into the future: Leveraging a-priori knowledge in predictive business process monitoring, 2017.
7. Nicola Di Mauro, Annalisa Appice, and Teresa MA Basile. Activity prediction of business process instances with inception cnn models. In *AI* IA 2019–Advances in Artificial Intelligence: XVIIIth International Conference of the Italian Association for Artificial Intelligence, Rende, Italy, November 19–22, 2019, Proceedings 18*, pages 348–361. Springer, 2019.
8. Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.
9. Joerg Evermann, Jana-Rebecca Rehse, and Peter Fettke. Predicting process behaviour using deep learning. *Decision Support Systems*, 100:129–140, 2017.
10. Makhlouf Hadji and Djamal Zeghlache. Minimum cost maximum flow algorithm for dynamic resource allocation in clouds. In *2012 IEEE Fifth International Conference on Cloud Computing*, pages 876–882, 2012.
11. Markku Hinkka, Teemu Lehto, Keijo Heljanko, and Alexander Jung. Classifying process instances using recurrent neural networks. In *Business Process Management Workshops: BPM 2018 International Workshops, Sydney, NSW, Australia, September 9-14, 2018, Revised Papers 16*, pages 313–324. Springer, 2019.
12. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.
13. Asjad Khan, Hung Le, Kien Do, Truyen Tran, Aditya Ghose, Hoa Dam, and Renuka Sindhgatta. Memory-augmented neural networks for predictive process analytics. *arXiv preprint arXiv:1802.00938*, 2018.
14. Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.
15. Akhil Kumar, Wil M.P. Van Der Aalst, and Eric M.W. Verbeek. Dynamic work distribution in workflow management systems: How to balance quality and performance, Jan 2002.
16. Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.
17. Li Lin, Lijie Wen, and Jianmin Wang. Mm-pred: A deep predictive model for multi-attribute event sequence. In *Proceedings of the 2019 SIAM international conference on data mining*, pages 118–126. SIAM, 2019.
18. Nijat Mehdiyev, Joerg Evermann, and Peter Fettke. A multi-stage deep learning approach for business process event prediction. In *2017 IEEE 19th conference on business informatics (CBI)*, volume 1, pages 119–128. IEEE, 2017.
19. Andreas Metzger and Adrian Neubauer. Considering non-sequential control flows for process prediction with recurrent neural networks. In *2018 44th Euromicro conference on software engineering and advanced applications (SEAA)*, pages 268–272. IEEE, 2018.

20. G Ayorkor Mills-Tettey, Anthony Stentz, and M Bernardine Dias. The dynamic hungarian algorithm for the assignment problem with changing costs. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-07-27*, 2007.
21. Nicolo Navarin, Beatrice Vincenzi, Mirko Polato, and Alessandro Sperduti. Lstm networks for data-aware remaining time prediction of business process instances. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7. IEEE, 2017.
22. Gyunam Park and Minseok Song. Prediction-based resource allocation using lstm and minimum cost and maximum flow algorithm. In *2019 International Conference on Process Mining (ICPM)*, pages 121–128, 2019.
23. Michael L. Pinedo. *Scheduling*. Springer US, 2012.
24. Mirko Polato, Alessandro Sperduti, Andrea Burattin, and Massimiliano de Leoni. Time and activity sequence prediction of business process instances. *Computing*, 100:1005–1031, 2016.
25. Efrén Rama-Maneiro, Juan C. Vidal, and Manuel Lama. Deep learning for predictive business process monitoring: Review and benchmark. *IEEE Transactions on Services Computing*, 16(1):739–756, 2023.
26. Ulysse Rugwiro, Chunhua Gu, and Weichao Ding. Task scheduling and resource allocation based on ant-colony optimization and deep reinforcement learning. *Journal of Internet Technology*, 20(5):1463–1475, 2019.
27. M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
28. Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. Predictive business process monitoring with lstm neural networks, 2017.
29. Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. Predictive business process monitoring with lstm neural networks. In *International Conference on Advanced Information Systems Engineering*, 2016.
30. Wil van der Aalst. *Process Mining*. Springer Berlin Heidelberg, 2016.
31. W.M.P. van der Aalst, M.H. Schonenberg, and M. Song. Time prediction based on process mining. *Information Systems*, 36(2):450–475, 2011. Special Issue: Semantic Integration of Data, Multimedia, and Services.
32. Nur Ahmad Wahid, Taufik Nur Adi, Hyerim Bae, and Yulim Choi. Predictive business process monitoring – remaining time prediction using deep neural network with entity embedding, 2019.