



INTERNATIONAL DOCTORAL
SCHOOL OF THE USC

Efrén
Rama Maneiro

PhD Thesis

Deep Learning Models for Predictive Monitoring of Business Processes

Santiago de Compostela, 2023

Doctoral Programme in Information Technology Research

TESE DE DOUTORAMENTO

DEEP LEARNING MODELS FOR PREDICTIVE MONITORING OF BUSINESS PROCESSES

Autor

Efrén Rama Maneiro

Directores: Manuel Lama Penín, Juan Carlos Vidal Aguiar

Titor/a: Manuel Lama Penín

PROGRAMA DE DOUTORAMENTO EN INVESTIGACIÓN EN TECNOLOXÍAS DA INFORMACIÓN

SANTIAGO DE COMPOSTELA

A mi familia

Mi mayor inspiración

*Science isn't about WHY. It's about WHY
NOT.*

Cave Johnson — Portal 2

*I survived because the fire inside burned
brighter than the fire around me.*

Joshua Graham — Fallout: New Vegas

Agradecimientos

Deseo expresar mi profundo agradecimiento a mis directores de tesis, Manuel Lama y Juan Carlos Vidal. Su inquebrantable confianza a lo largo de estos 4 años ha sido fundamental. Valoro especialmente la autonomía que me han concedido para proponer y desarrollar ideas innovadoras, así como la paciencia con la que han guiado cada etapa de esta investigación.

Mi gratitud también se extiende al Departamento de Electrónica y Computación y al Centro de Investigación en Tecnologías Inteligentes (CiTIUS) de la Universidad de Santiago de Compostela. El ambiente académico y los recursos brindados han sido cruciales para la culminación exitosa de este trabajo. No puedo dejar de mencionar a los incansables miembros del personal de apoyo del CiTIUS, cuyo esfuerzo, aunque a menudo en la sombra, ha sido esencial para mis logros. De igual manera, agradezco al CESGA por ofrecerme acceso a herramientas computacionales de vanguardia, destacando el invaluable apoyo del Finisterrae III.

Un agradecimiento especial al Profesor Fabio Patrizi, de la Sapienza Università di Roma, y a todos los compañeros del Dipartimento di Ingegneria Informatica Automatica e Gestionale (en particular a los del laboratorio B221), que me acogieron durante mi estancia en Roma e hicieron de esos 3 meses una experiencia sumamente enriquecedora.

Además, también me gustaría agradecer a todas aquellas entidades que, de una manera u otra, han financiado la presente investigación. En particular, al Ministerio de Educación, Cultura y Deporte (FPU18/05687) y al Fondo Europeo de Desarrollo Regional (FEDER). También al TAILOR Connectivity Fund por financiar mi estancia de investigación.

En lo personal, agradecer a todos los compañeros que he conocido en el CiTIUS, tanto aquellos veteranos del ya extinto LS02, como aquellos que vinieron después. También a mis amigos, tanto los de siempre como los de la carrera, con los que he compartido buenos y malos momentos. También agradecer a los compañeros de la Banda de Música de Noia, que

EFRÉN RAMA MANEIRO

me conocen desde que el saxofón era más grande que yo, y que han sido un refugio para mi mente y espíritu en los momentos más estresantes.

Por último, agradecer también a mi familia por el apoyo brindado durante estos años. En particular, a mis padres, que siempre han respetado mis decisiones y me han apoyado siempre. También a mi hermano, que me ha sido mi principal inspiración para realizar esta tesis.

10 de Noviembre de 2023

Resumo

Resúmeo en vinte palabras ou menos.

— Saitama, *One Punch-Man*

Os procesos intégranse a miúdo no noso día a día de forma inadvertida en diversos sectores, como a manufactura [60, 12], a saúde [30, 75, 48] ou a enxeñaría do software [24, 58]. Neste contexto, a análise predictiva de procesos en execución está emerxendo como unha ferramenta indispensable. A capacidade de identificar e corrir problemas de forma preventiva ofrece ás organizacións unha vantaxe competitiva crucial. Ademais, co crecemento implacábel da produción de datos, o énfase en deseñar metodoloxías para analizar datos de procesos para elaborar modelos predictivos sólidos nunca foi máis importante. Ditos modelos deben proporcionar predicións precisas para indicadores de rendemento que son de suma importancia para os interesados no proceso de negocio.

Para satisfacer estes requisitos, téñense proposto múltiples técnicas para modelar, interpretar e mellorar o rendemento dos procesos de negocio. Estas técnicas agrúpanse nunha disciplina denominada Xestión de Procesos de Negocio (BPM, polas súas siglas en inglés). A BPM establece un procedemento cíclico no cal un proceso pasa por unha serie de pasos: (i) identifícase o proceso relevante para o problema; (ii) descóbrese un modelo que reflexa a execución real do proceso; (iii) analízase o proceso utilizando medidas de rendemento cuantificábeis para detectar posibles problemas; (iv) o proceso redeséñase e fanse melloras nel en función dos problemas detectados; (v) o proceso impleméntase nunha contorna de produción; e (vi) faise un seguimento e compárase o rendemento do novo proceso co rendemento do antigo.

Un dos principais problemas das técnicas de BPM é que a maioría delas requiren intervención manual ou un alto nivel de interacción do usuario. Para resolver este problema, a

Minaría de Procesos (PM, polas súas siglas en inglés) xurde como un novo paradigma que serve de ponte entre a BPM, a Minaría de Datos e a Aprendizaxe Automática. O principal obxectivo da Minaría de Procesos é automatizar a maioría das tarefas da BPM aproveitando a información rexistrada das diferentes execucións dos procesos. Así, as técnicas de PM permiten entender o que realmente está sucedendo co proceso no lugar do que presupoñemos que está sucedendo, analizando para elo os datos reais xerados polo proceso. La maioría das técnicas de PM operan sobre un artefacto denominado *rexistro de eventos*, que se define como unha colección de trazas executadas que, á súa vez, están compostas por eventos que rexistran a execución ordenada no tempo das actividades dentro do proceso. Estes eventos tamén poden conter información sobre o momento no que foron executados, os recursos que os executaron ou información adicional como, por exemplo, o prezo dun produto que se elabora no marco do proceso.

Normalmente, las técnicas de PM divídense en tres áreas diferentes. A primeira, *descubrimiento*, encárgase de construír e refinar un modelo de proceso que encapsula o comportamento rexistrado no rexistro de eventos. Estes modelos de proceso poden clasificarse en dúas categorías: *modelos imperativos*, que capturan todas as relacións posibles entre as actividades para definir o fluxo do proceso, e *modelos declarativos*, que definen unha serie de restricións que definen o comportamento permitido entre certas actividades. Existen moitos algoritmos de descubrimento dispoñibles, como o algoritmo α , o *Inductive Miner*, o *Heuristics Miner* ou o *Split Miner*. A segunda área, *verificación de conformidade*, encárgase de comparar o modelo de proceso co rexistro de eventos para detectar canto comportamento do modelo de proceso está realmente presente no rexistro de eventos, e viceversa. Nesta área, existen tres categorías de técnicas: técnicas de *axuste*, que miden canto se desvían as execucións do rexistro de eventos do modelo de proceso; técnicas de *precisión*, que miden a cantidade de comportamento que é recollido polo modelo en comparación coas execucións do rexistro de eventos; e técnicas de *simplicidade*, que miden como de estruturalmente complexo é o modelo de proceso. A terceira área, *mellora do proceso*, ten como obxectivo analizar o funcionamento do proceso para estendelo e refinalo, co fin último de melloralo. Paga a pena sinalar que as técnicas de predición en minaría de procesos, o foco principal desta tese doutoral, forman parte desta categoría, permitindo, por exemplo, predicir os tempos de execución dun proceso con precisión.

Por outro lado, as técnicas de PM tamén poden clasificarse segundo a perspectiva dende a cal se realiza a análise do proceso. Unha das perspectivas máis comúns é a perspectiva de *fluxo*

de control, que se centra en analizar a orde de execución das actividades dentro do proceso, dirixida a revelar as restricións e dependencias entre as súas actividades . Outra perspectiva é a *temporal*, que ten como obxectivo analizar os aspectos relacionados co tempo do proceso para, por exemplo, predicir o tempo restante ata que un caso termine ou para coñecer canto tempo se perde esperando a execución da seguinte actividade. A perspectiva do *caso* céntrase en analizar as interdependencias entre diferentes atributos das distintas execucións do proceso para coñecer que posúen en común. Finalmente, outra perspectiva habitual é a *organizacional*, que ten como obxectivo analizar o proceso dende o punto de vista dos recursos que executan as actividades para, por exemplo, detectar restricións de recursos ou para predicir a súa carga de traballo. Cabe destacar que esta tese doutoral centrarase na perspectiva de fluxo de control, aínda que tamén se usarán as outras perspectivas como características de entrada as propostas neuronais.

A maioría das técnicas de minaría de procesos están enfocadas principalmente nas trazas xa finalizadas, proporcionando só información reactiva baseada nunha análise *post-mortem* realizado despois de que os problemas ocorresen. Polo tanto, a aplicación deste tipo de técnicas é bastante limitada en canto ó seu potencial preventivo. Hoxe en día, no eido da PM existe unha tendencia cada vez maior cara o desenvolvemento de técnicas de análise de execución de procesos en curso. Estas técnicas proactivas permiten a anticipación e prevención de problemas potenciais antes de que ocorran. Así, estas técnicas predictivas non só proporcionan un prognóstico pasivo; informan activamente das decisións, asegurando que os procesos permanezan aliñados cos obxectivos estratéxicos da organización. Por exemplo, na xestión da cadea de subministración, os modelos predictivos poden prever aumentos de demanda, permitindo as empresas prepararse con anticipación. No ámbito sanitario, poden predicir a afluencia de pacientes, asegurando que os recursos sexan asignados de maneira óptima. A *monitorización predictiva* é un subcampo da minaría de procesos que xurde como resposta a esta necesidade, utilizando información do rexistro de eventos para anticipar a evolución dunha instancia de proceso en curso. Na monitorización predictiva, existen varias tarefas que poden levarse a cabo, como predicir a seguinte actividade, a seguinte marca temporal, o tempo restante ata que finalice un caso en execución, o resultado do caso, etc. Esta tese doutoral céntrase en predicir o fluxo de control de instancias de procesos en curso. Polo tanto, poñen o foco en predicir, en primeiro lugar, a seguinte actividade dado un caso en execución incompleto (prefixo de eventos), e, en segundo lugar, en predicir a secuencia de actividades a partires dun prefixo de eventos (sufixo de actividades). Cabe destacar que estes dous tipos de predición

constitúen o obxectivo da tese doutoral.

Tradicionalmente, a monitorización predictiva emprega técnicas clásicas de aprendizaxe automática, como Árbores de Decisión, Máquinas de Factorización, Autómatas Finitos Probabilísticos ou Máquinas de Vectores de Soporte. Sen embargo, esta tendencia cambiou nos últimos anos debido a dous factores principais: o primeiro é o volume cada vez maior de datos xerados a partir de procesos, que require modelos predictivos máis sofisticados que sexan capaces de aproveitar esta información, e o segundo é o gran éxito que as técnicas de aprendizaxe profunda tiveron no manexo de datos non estruturados, principalmente en áreas como a Visión por Computador e o Procesamento da Linguaxe Natural. Neste contexto, as Redes Neurais Recurrentes (RNNs) xudiron como unha solución axeitada para moitas das tarefas de monitorización predictiva, principalmente debido a súa capacidade para manexar datos secuenciais e á súa capacidade para aproveitar características relevantes dos datos de entrada, diminuíndo a cantidade de enxeñaría manual de características que debe realizarse. No caso da monitorización predictiva, traballos recentes amosaron que as RNNs, e en xeral as técnicas de aprendizaxe profunda, ofrecen un rendemento comparábel ou mellor cós métodos clásicos de aprendizaxe automática, consolidándoas como a solución *de facto* para problemas novos neste campo. Sen embargo, actualmente estanse explorando outras arquitecturas de redes neuronais, como as Redes Neurais Convolucionais, as Redes Adversarias Xenerativas ou os *Transformers*.

Tipicamente, as técnicas de monitorización predictiva baseadas en aprendizaxe profunda aprenden modelos predictivos utilizando a información que está directamente dispoñible nas trazas do rexistro de eventos, como a secuencia de actividades, as características temporais e os atributos, incluíndo os recursos. Sen embargo, estes métodos ignoran a información estrutural dispoñible nos modelos de proceso. Así, cando se adestra un modelo de aprendizaxe profunda, é posible que non aprenda os patróns de comportamento, como bucles e paralelos, de forma axeitada, xa que os modelos neuronais poderían ter dificultades para detectar estes patróns só a partir da información das trazas. Polo tanto, a agregación de dita información sobre os modelos de proceso podería simplificar o adestramento da rede neuronal, posto que o modelo neuronal tería acceso directo á información estrutural. Ademais, proporcionaría información adicional sobre as posibles rutas de execución que unha instancia de proceso en curso podería tomar, o cal é particularmente útil ao predicir o sufixo de actividades restante dado un prefixo de eventos. Baseándonos nestas consideracións, podemos formular a hipótese sobre a cal se artella esta tese: **incorporar información estrutural explícita do modelo de**

proceso nun modelo de monitorización predictiva baseado en aprendizaxe profunda pode mellorar de maneira significativa o seu rendemento ao permitir unha mellor comprensión da dinámica do proceso de negocio. Así, traballando nesta hipótese, propónse un modelo neuronal para a predición da seguinte actividade e outro modelo predictivo, baseado no anterior, centrado na predición do sufixo de actividades. Logo, tendo en conta os resultados do modelo de predición do sufixo de actividades, propónse outro modelo de proba de concepto que resolve o problema de elección de actividades, o cal xurde de xeito natural ao predicir o sufixo. Cada modelo está validado pola mesma metodoloxía e comparativa experimental de tal maneira que se realiza sempre unha comparación xusta e robusta entre as distintas aproximacións.

Tendo en conta esta hipótese, neste tese doutoral, en primeiro lugar, deseñouse unha metodoloxía de avaliación de propostas de monitorización predictiva. A necesidade desta metodoloxía radica no feito de que os axustes experimentais utilizados polas aproximacións do estado da arte son inconsistentes, polo que unha avaliación xusta, robusta e integral de calquera modelo proposto é imposíbel. Polo tanto, nesta tese doutoral propoñemos unha metodoloxía fundamentada en catro alicerces: (i) unha selección integral de rexistros de eventos que varía de xeito significativo en termos da súa complexidade e características, o que garante que calquera avaliación realizada non estea sesgada cara un tipo particular de proceso; (ii) unha estratexia de preprocesado dos conxuntos de datos que ten en conta a natureza única dos datos de minaría de procesos e que emprega un sistema de avaliación de validación cruzada, de tal maneira que a estimación do rendemento do modelo sexa precisa, reducindo a probabilidade de reportar resultados enganosos; (iii) unha selección de métricas axeitadas para cada un dos problemas de monitorización predictiva que permita unificar a comparativa en torno a medidas de rendemento comúns a todas as aproximacións; e (iv) unha comparativa baseada en tests estatísticos en dúas fases que proporciona unha comparación robusta e completa para o conxunto de aproximacións probadas. Esta metodoloxía simplifica a interpretación dos resultados, facilitando a obtención de conclusións significativas.

Despois de definir a metodoloxía de avaliación, desenvolveuse TACO, un modelo neuronal profundo deseñado para predicir a seguinte actividade dunha execución de proceso en curso. TACO é único na súa capacidade para aproveitar non só á información directamente dispoñible no rexistro de eventos, senón tamén a información estrutural do modelo de proceso e as súas interaccións dinámicas co prefixo de actividades. Para realizar esta aprendizaxe, TACO utiliza Redes Neurais Convolucionais de Grafos (GCNs, polas súas siglas en inglés), que son modelos neuronais especificamente deseñados para operar en datos non-euclidianos, é dicir,

operan directamente sobre a estrutura do grafo. Para permitir o procesamento de toda a secuencia de interaccións do prefixo de actividades co modelo de proceso, TACO estende o modelo GCN a unha Rede Neuronal Recorrente de Grafos (GRNN, polas súas siglas en inglés). Este enfoque permite a TACO capturar e aproveitar as dependencias espazo-temporais entre as actividades ao longo do tempo, ofrecendo un rendemento predictivo superior. TACO foi validado e comparado con 10 aproximacións diferentes do estado do arte en 10 rexistros de eventos diferentes. Os resultados resaltan a consistencia de TACO e o seu rendemento predictivo maior que as outras aproximacións, demostrando a idoneidade das GRNNs para a monitorización predictiva de procesos.

O segundo modelo neuronal proposto é ASTON, que estende traballo realizado en TACO. ASTON é unha aproximación novidosa que aborda o problema de predición do sufixo de actividades integrando coñecemento explícito sobre o modelo de proceso. Inspirándose nas leccións aprendidas durante o desenvolvemento de TACO, ASTON mellora aínda máis o mecanismo de codificación utilizado para representar a información do modelo de proceso. Ademais, propón un mecanismo de atención que aborda os desafíos propostos polo problema de predición do sufixo. En concreto, ASTON incorpora un método novidoso para integrar a información do modelo de proceso nun esquema de codificador-descodificador, mellora o esquema de codificación da información do modelo de proceso e introduce una estratexia de selección de actividades novidosa, a cal é simple, efectiva e robusta ante as características subxacentes do rexistro de eventos. ASTON foi avaliado utilizando 12 rexistros de eventos reais e comparado con 6 aproximacións do estado do arte. Os resultados amosan que ASTON supera de xeito significativo a todas as outras aproximacións dende un punto de vista estatístico, acadando o mellor resultado en 8 rexistros de eventos, o segundo mellor en 3 e o terceiro mellor en 1. O que é máis importante, ASTON mostra robustez ante as características subxacentes do rexistro de eventos, ofrecendo un rendemento consistente entre a gran variedade de rexistros avaliados.

Finalmente, a última aproximación que se propuxo na presente tese é DOGE, que resolve un dos problemas atopados durante o desenvolvemento de ASTON. En particular, os modelos de predición do sufixo de actividades dependen da estratexia de selección utilizada para elixir cada actividade do sufixo predito. Ademais, o rendemento da estratexia de selección está directamente relacionada coas características subxacentes do rexistro de eventos. Polo tanto, aínda que a estratexia de selección proposta en ASTON parece o suficientemente robusta, aínda presenta dous hiperparámetros que deben ser axustados para cada rexistro de

eventos, non existindo ningunha garantía de que a estratexia siga sendo superior nun novo rexistro de eventos. Polo tanto, neste contexto, a estratexia de selección convértese nun novo hiperparámetro do modelo neuronal que, a pesares de calquera mellora na arquitectura do modelo, condiciona de maneira significativa o rendemento final. Por esta razón, DOGE propón un modelo neuronal para a predición do sufixo de actividades baseado en técnicas de Aprendizaxe por Reforzo (RL, polas súas siglas en inglés). RL ofrece un marco que facilita o adestramento dun modelo para maximizar a recompensa de toda a secuencia do sufixo, no lugar de cada unha das súas actividades individuais, que sería o caso de todas as aproximacións do estado do arte, excepto da aproximación baseada en GANs, as cales se adestran usando a Estimación de Máxima Verosimilitud (MLE, polas suas siglas en inglés). Así, en DOGE o problema da predición do sufixo redúcese a seleccionar sempre a actividade máis probable emitida pola política da rede, xa que a distribución de probabilidade aprendida das actividades está condicionada polo rendemento de todo o sufixo e non polas actividades individuais, polo que a "estratexia de selección óptima" aprendese implicitamente durante o adestramento. DOGE avalíouse utilizando os mesmos rexistros de eventos que ASTON, e a experimentación mostra que o rendemento de DOGE é igual ou superior ao de ASTON, sendo máis sinxelo de implementar y sen necesitar a elección manual dunha estratexia de selección de actividades.

Co desenvolvemento destes modelos neuronais, podemos considerar os obxectivos da tese doutoral completados, xa que logo os modelos predictivos desenvoltos superan as propostas actuais do estado do arte. Ademais, cabe sinalar que a metodoloxía implementada nesta tese doutoral para a comparación das aproximacións de monitorización predictiva está sendo adoptada pola comunidade de minaría de procesos.

Summary

Keep it to twenty words or less.

— Saitama, *One Punch-Man*

Processes pervade various facets of our daily existence, frequently assimilating into our routines even without our explicit acknowledgment. This phenomenon is particularly salient in sectors like manufacturing [60, 12], healthcare [30, 75, 48], and software engineering [24, 58] where structured processes are employed to support the operations. Given this context, the predictive analysis of process executions is emerging as an indispensable tool. The ability to preemptively identify and rectify issues offers organizations a distinct competitive edge. Furthermore, with the relentless growth of data production, the emphasis on devising methodologies for analyzing process data to craft robust predictive models has never been more important. Such models must provide precise forecasts for metrics that are of paramount significance to the process overseers.

In order to satisfy these requirements, multiple techniques have risen in order to allow modeling, interpreting and improving process performance. These techniques are grouped in a discipline named Business Process Management (BPM). The BPM establishes a cyclical procedure in which a process undergoes a series of steps: (i) the process relevant to the problem is identified, (ii) a model that mirrors the actual process execution is discovered, (iii) the process is analyzed using quantifiable performance measures to detect potential issues, (iv) the process is redesigned and improved based on the problems detected, (v) the process is implemented in a production environment, and (vi) the performance of the new process is monitored and compared with the performance of the old one.

One of the main problems of BPM techniques is that most of them require manual intervention or a high level of user interaction. In order to solve this issue, Process Mining

(PM) arises as a new paradigm that serves as a bridge between the BPM, the Data Mining and Machine Learning. The main task of Process Mining is to automate most BPM tasks by leveraging the information recorded from the execution of the process. Thus, PM techniques allow to understand what is genuinely happening with the process instead of what we presume is happening by the through analysis of the actual data generated by the process. Most PM techniques operate on an artifact denominated *event log*, which are a collection of executed traces made of events that record the time-ordered execution of tasks within the process. These events can also contain information about the time when they were executed, the resources that executed them, or additional information such as the cost of execution.

Normally, PM techniques are divided in three different areas. The first one, *discovery*, is concerned with constructing and defining a process model that encapsulates the behavior recorded in the event log. These process models can be classified in two categories: *imperative models*, which capture every possible relationship between the activities to define the process flow, and *declarative models*, which define a series of constraints that define the allowed behavior between certain activities. There are many discovery algorithms available, such as the α -algorithm, the Inductive Miner, the Heuristics Miner, and the Split Miner. The second one, *conformance checking*, is concerned with comparing the process model with the event log in order to detect how much behavior from the process model is actually present in the event log and viceversa. Here, there are three different categories of techniques: *fitness* techniques, which measure how much the executions of the event log deviate from the process model; *precision* techniques, which measure the amount of behavior that is supported by the model compared with the event log executions; and *simplicity* techniques, which measure how structurally complex is the process model. The third one, *process enhancement*, aims to analyze the process behavior to improve it from various perspectives. It is worth mentioning that the predictive techniques in process mining, the main focus of this dissertation, belong to this category, allowing, for example, the accurate prediction of the running times in a business process.

On the other hand, PM techniques can also be classified depending on the perspective from which they analyze the process. One of the most common perspectives is the *control-flow* perspective, which focuses on the order of execution of the activities in the process. Here, the analysis is targeted to reveal the constraints and dependencies between the activities of the process. Another perspective is the *time* perspective, which aims to analyze time-related aspects of the process. For example, the process could be analyzed to predict the remaining

time until a case finishes or to know how much time is wasted waiting for the execution of the next activity. The *case* perspective focuses on analyze interdependencies between different attributes of process instances so that common insights can be obtained. Finally, another usual perspective is the *organizational* perspective, which aims to analyze the process from the point of view of the resources that execute the activities. For example, the process could be analyzed to detect resource constraints or to predict the workload of the resources. Note that these perspectives are not mutually exclusive, and a process can be analyzed from multiple perspectives at the same time.

Processes are present in virtually every aspect of the day-to-day operations of businesses. Therefore, their efficiency and accuracy directly impact organizational success. As our modern world becomes increasingly data-driven, the predictability of processes has never been more essential. Predictive models in BPM are transformative in this regard, serving as a nexus between data insights and actionable decision-making. In a world brimming with complexities, predictive models offer a clear lens through which managers can view potential outcomes, risks, and opportunities. These models do not merely provide a passive forecast; they actively inform decisions, ensuring processes remain aligned with strategic goals. For instance, in supply chain management, predictive models can forecast demand surges, allowing businesses to prepare and capitalize. In healthcare, they can predict patient inflows, ensuring resources are optimally allocated. The beauty of predictive models lies in their ability to harness vast amounts of data, transforming them into comprehensible insights. In the realm of BPM they allow organizations to not only understand their current operations but also to envision and strategize for future scenarios. By integrating predictive models, organizations embed a proactive approach within their BPM framework, ensuring they are always a step ahead, maximizing efficiency, minimizing risks, and continuously aligning with their evolving objectives.

Most process mining techniques are largely focused on only completed traces, providing only reactive insights based on a *post-mortem* analysis done after problems have occurred. Therefore, the application of this kind of approaches is fairly limited as far as preventive potential is concerned. Nowadays, there is an ever-growing trend towards applying process mining techniques to ongoing process instances. These proactive approaches allow the anticipation and prevention of potential problems before they occur. *Predictive monitoring* is a subfield of process mining that emerges as a response to this need, using event log information to anticipate the evolution of an ongoing process instance.

In predictive monitoring, there are many tasks that can be carried out such as predicting the next activity, the next timestamp, the remaining time until a running case finish, the outcome of the case, etc. This Ph.D dissertation focuses on forecasting the control-flow of ongoing process instances. Therefore, the focus is put into forecasting the next activity given an incomplete running case (event prefix), and predicting the sequence of activities given an event prefix (activity suffix).

Traditionally, predictive monitoring is based on the use of classic machine learning techniques, such as Decision Trees, Factorization Machines, Probabilistic Finite Automats, or Support Vector Machines. However, this trend has shifted due to two main factors: the increasingly growing volume of data generated from business processes, which requires more sophisticated models that are able to take advantage from this information, and the huge traction that deep neural networks have gained in dealing with unstructured data in areas such as Computer Vision and Natural Language Processing. In this context, Recurrent Neural Networks (RNNs) have emerged as a fitting solution to many predictive monitoring tasks, primarily due to their ability to handle sequential data and their capacity to harness relevant features from input data, diminishing the amount of manual feature engineering that must be performed. RNNs, and in general deep learning techniques, have been found to deliver comparable or better performance than classical machine learning methods in predictive monitoring, solidifying them as the go-to solution. Nevertheless, other architectures are currently being explored, such as Convolutional Neural Networks, Generative Adversarial Networks, or Transformers.

Normally, deep-learning-based predictive monitoring techniques fit predictive models only using information directly available in the traces of the event log, such as the sequence of activities, time features, and attributes. However, these methods overlook the structural information available in process models. Behavioral patterns such as loops and parallels may be inadequately captured, as the neural models might struggle to detect these patterns only from trace information. Thus, adding such information about the process models could simplify the training of the neural network because the neural model would have direct access to the structural information, and it would provide additional information about the potential execution paths that an ongoing process instance could take, which is particularly useful when predicting the remaining activity suffix given an event prefix. Based on these considerations, we can formulate the hypothesis upon which this thesis is developed: **incorporating explicit structural information from the process model into a deep-learning predictive monitoring**

model can significantly enhance its performance by enabling a better understanding of the business process dynamics. In order to validate these assumptions, we experimented with the two main scenarios in predictive monitoring: next activity and activity suffix prediction. Every proposal from this dissertation is supported by a comparison methodology and benchmark that allows a fair and robust comparison between approaches

Thus, first we designed a benchmarking methodology that supports the evaluation of every proposed algorithm in this Ph.D dissertation. The need of this methodology resides in the fact that the reported experimental setups of the state-of-the-art approaches are inconsistent, so a fair, robust and comprehensive evaluation of any proposed model is impossible. Therefore, in this Ph.D dissertation we propose a methodology that is fundamented in three pillars: *(i)* a comprehensive selection of event logs that varies significantly in complexity and characteristics, which ensures that any evaluation performed is not biased towards a particular type of process; *(ii)* a dataset preprocessing strategy that takes into account the unique nature of process mining data and employs a cross-validation evaluation system, so the model performance estimation is accurate, reducing the likelihood of reporting misleading results; *(iii)* an adequate selection of metrics for each of the predictive monitoring problems that allows the unification of the comparisons around common performance measures for every approach considered; and, *(iv)* a two-stage statistical testing methodology that provides a robust and comprehensive comparison for the set of tested approaches. This methodology simplifies the interpretation of the results, making it easier to draw meaningful conclusions.

After defining the evaluation methodology, we introduced TACO, a deep neural model designed to predict the next activity in an ongoing business process instance. TACO is unique in its ability to leverage not only the information directly available in the event log, but also the structural information from the process model and its dynamic interactions with the activity prefix. To accomplish this objective, TACO takes advantage of Graph Convolutional Networks (GCNs), which are neural models that are designed to process non-euclidean data, i.e., they directly operate on the graph structure. In order to better learn from the activity prefix sequence, TACO extends the GCN model into a Graph Recurrent Neural Network (GRNN). This approach enables TACO to capture and exploit the spatio-temporal dependencies between activities over time, offering superior predictive performance. TACO has been validated against 10 different approaches from the state-of-the-art in 10 different event logs. The results underscore TACO's consistency and superior predictive performance relative to the state-of-the-art approaches, proving the suitability of GRNNs for predictive process monitoring.

The second proposed neural model is ASTON, which builds upon TACO. ASTON is a novel approach to predictive monitoring, that tackles the activity suffix prediction problem by integrating knowledge from the process model. Drawing inspiration from the knowledge gained during the development of TACO, ASTON further improves the encoding of the process model. Furthermore, it proposes an attention mechanism that addresses the challenges posed by the suffix prediction problem. Specifically, ASTON proposes (i) a novel method for integrating the process model information into an encoder-decoder scheme, (ii) improves the encoding scheme and (iii) introduces a novel sampling strategy which is simple, effective and robust to the underlying characteristics of the event log. ASTON has been evaluated using 12 real-life event logs against 6 state-of-the-art approaches. The results show that ASTON significantly outperforms all other approaches from a statistical perspective. More importantly, ASTON exhibits robustness to the underlying characteristics of the event log, delivering consistent performance.

Finally, the last neural model we propose is DOGE, and solves one of the problems found during the development of ASTON. In particular, activity suffix prediction models are way too reliant on the sampling strategy used to select each activity of the predicted suffix. Furthermore, the performance of sampling strategy is directly tied of the underlying characteristics of the event log. Thus, even though the proposed strategy in ASTON seems robust enough, it still poses two hyperparameters that need to be tuned for each event log, and there is no guarantee that the strategy will continue to show superior performance when applied to new event logs. Therefore, in this context, the sampling strategy becomes still a new hyperparameter of the neural model, which, despite any architectural improvements made to the model, still conditions its final performance significantly. Therefore, we propose DOGE, a neural approach to activity suffix prediction based on Reinforcement Learning (RL) techniques. RL offers a framework that eases the training of a model to maximize the reward of the entire suffix sequence, rather than individual tokens, which is the case of every approach from the state of the art that is trained using Maximum Likelihood Estimation (MLE), apart from GAN approaches. In this setup, the suffix prediction problem is reduced to always selecting the most probable activity outputted by the policy network, since the learned probability distribution of the activities is conditioned on the performance of the whole suffix and not individual activities, so, the “optimal sampling strategy” is implicitly learned during training. DOGE has been evaluated using the same event logs as ASTON and the experimentation shows that the performance of DOGE is equal or superior to the one of ASTON, while being simpler to implement and

without needing the manual selection of a activity sampling strategy.

With the development of these neural models, we can consider the objectives of the Ph.D dissertation completed, resulting in three approaches that outperform the state-of-the-art proposals in control-flow prediction in predictive monitoring and a testing methodology that is being adopted by the predictive monitoring community.

Contents

Resumo	ix
Summary	xvii
1 INTRODUCTION	1
1.1 Motivation	1
1.2 Hypothesis and objectives	12
1.3 Research contributions	14
1.4 Publications	15
1.5 National conferences	16
1.6 Participation in R&D projects, contracts and networks	16
1.7 Other Contributions	18
1.8 Methodology	18
1.9 Document structure	19
2 RELATED WORK	21
2.1 Taxonomy	22
2.2 Next activity prediction	26
2.3 Activity suffix prediction	28
2.4 Conclusions	30
3 BENCHMARKING	33
3.1 Event logs	35
3.2 Preprocessing	39
3.3 Metrics	40

3.4	Statistical tests	42
3.5	Conclusions	50
4	GRAPH-BASED NEXT ACTIVITY PREDICTION	53
4.1	Graph Recurrent Neural Networks	54
4.2	Encoding	58
4.3	Architecture	61
4.4	Loop overwriting	65
4.5	Experimentation	67
4.6	Conclusions	73
5	ENCODER-DECODER GRAPH-BASED ACTIVITY SUFFIX PREDICTION	75
5.1	ASTON architecture for training	76
5.2	Encoding	80
5.3	Attention mechanism	82
5.4	ASTON architecture for inference	84
5.5	Experimentation	89
5.6	Conclusions	95
6	REINFORCEMENT LEARNING-BASED ACTIVITY SUFFIX PREDICTION	97
6.1	Reinforcement learning	99
6.2	Architecture	103
6.3	Experimentation	108
6.4	Conclusions	111
7	CONCLUSIONS	113
A	SUPPLEMENTARY MATERIAL FOR THE BENCHMARKING CHAPTER	119
A.1	Code adaptations	119
A.2	Additional next activity experimentation	121
A.3	Experimentation with 64/16/20 split	123
B	ADDITIONAL EXPERIMENTS FOR GRAPH-BASED NEXT ACTIVITY PREDICTION	129
B.1	Process model statistics	129
B.2	Loop behavior	129

B.3	Process discovery algorithm effect	131
B.4	Hyperparameter effect	133
B.5	Computation time and memory	136
C	LIST OF PUBLICATIONS, AUTHOR’S CONTRIBUTION AND COPYRIGHT INFORMATION	141
	Bibliography	145
	List of Figures	161
	List of Tables	165

CHAPTER 1

INTRODUCTION

Let the games begin!

— Nefarian, *World of Warcraft*

1.1 Motivation

Processes are ubiquitous, shaping every facet of our existence. Nearly all organizations—from manufacturing sectors [60, 12] and software engineering [24, 58] to healthcare [30, 75, 48]—employ processes to conduct their tasks and optimize their operations. Given the omnipresence nature of processes, it is crucial to have the capability to predict valuable insights about their ongoing execution. As a result, Business Process Management (BPM) emerges as a discipline offering a collection of techniques, methods, and tools for identifying, discovering, analyzing, executing, and monitoring business processes [31]. In order to ease the improvement of processes, BPM institutes a cyclic procedure, depicted in Figure 1.1, in which the process undergoes a series of steps: identifying the process relevant to the problem, discovering a model that closely mirrors actual process execution, analyzing the process using quantifiable performance measures to detect potential issues, redesigning the process based on detected problems, implementing the new process in a production environment, and monitoring the performance of the new process compared to the old one.

Traditionally, most BPM tasks require manual intervention or a high level of user interaction—for instance, when defining the process behavior or identifying process issues—which makes it difficult to handle processes with a high number of executions. To tackle these BPM

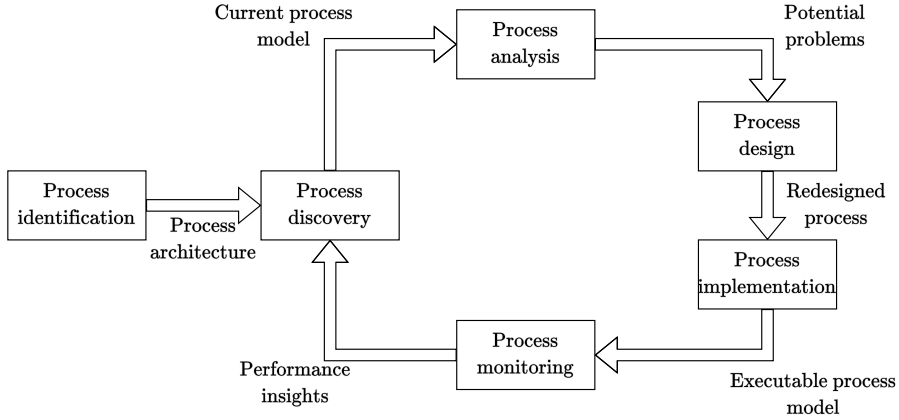


Figure 1.1: Business Process Management lifecycle —adapted from (31)—.

challenges, Process Mining (PM) emerges as a new discipline aiming to automate most BPM steps by leveraging the information recorded during the execution of the process. Consequently, PM serves as a link between the realms of Data Mining and Machine Learning, and BPM and Business Intelligence. PM, as illustrated in Figure 1.2, relies on extracting valuable insights from event logs. These logs record the process activities execution, represented as events and traces. By utilizing the actual data generated by the process, PM can assist in understanding what is genuinely occurring within the process instead of what we presume is happening. PM focuses on the analysis of individual process executions, also referred to as *cases*, which generate a sequence of activities captured in the event log. Within this framework, a *trace* is a chronologically ordered sequence of performed activities representing a single case. Each activity within a trace is denoted by an *event*, which encapsulates key data such as the activity name, timestamp, and other contextual information. The collection of these events across all cases is stored in an *event log*, which serves as the primary data source for process mining algorithms.

Definition 1: *Event*

Let AC be the universe of activities, C the universe of case identifiers, TI the time domain, and $\mathcal{D}_1 \cdots \mathcal{D}_m$ the universes of each of the attributes of the events of the event log, with $m \geq 0$. An event $e \in E$ is a tuple $(a, c, t, d_1, \dots, d_m)$ where $a \in AC$, $c \in C$, $t \in TI$,

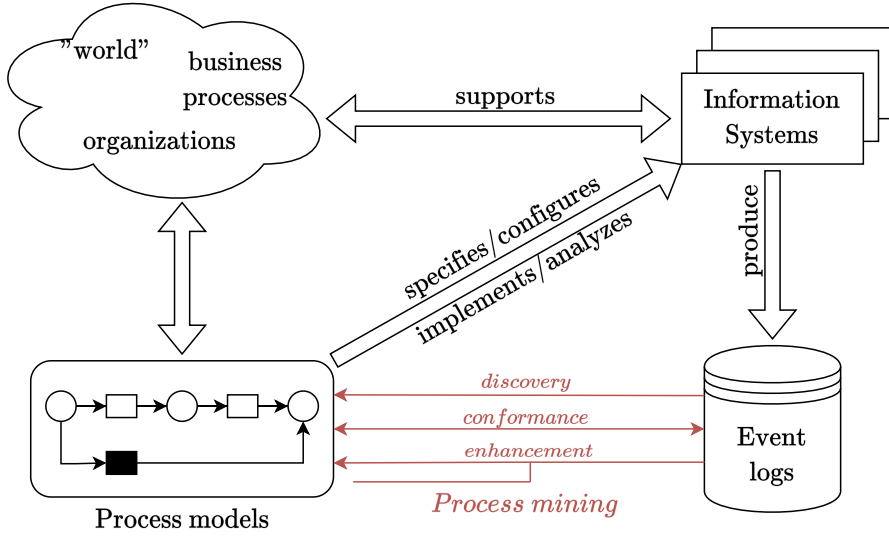


Figure 1.2: Process mining framework (adapted from (104)).

and $d_i \in \{D_i \cup \epsilon\}$, with $i \in [1, m]$ and ϵ being the empty element. The only mandatory elements of any event are the activity, a , and the case identifier, c .

Definition 2: projection function

Let $e = (a, c, t, d_1, \dots, d_m)$ be an event. Then, its projection functions in its activity, timestamp, and attributes are denoted as follows: $\pi_{AC}(e) = a$, $\pi_C(e) = c$, $\pi_{TI}(e) = t$, and $\pi_{D_i}(e) = d_i$. These functions allow us to extract the individual components that compose each event.

Definition 3: trace

Let $\mathcal{S}$ be the universe of traces. Then, a trace $\sigma \in \mathcal{S}$ is a non-empty sequence of events $\sigma = \langle e_1, \dots, e_n \rangle$, such that for all $e_i, e_j \in \sigma$, with $i, j \in [1, n]$; if $j > i$, then $\pi_C(e_i) = \pi_C(e_j)$ and $\pi_{TI}(e_j) \geq \pi_{TI}(e_i)$. Here $|\sigma| = n$. Therefore, given a sequence of events from a case, a trace is the ordered sequence of events that belong to the same case. Note that the previously defined projection functions can also be applied to the whole trace

to extract the sequence of activities, timestamps, and attributes.

Definition 4: event prefix and suffix

Let $\sigma \in S$ be a trace from the universe of traces S such that $\sigma = \langle e_1, \dots, e_n, e_{end} \rangle$, where e_{end} is an special “end-of-case token” that signals that a running case has finished and let $k \in [1, \dots, n]$. Then, the k -event prefix is defined as $hd^k(\sigma) := \langle e_1, \dots, e_k \rangle$ while $tl^k(\sigma) := \langle e_{k+1}, \dots, e_n, e_{end} \rangle$ is the k -suffix. Therefore, given any point of time in a trace, we can split it into prefix and suffix, being the shortest prefix the one that has only one event (the first one) and the shortest suffix the one that only has the end-of-case token (e_{end}).

Definition 5: log

An event log is a subset of the universe of traces, i.e, $L = \{\sigma_1, \dots, \sigma_l\} \subseteq S$, where not other trace has the same case identifier as another one in the event log, i.e, for all $\sigma_i, \sigma_j \in L$, $\pi_C(\sigma_i) \neq \pi_C(\sigma_j)$.

Table 1.1 shows an example of an event log comprising three different traces —identified as *Case1934*, *Case2032*, and *Case2100*. Each row in the table represents a distinct event; therefore, this example event log contains 15 events in total. Additionally, there are 10 different activities and one attribute associated with each event: the *resource* responsible for executing the event, which takes on eight distinct values.

Referring back to Figure 1.2, process mining is typically divided into three distinct areas: *discovery*, *conformance checking*, and *process enhancement*, as described in [104]. The first area, *discovery*, centers around constructing a process model that encapsulates the behavior recorded in the event log, a record of process events execution. Numerous discovery algorithms exist, such as the α -algorithm [106], the Inductive Miner [57], the Heuristics Miner [112], and the Split Miner [5]. These techniques output a process model, usually represented as a Petri net due to their mathematical robustness, graphical representation, and versatility. Petri nets offer an intuitive view of the process model and can be converted to other representations such as process trees [57] or causal nets [105] without any loss of information. Hence, we chose Petri nets as our preferred process model representation in this dissertation.

Table 1.1: Example of a event log with 3 different traces and 15 events. It records the exeuction of 10 different activities by 8 resources

Case ID	Activity	Timestamp	Resource
Case1934	Registration	15-02-2021 08:34:20	Jotaro
Case1934	CT scan	16-02-2021 12:35:12	Dio
Case1934	Consultation	20-03-2021 11:23:53	Josuke
Case1934	Blood analysis	21-03-2021 10:54:12	Jonathan
Case1934	Follow-up visit	05-04-2021 14:30:23	Joseph
Case2032	Registration	17-02-2021 12:57:25	Jolyne
Case2032	Blood analysis	17-02-2021 13:05:24	Dio
Case2032	Consultation	17-02-2021 14:15:48	Jonathan
Case2032	MRI scan	17-02-2021 16:33:17	Pucci
Case2032	Discharge	17-02-2021 18:46:26	Giorno
Case2100	Registration	19-02-2021 09:25:43	Jotaro
Case2100	X-ray	19-02-2021 10:17:18	Jonathan
Case2100	Consultation	19-02-2021 13:46:35	Jonathan
Case2100	Surgery	20-02-2021 08:00:00	Dio
Case2100	Discharge	25-02-2021 12:37:43	Joseph

Definition 6: Petri net

A Petri net is a tuple $PT = (\mathcal{P}, \mathcal{T}, \mathcal{F})$, comprising:

- $\mathcal{P}$: A set of places.
- $\mathcal{T}$: A finite set of transitions, differentiated into *observable transitions* —associated with a process activity— and *silent transitions* —associated with control flow routing—.
- $\mathcal{P} \cap \mathcal{T} = \emptyset$
- $\mathcal{FF} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$: A set of directed arcs.

Here, if $x \in T \cap P$ is a node of the Petri net, $\bullet xy | (y, x) \in F$ represents the set of input nodes of x , and $x\bullet = (x, y) \in F$ refers to the set of output nodes of x . The *marking* of a Petri net is a function $\mathcal{M} \rightarrow \mathbb{N}$ counting tokens in a place. A transition $t \in T$ is enabled if all its input places have at least one token ($m(p) > 0$). When an enabled transition fires, it removes a token from each input place and inserts one in every output place.

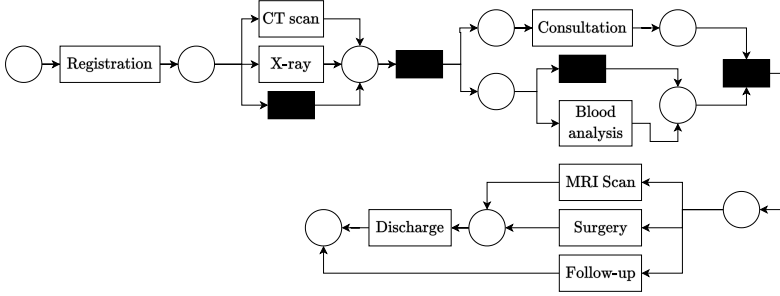


Figure 1.3: Petri net representing the behavior from the event log of Table 1.1.

Figure 1.3 presents a graphical representation of the Petri net constructed from the event log depicted in Table 1.1. In this diagram, each circle symbolizes a place, while observable transitions are represented by white squares, and hidden transitions are embodied by black squares. For the purpose of this Ph.D. dissertation, we utilize workflow nets to represent our process models.

Definition 7: *Workflow net*

A Petri net $PT = (\mathcal{P}, \mathcal{T}, \mathcal{F})$ qualifies as a workflow net if it fulfills the following conditions:

- There exists exactly one $p_{in} \in \mathcal{P}$ with $\bullet p_{in} = \emptyset$, implying that the Petri net has a unique input place.
- There exists exactly one $p_{out} \in \mathcal{P}$ with $p_{out} \bullet = \emptyset$, implying that the Petri net has a unique output place.
- Adding a transition t' to the net, with $\bullet t' = p_{out}$ and $t' \bullet = p_{in}$, results in a strongly connected graph, ensuring a path from p_{in} to p_{out} for every $t \in T$.

The second group of process mining techniques, *conformance checking*, compares an existing process model with an event log recording the same process to detect discrepancies. This analysis serves dual purposes: it can reveal inaccuracies in the process model, and it can spotlight specific incidents in the log to preempt future occurrences. Conformance checking techniques can compare the extent to which the event log behavior aligns with the process model, indicating the model comprehensiveness. It also can identify potential, yet unobserved,

process paths implied by the model but not recorded in the log, exposing potential areas for process exploration or where data capture might be inconsistent. Lastly, it can provide metrics on the complexity of the process model, illuminating its manageability and understandability.

Within the domain of conformance checking, the concept of *token replay* in a Petri net stands out as a fundamental technique. At its core, token replay is a simulation procedure that mimics the execution of the events from the log on the process model. In this simulation, each event corresponds to tokens that traverse the model, following its defined pathways. The movement and accumulation of these tokens within the structure of the Petri net can reveal discrepancies between the event log and the process model.

Definition 8: *Token Replay*

Let $N = (\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{M}_i)$ be a Petri net, and let $\pi_A(\text{hd}^k(\sigma))$ the activity prefix of trace σ . The token replay of $\pi_A(\text{hd}^k(\sigma))$ on N is the sequence of transitions needed to be fired to reproduce the behavior of the activity prefix, starting from the initial marking $\mathcal{M}_i$. For each transition t , the token replay procedure: (i) checks if there are enough tokens in the input places of t according to $\mathcal{F}$; (ii) if so, remove these tokens and adds tokens to the output places of t , according to $\mathcal{F}$ (this is known as firing the transition); and (iii) if not, fills the needed input tokens with exactly one token and performs the transition firing as usual.

The third and final category in the taxonomy of process mining techniques, known as *process enhancement*, focuses on augmenting the existing process model and/or the event log with insightful information to facilitate optimization and improvement of the business process. This category incorporates a broad array of techniques, each catering to unique requirements. For instance, *model repair* techniques [1, 4, 79] aim to modify the process model to more accurately reflect the behavior documented in the event log, thereby enabling closer alignment between the model and operational reality. In contrast, *extension* techniques [93, 64, 80] aim to augment the process model by assimilating additional data from the event log, such as timestamps and resources, providing a more detailed view of process operations. It is worth noting that techniques such as *predictive monitoring* —the main focus of this dissertation and further detailed in Section 1.1.1— along with *activity prescription* and *process simulation*, also fall under this category due to their contribution of additional insights into the process. Utilizing these techniques, one could, for example, predict execution times with greater precision, offering vital foresight into potential timeframes and thereby aiding in

process efficiency. Techniques such as bottleneck prediction [89, 78] could help identify areas of process flow congestion to improve overall throughput, while anomaly detection [83, 28] could flag unusual process behaviors, potentially highlighting errors or inefficiencies for remediation.

According to all this information, event logs can be analysed from different perspectives [104], from which the most common ones are the following:

- *Control-flow perspective*: this perspective focuses on the specific order of execution of the activities within the process. Therefore, the analysis made in this perspective are targeted to reveal the constraints and dependencies between the activities of the process. This perspective is one of the most common, and, often, is the entry point for any process mining analysis. For example, one could use the control-flow perspective to determine whether the process is being followed as intended, or leverage predictive monitoring techniques to forecast upcoming activities, thus providing early insights into potential deviations or efficiencies in the control flow.
- *Time perspective*: this perspective focuses on time-related analysis about the different aspects of the process. Normally, these analysis focus on the time spent in the execution of the different activities, how much time is wasted waiting for the execution of the next activity, the time frequency of the activities, etc. Furthermore, techniques that focus on detecting bottlenecks or predicting the remaining time until a given running case finishes also fall here. For example, one could use the time perspective to identify activities that are taking too long to complete, or to determine how long it typically takes to complete the process from start to finish.
- *Case perspective*: this perspective focuses on analyzing interdependencies between the different attributes of the process instances. Therefore, knowing what does have in common multiple instances that deviate from the expected behavior can help predict when a new instance will deviate from the process model. For example, one could use the case perspective to identify common causes of delays in the process, or to determine which cases are most likely to be completed successfully according to the aggregated information from all cases from the event log.
- *Resource perspective*: this perspective focuses on analyzing information about resources available in an event log. This analysis mainly focuses on analyzing the structure of an

organization by constructing social network graphs and detecting resource constraints of a process with respect to executed activities. Note that resources can be either human or non-human; machines that execute activities of a process fall into this latter category.

There are more perspectives from which process mining can be applied, but the aforementioned list covers the most common cases for process mining analysis. Furthermore, combining multiple perspectives at the same time can enhance the analysis and help obtain more accurate insights about the process. In this Ph.D. dissertation we focus on analyzing the control-flow perspective by taking as inputs the control-flow, time, and resource perspectives at the same time.

1.1.1 Predictive monitoring

Traditional process mining analyses are largely focused on completed traces, providing *reactive* insights based on a *post-mortem* analysis. These reactive approaches only highlight process issues post-occurrence, leading to limitations in their preventative potential. However, there is a growing trend towards applying process mining techniques to ongoing process instances [65]. Such proactive approaches allow for the anticipation and prevention of potential problems before they occur. For example, resources could be allocated more efficiently before they are wasted, minimizing potential deviations before they occur. Recalling the example from the event log in Table 1.1, we could predict the remaining time until a patient is discharged, what will be the next medical trial applied to a patient or forecast whether the patient is going to be under surgery or not. *Predictive monitoring* embodies this proactive approach, by utilizing event log information to anticipate the evolution of an ongoing process case.

Predictive monitoring can address a variety of tasks, such as next activity prediction, activity suffix prediction, next timestamp prediction, remaining time prediction, and outcome predictions. Specifically, *next activity prediction* aims to forecast the immediate subsequent activity in a process instance; *activity suffix prediction* focuses on predicting the sequence of activities that will follow the current point in the process; *next timestamp prediction* estimates the time when the next activity will occur; *remaining time prediction* calculates the expected time left until a case is completed; and *outcome prediction* projects the likely end-state or result of a process instance. This Ph.D. dissertation will specifically focus on the control-flow perspective, thereby emphasizing next activity and activity suffix prediction tasks.

Definition 9: Next Activity Prediction

Let $hd^k(\sigma) = \langle e_1, \dots, e_k \rangle$ represent a prefix of a trace $\sigma \in \mathcal{S}$ with length k from the universe of traces $\mathcal{S}$, and let $hd^1(tl^k(\sigma)) = \langle \pi_{AC}(e_{k+1}) \rangle$ be the first event of the suffix of σ after the k -th event. The next activity prediction problem aims to discover a function Ω_{AC} such that $\Omega_{AC}(hd^k(\sigma)) = \pi_{AC}(e_{k+1})$ for all $\sigma \in \mathcal{S}$ and all valid k ranges, where π_{AC} is the projection function for activities.

Definition 10: Activity Suffix Prediction

Let $hd^k(\sigma) = \langle e_1, \dots, e_k \rangle$ be a prefix of length k of a trace $\sigma \in \mathcal{S}$ from the universe of traces $\mathcal{S}$. The aim of the activity suffix prediction problem is to find a function Ω_{AS} such that $\Omega_{AS}(hd^k(\sigma)) = \pi_{AC}(tl^k(\sigma))$ holds for all $\sigma \in \mathcal{S}$ and for all valid k ranges.

These two problems are closely related: the suffix prediction problem can be approached by iteratively predicting the next activity until the end of the trace is identified. This iterative approach can be formally defined as follows:

Definition 11: Iterative Activity Suffix Prediction

The function Ω_{AS} can be constructed by iteratively predicting the next activity using Ω_{AC} following the recursive function. Let $\oplus$ be the concatenation operator, and e_{end} be the end-of-case token. Then Ω_{AS} can be defined as follows:

- $\Omega_{AS}(hd^k(\sigma)) = \sigma$ if $\Omega_{AC}(hd^k(\sigma)) = e_{end}$.
- $\Omega_{AS}(hd^{k+1}(\sigma \oplus e)) = \Omega_{AS}(hd^k(\sigma)) \oplus a$, with $\pi_A(a) = \Omega_{AC}(hd^k(\sigma))$, otherwise.

In predictive monitoring, event traces are usually divided into all possible prefix-suffix pairs. The machine learning model is then trained using the prefixes, with the target variable constructed from information derived from both the prefixes and suffixes, as required. This procedure enables the prediction of the future evolution of the process instance from the knowledge of its past and current state. Figure 1.4 illustrates how this procedure is applied to the trace *Case2032* from the event log represented in Table 1.1, with the aim of predicting the next activity. In this example, we use only control-flow information as inputs to the machine

learning algorithm. Here, X represents the feature set for the machine learning model, while Y represents the target variable. Each row in the X matrix corresponds to a different prefix of the trace, and the corresponding row in the Y vector denotes the next activity to be predicted for that prefix. Note that the end-of-case token is added as a target variable so that the model is able to predict when the case has finished.

Classically, predictive monitoring primarily employed traditional machine learning techniques, such as Decision Trees [35], Factorization Machines [55], Probabilistic Finite Automats [8, 14], or Support Vector Machines [50, 19]. However, two factors have recently started shifting this trend: the growing volume of data generated from business processes, which requires more sophisticated models to leverage this information, and the huge traction that deep neural networks have gained in dealing with unstructured data in areas like Computer Vision [43, 87] and Natural Language Processing [108, 95].

In this context, Recurrent Neural Networks (RNNs) have emerged as a fitting solution to the predictive monitoring problem [33, 99, 85, 22, 59, 45, 49, 36, 100, 94], mainly due to their inherent ability to handle sequential data—a characteristic common to business processes—, their capacity to work with the growing amount of data recorded in business processes, and their capability to automatically extract relevant features from input data, diminishing the amount of manual feature engineering that has to be done to obtain a good performance. RNNs, and in general deep learning techniques, have been found to deliver comparable or better performance than other machine learning methods [98], solidifying them as the current go-to solution in predictive monitoring. However, it is worth mentioning that other deep learning approaches to predictive monitoring are also being explored, including autoencoders [69, 70], Generative Adversarial Networks (GANs) [101, 100], and Convolutional Neural Networks [68, 73]. Due to the prominence of deep-learning based approaches for predictive monitoring, we will focus exclusively on these kind of models for the evaluation of our proposed approaches. Nonetheless, it is important to note that, despite the advancements and promising results, when a benchmarking is conducted following a well-established methodology there remains considerable room for improvement in achieving more accurate and robust predictive monitoring outcomes.

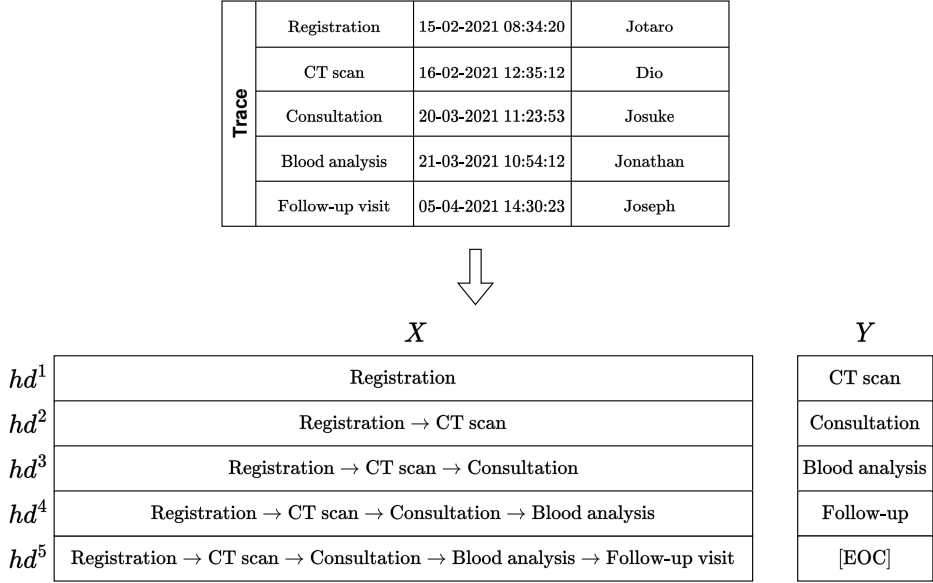


Figure 1.4: Example of prefixes and target variables for the trace *Case2032* in the event log depicted in 1.1. (EOC) represents the end-of-case token.

1.2 Hypothesis and objectives

Typically, deep-learning-based predictive monitoring techniques fit predictive models only using information directly available from the event log, such as sequences of activities, time features, and event attributes. However, these methods often overlook the structural information available in process models. Behavioral patterns such as loops and parallelisms may be inadequately captured, as the neural networks might struggle to detect these patterns solely from trace information. The incorporation of explicit information from the process models could simplify the training of the neural networks. Not only does this provide the networks with direct access to structural information, but it also offers an understanding of potential execution paths that an ongoing process instance can take, which is particularly useful for the activity suffix prediction task. Consequently, this should enhance convergence by learning the interdependencies between the control-flow and the other attributes of the case. In an ideal scenario, the amount of manual feature engineering in the extraction phase of the process model features should be minimized as much as possible. This would allow us to offload the

feature extraction process to the neural network, potentially resulting in improved performance. Based on this rationale, the hypothesis for this Ph.D. research can be formulated as follows:

Incorporating explicit structural information from the associated process model into a deep learning-based predictive monitoring approach can significantly enhance its performance by enabling a better understanding of the business process dynamics.

Considering this hypothesis, the primary aim of this Ph.D. dissertation is to develop an approach that can effectively learn the structural properties of the process model without the need for manual feature extraction. The devised solution should consider how the interactions of the prefix with the process model occur without losing information during the encoding of this interactions, and it should consider the full interaction from the beginning of the prefix until the end. This main objective can be divided into three key sub-objectives:

O1. Improve the impartial comparison of predictive monitoring approaches by developing a comprehensive methodology and benchmark. The first objective, thus, involves designing and implementing a method that allows fair comparison of predictive monitoring techniques. As a reference the most important deep learning approaches in predictive monitoring should be evaluated.

O2. Improve the performance of next activity prediction by designing an algorithm that integrates the process model as an essential input to the neural network. The problem of next activity prediction, being simpler than activity suffix prediction, provides an ideal starting point for assessing whether the proposed approach can yield more information in an automated manner.

O3. Enhance the performance of next activity suffix prediction by developing an algorithm that integrates the process model into the inputs of the neural network. Upon successful implementation of the next activity prediction algorithm, the next step involves adapting it for the next activity suffix prediction, allowing for further refinement the unique aspects of the activity suffix prediction problem. This algorithm should incorporate an inference phase designed to select the best candidate activities for the suffix prediction, leveraging both the predictions from the neural network and the existing process model information for more accurate and context-aware predictions.

O4. Improve the efficiency and performance of next activity suffix prediction by creating an adaptive algorithm that automatically learns the optimal activity sampling strategy. Upon establishing the framework for the next activity suffix prediction, the objective shifts towards devising an algorithm capable of independently discerning the most effective sampling strategy for a given event log. As the choice of the sampling strategy significantly influences the performance of the algorithm, an adaptive selection mechanism would bolster the robustness of the approach across varying event logs, alleviating the necessity for manual intervention in strategy tuning.

1.3 Research contributions

The research developed in this Ph.D. dissertation led to the following contributions:

- **A methodology for the fair and robust comparison of multiple predictive monitoring approaches applicable to any predictive task.** This methodology not only ranks the approaches based on performance but also quantifies the extent of improvement offered by each approach, verifying if such improvement is statistically significant.
- **TACO: A neural network model for next activity prediction that incorporates the process model.** It automatically extracts features from both the process model structure and its interactions with the activity prefix, learning behavioral patterns without manual feature extraction. TACO main contributions include: (i) a neural model that learns spatio-temporal information from both the event log and the process model; and (ii) a process model encoding scheme that enables the learning of structural information from the process model.
- **ASTON: A neural model for next activity suffix prediction that integrates the process model.** It enhances TACO encoding scheme and considers the integration of the model information within the suffix prediction problem. ASTON contributions include: (i) an encoder-decoder neural model that predicts the next activity suffix by decoupling the learning representation phase from the prediction phase, (ii) an improved encoding scheme over TACO, facilitating a simpler learning phase for the neural model; and (iii) a refined heuristic search strategy that selects the most likely suffix rather than iteratively predicting the next activity, striking an optimal balance between performance and computational efficiency.

- **DOGE: A neural model for next activity suffix prediction that automatically learns the best sampling strategy.** Building on the insights gained from developing ASTON, we observed that the sampling strategy used during inference significantly impacts the neural model performance. Hence, we propose a new proof-of-concept approach, DOGE, that employs deep reinforcement learning to autonomously learn the optimal sampling strategy for activity suffix prediction. The key contributions of DOGE include: (i) a novel reinforcement learning-based neural model, designed in an encoder-decoder fashion, which predicts activity suffixes; (ii) a well-aligned reward mechanism with the suffix prediction task that facilitates more accurate learning of the interdependencies among the predicted activities within the suffix; and (iii) enhanced efficiency in speed: unlike methods reliant on search procedures during inference (like beam search), DOGE deep reinforcement learning approach negates the need for any such procedures, resulting in markedly enhanced inference speed.

1.4 Publications

All the contributions shown in this dissertation are included in the following publications:

Journals

- Efrén Rama-Maneiro, Juan Vidal and Manuel Lama. **Deep Learning for Predictive Business Process Monitoring: Review and Benchmark.** *IEEE Transactions on Services Computing*, 16:739-756, doi: 10.1109/TSC.2021.3139807.

Journal Impact Factor (JCR 2022): 8.1

13/158 (Q1) COMPUTER SCIENCE, INFORMATION SYSTEMS

4/108 (Q1) COMPUTER SCIENCE, SOFTWARE ENGINEERING

- Efrén Rama-Maneiro, Juan Vidal and Manuel Lama. **Embedding Graph Convolutional Networks in Recurrent Neural Networks for Predictive Monitoring.** *IEEE Transactions on Knowledge and Data Engineering*, doi: 10.1109/TKDE.2023.3286017.

Journal Impact Factor (JCR 2022): 8.9

18/145 (Q1) COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE

9/158 (Q1) COMPUTER SCIENCE, INFORMATION SYSTEMS

- Efrén Rama-Maneiro, Pablo Monteagudo-Lago, Juan Vidal and Manuel Lama. **Exploiting Recurrent Graph Neural Networks for Suffix Prediction in Predictive Monitoring**. [abs/2211.16106](https://arxiv.org/abs/2211.16106).

1.5 National conferences

- Efrén Rama-Maneiro, Juan Vidal and Manuel Lama. **Deep Learning for Predictive Business Process Monitoring: Review and Benchmark (Summary)**. *Jornadas de Ciencia e Ingeniería de Servicios (JCIS)*, 2022. hdl.handle.net/11705/JCIS/2022/013.
- Efrén Rama-Maneiro, Juan Vidal and Manuel Lama. **Hacia una Mejor Monitorización Predictiva: Combinando Redes Neuronales de Grafos con Redes Neuronales Recurrentes en Minería de Procesos**. *Jornadas de Ciencia e Ingeniería de Servicios (JCIS)*, 2023. hdl.handle.net/11705/JCIS/2023/6402. Best paper award.

1.6 Participation in R&D projects, contracts and networks

Apart from the aforementioned publications, the development of this Ph.D. thesis has led to the following collaborations in R&D national projects:

- **BIGBISC - Fueling Intelligence to Business Processes with Soft Computing in Big Data Scenarios**. The main goal of BIGBISC is to develop an extensive collection of algorithms for process mining, relying heavily on soft computing techniques such as fuzzy logic, neural networks, and evolutionary algorithms. These algorithms have been integrated into a cloud-based architecture coined as “Big-mining-as-a-service”. This infrastructure serves as a backbone for delivering business intelligence solutions tailored for Big Data environments. A key milestone of this project was the rigorous validation of the developed algorithms across multiple real-world datasets, which span different industries and application domains. For instance, within the scope of this project, predictive monitoring approaches were tested on real-world logs related to

the management of patients suffering from valvular heart diseases. These logs were sourced from the cardiology service of the University Hospital Complex of Santiago de Compostela (CHUS), proving the practical utility and adaptability of the algorithms developed in this Ph.D dissertation.

- **RAI4P - Responsible AI for Process Mining 2.0.** RAI4P aims to advance towards Process Mining 2.0 by incorporating principles of responsible AI. It focuses on predictive monitoring of business processes, intending to design models capable of forecasting various key aspects such as upcoming activities, time remaining for case completion, necessary resources, and likely outcomes. Unlike traditional models, the predictions generated could either target specific moments or cover the entire lifespan of a process, offering both short-term and long-term strategic insights. This is particularly crucial in domains requiring ethical and responsible decision-making. The algorithms that are part of this Ph.D. dissertation were conceived within the context of RAI4P to account for details such as inferred trace information and the underlying process model.
- **INSIDE - Emotion-aware Process Mining for Improving the Adherence in Cardiac Rehabilitation.** INSIDE sets out to integrate process mining techniques with emotion recognition via wearable devices, aiming to generate deep insights into patient adherence levels in Cardiac Rehabilitation processes. The key innovation here is the fusion of traditional process mining metrics with emotional data, leading to multi-dimensional analyses that can predict patient behaviors such as dropouts. The algorithms designed as part of this Ph.D. research play a crucial role in forecasting dropout rates, which can, in turn, assist healthcare providers in taking preemptive measures to improve patient adherence.

Furthermore, the results of this Ph.D. thesis have also been applied to the following R&D contract:

- **XIA - Analysis and experimentation of the application of intelligent technologies to the administrative procedures of the Xunta de Galicia.** XIA is designed to enhance the way citizens interact with the Xunta de Galicia through the incorporation of cutting-edge technologies like artificial intelligence, machine learning, and process mining. The project aims to offer a highly personalized, efficient, and user-friendly experience for citizens who engage with governmental processes. Algorithms developed in this

Ph.D. dissertation contribute significantly by forecasting various key metrics. For example, they can predict the likelihood of a user failing in the submission process of an application, hence facilitating early interventions to improve user experience and process efficiency.

1.7 Other Contributions

In the spirit of bolstering collaborative research and furthering the state of the art in the field, we have made all algorithms developed during the course of this dissertation openly available for academic purposes. The source code provides researchers with a comprehensive toolkit to replicate, extend, or integrate these algorithms into their own work.

- *Benchmark Framework*: aimed at providing a standardized evaluation procedure, this repository offers a robust benchmarking framework tailored for the process mining community¹.
- *TACO*², *ASTON*³ and *DOGE*⁴: the code for the approaches developed in this Ph.D dissertation is available to the scientific community for academic purposes.

1.8 Methodology

The research delineated in this dissertation has been pursued employing the scientific method. It involved an iterative procedure where the state-of-the-art techniques were studied, a solution was developed and validated, and modifications were made to the algorithms based on validation results. It is important to note the flexibility of this process as one phase might initiate before another concludes, they may overlap, and revisions can be made to previous steps to enhance the performance of the approach. For instance, the theoretical framework of the solution started its development before the state-of-the-art analysis had finalized. Consequently, the following steps were applied:

- *Definition of the objective*. This is the initial phase where the problem is clearly identified and research goals are established. This identification phase includes thoroughly

¹<https://gitlab.citius.usc.es/efren.rama/pmdlcomparator>

²<https://gitlab.citius.usc.es/efren.rama/neuralpetrinet>

³<https://gitlab.citius.usc.es/efren.rama/aston>

⁴<https://gitlab.citius.usc.es/efren.rama/rl-ppm>

describing the problem at hand, the requirements for the validation, and anticipating the limitations of the future solution.

- *Analysis of the state of the art.* Here, a comprehensive review of current methods and approaches relevant to the research objectives is conducted. This study helps identify existing gaps in the field and provides a foundation for the development of new solutions.
- *Development of the solution.* Based on the insights obtained from the state-of-the-art analysis, a unique solution is formulated to address the identified problem. This stage involves theoretical framework development, algorithm design, and system implementation. An incremental approach is employed for software development, iterating between the implementation of a feature and its testing.
- *Validation of the solution.* This final step involves rigorous testing and validation of the developed solution to ensure its effectiveness and reliability. The performance of the solution is quantified and compared to existing methods. If the performance of the solution does not meet the expectations, this process reverts to previous steps so that the solution can be modified accordingly.

1.9 Document structure

The remainder of this Ph.D dissertation is as follows:

- Chapter 2 provides an extensive review of the current state of the art in deep learning based predictive monitoring approaches. Given the high number of existing approaches and their diverse objectives, a taxonomy is proposed to systematically classify these deep learning approaches across various dimensions.
- Chapter 3 presents the methodology used to evaluate the state-of-the-art approaches in a fair way. We gathered the publicly available approaches from the previous chapter and we performed a rigorous experimentation with all of them. The result interpretation is aided by the usage of statistical tests, that define an overall ranking of the approaches and quantify the actual differences between them. This methodology supports also all the experimental evaluation carried out in this Ph.D dissertation.

- Chapter 4 elaborates on TACO, a novel predictive monitoring approach for next activity prediction. TACO leverages the structural information of the process model and automatically extracts features from both the model and its interaction with the activity prefix using graph recurrent neural networks. Its effectiveness is validated using the benchmarking results and methodology outlined in Chapter 3.
- Chapter 5 introduces ASTON, another predictive monitoring approach designed for next activity suffix prediction. ASTON builds on TACO learned lessons, improving its encoding scheme and adapting it to the suffix prediction problem. Validation for ASTON is also conducted using the benchmarking results and methodology from Chapter 3.
- Chapter 6 presents DOGE, a predictive monitoring model specifically devised for the task of next activity suffix prediction. DOGE advances upon the foundational work of ASTON, recognizing that one of the most important factors in suffix prediction lies in the choice of the sampling strategy during inference. DOGE aspires to automate the selection of this optimal sampling strategy by leveraging deep reinforcement learning techniques. The efficacy of DOGE is evaluated using the established benchmarking results and methodology from Chapter 3.
- Chapter 7 presents the final conclusions of the dissertation, describing the most important findings obtained from the research. Additionally, potential avenues for future research, building upon the groundwork laid in this dissertation, are discussed.

CHAPTER 2

RELATED WORK

*Never perfect. Perfection goal that changes.
Never stops moving. Can chase, cannot catch.*
— Abathur, *StarCraft II*

As introduced in Chapter 1, forecasting how an ongoing business process case will unfold can provide important insights for proactive problem prevention. Nowadays, deep learning techniques dominate the predictive monitoring landscape due to their advantages with respect to traditional machine learning approaches: they benefit from the huge amount of data that business processes are generating nowadays, their ability of handling sequential data—intrinsic of business processes—and their ability to automatically extract features from their input data, without the need of much manual engineering.

The past five years have witnessed a surge in deep learning based predictive monitoring approaches. These approaches differ in terms of event encoding, input data usage, predictive model neural architecture, tackled predictive tasks, and prefix encoding. To facilitate the understanding of the state-of-the-art approaches, we propose a taxonomy to easily discern the differences among them. Then, we focus our discussion on approaches that aim at forecasting the control-flow aspects of the business process, as this is the objective of this Ph.D dissertation.

The contents of this chapter are extracted from the following publication:

Efrén Rama-Maneiro, Juan Vidal and Manuel Lama. **Deep Learning for Predictive Business Process Monitoring: Review and Benchmark.** *IEEE Transactions on Services Computing*, 16:739-756 (Date of Publication: 31 December 2021) ISSN:1939-1374, doi: 10.1109/TSC.2021.3139807.
Author's contributions:

Efrén Rama-Maneiro: Methodology, Software, Validation, Formal Analysis, Investigation, Data Curation, Writing — Original Draft.

Juan Vidal: Conceptualization, Resources, Writing — Review & Editing, Visualization, Supervision, Project administration, Funding acquisition, Investigation.

Manuel Lama: Conceptualization, Resources, Writing — Review & Editing, Visualization, Supervision, Project administration, Funding acquisition, Investigation.

2.1 Taxonomy

A deep-learning-based predictive monitoring approach can be classified across five key dimensions according to how the neural network is configured: (1) *Input data*: the subset of data from the event logs used for training the predictive model; (2) *Predictions*: the specific elements of the events that the neural network is trained to predict; (3) *Neural Network Type*: the type of neural network architecture employed for building the predictive model, such as feedforward, autoencoder, convolutional, recurrent, etc; (4) *Sequence encoding*: the methodology used to convert event prefixes into learnable tensors for the neural network that are fed in minibatches; (5) *Event encoding*: the strategy employed to encode individual categorical and continuous variables from the events into feature vectors.

The specific subcategories for the aforementioned categories are summarized in the morphological box represented in Table 2.1. In the stated taxonomy, the subcategories for *Network Type* and *Input data* are self-explanatory as they refer to well-known deep learning models and typical information available in any event log, respectively. The subcategory of *Time features* is also self-explanatory, generally referring to the temporal information of the event logs that may influence the model predictive performance. Regarding the *Sequence encoding* category, the following subcategories can be distinguished:

- *Continuous*: in this encoding technique, the whole log is viewed as a single text, each trace as a sentence of that text, and each activity as a word of a sentence. Here, the

Network Type	Long Short Term Memory (LSTM) Autoencoder (AE) Deep Feedforward Network (DFNN) Differentiable Neural Computer (DNC) Gated Recurrent Unit (GRU) Bidirectional LSTM (B-LSTM) Transformer (TRANS) Key-Value-Predict (KVP) Generative Adversarial Networks (GAN) Gated Convolutional Neural Networks (GCNN) Graph Convolutional Networks (GCN) Large Language Models (LLM)
Sequence Encoding	Continuous (CONT) Prefixes Padded (PRFX) N-gram (N-GRAM) Single Event (SE) Timed State Sample (TSS) Last State of the Prefix (LS)
Event Encoding	Embedding (EMB) One-Hot Encoding (OH) Pretrained Embedding (P-EMB) Frequency-Based (FB) Graph-Based Encoding (GRPH)
Input Data	Activity (ACT) Linear Temporal Logic (LTL) Attributes (ATTR) Clustered Attributes (C-ATTR) Process Model (PM) Role (R) Directly Follows Graph (DFG)
Time Features	Time Since Start of the Case (TSSC) Time Since Previous Event (TSP) Day of the Week (WK) Time Since Midnight (TMD) Decayed Time Since Previous Place Activation (DTPA) Duration of the Event (DUR) Date of the Timestamp: Hour, Day, Month, Year (DATE)

Table 2.1: Morphological box of every possibility for each characteristic.

minibatches used to train the network are constructed with windows of events of size W , and each window can include information from multiple different traces.

- *Prefixes padded*: in this encoding scheme considers every possible set of event prefixes hd^k . During training, minibatches are constructed by sampling from this set of prefixes. In this encoding type, each timestep only considers the W most recent events. If the

length of a prefix is less than W , it is padded with special *padded tokens*, typically identified by the value 0. These tokens serve as placeholders, ensuring a uniform input size for the neural network. They should not be confused with end-of-case tokens, which signify the conclusion of a sequence. Conversely, if a prefix exceeds W in length, it is truncated. It should be noted that W could be the maximum trace length of the event log. Padding tokens are especially useful for models that require fixed-length input sequences and should not be confused with end-of-case tokens, which typically signify the conclusion of a sequence.

- *N-gram*: this encoding represents each trace as a set of all subsequences up to length k . Due to the potential for a high dimensional feature space, the *hashing trick* [113] is employed to manage dimensionality. The hashing trick involves applying a hash function to each N-gram and using the resulting hash value as an index in a fixed-length feature vector. This allows for a constant and manageable feature vector size, albeit with the risk of hash collisions where different N-grams could map to the same index.
- *Single event*: in this encoding, only the last event and its attributes are considered, disregarding the sequence of events in the trace.
- *Timed state*: this encoding represents the inner state of a process model, represented as a Petri net, after replaying a prefix in it. Each place in the Petri net is enhanced with a “decay function” that counts the time between the current timestamp and the last time a token was in a given place. Therefore, the prefix is compressed into a single manually engineered feature vector that summarizes the state of the Petri net after the replay process.
- *Last state of the prefix*: this encoding summarizes all the information from the prefix into a singular vector. In this scheme, the explicit hard ordering of the sequence is either disregarded or implicitly inferred from the encoding structure, providing a summarized view of the prefix up until the current state.

On the other hand, regarding the *Event encoding* category, the following subcategories can be distinguished:

- *One-hot*: the categorical feature is represented as a binary vector, the size of which corresponds to the number of possible distinct values for that variable. A number “one” is placed in the position in the vector that corresponds to the category of the variable.

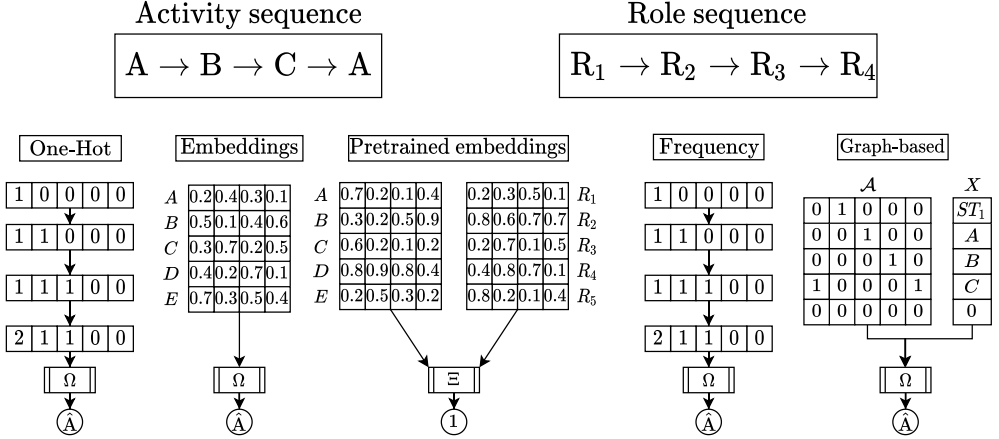


Figure 2.1: Different event encoding schemes applied in predictive monitoring.

- *Embedding*: the embedding encoding creates a learnable parameter matrix where each row corresponds to each of the categories of the encoded variable.
- *Frequency-based*: this type of encoding indicates how many times each activity has occurred up to the current event in the prefix. This encoding is useful when temporal information must be added to the encoding of the activities.
- *Pretrained embedding*: instead of directly training the embedding vectors as in the *embedding* approach, the embeddings are pretrained using a related task—for example, distinguishing between existing or non-existing pairs of resource-activity—.
- *Graph-based encoding*: in this approach, the features are arranged in a matrix that mirrors the topological structure of a process model. For instance, if the process model is a Petri net, the number of rows of the encoded tensor might depend on the number of places in the Petri net. This type of encoding captures the structural context inherent to the process model.

Figure 2.1 shows the various event encoding schemes applied to a sample sequence of activities and roles. Here, Ω denotes the predictive model for next activity prediction, and Ξ signifies a surrogate neural model for a different task. In the *One-hot* and *Frequency-based*

encoding schemes, the full encoded sequence is displayed, with the nuance that in *Frequency-based* encoding only the encoding after the last processed event is actually utilized. For the *Embedding* and *Pretrained Embedding*, the complete embedding matrices are shown. The main distinction between these two is that in the *Embedding* jointly optimized with the main predictive task Ω , while in the *Pretrained Embedding*, it is optimized for a different task such as discerning between positive and negative pairs of Activity-Role. Finally, the *Graph-based encoding* capitalizes on the inherent structure of the underlying process model, represented by the adjacency matrix $\mathcal{A}$ and the feature matrix X . Here ST_n denotes a silent transition in the context of a Petri net, while A , B , and C are the identifiers of the activities. Note that, in this latter case, the feature matrix may require additional processing using another technique mentioned here, such as *Embeddings*. Even though these are the most common ways to encode events in predictive monitoring, newer ways to encode the events are currently being researched [37].

2.2 Next activity prediction

Table 2.2 shows the works from the state-of-the-art that focus on next activity prediction. The earliest works in predictive monitoring with deep learning relied on Recurrent Neural Networks (RNNs) to develop predictive models. [99] used Long Short-Term Memory (LSTM) neural networks to predict both the next activity and the next timestamp in an ongoing process instance. The activities were encoded as a one-hot vector, and time features, such as the time elapsed since the previous event or since the beginning of the case, were also considered. [33] also uses an LSTM to predict the next activity, encoding the activity through embeddings instead of an one-hot vector, thereby reducing the dimensionality of the input.

Other types of architectures rooted in recurrent neural networks have been studied as well. [52] utilize a Differentiable Neural Computer [41], a type of neural network that incorporates an external memory to enhance the representation of long-term dependencies within a sequence. They developed an encoder-decoder model trained to predict either the upcoming activity or the subsequent timestamp. [49] also employ an encoder-decoder network, but their approach is enhanced by an attention mechanism that takes into account every hidden state of the LSTM learning from the input sequence. While these sophisticated architectures can potentially produce improved results, they also introduce increased network complexity and potential challenges during the training process, compared to simpler types of RNNs. In [59], an

Author, Year	Reference	Network type	Sequence encoding	Event encoding	Input data	Time features
Evermann et al., 2017	[33]	LSTM	CONT	EMB	ACT	-
Mehdiyev et al., 2017	[69]	AE + DFNN	NGRAM	-	ACT, ATTR	-
Tax et al., 2017	[99]	LSTM	PRFX	OH	ACT	TSP, TSSC, WK, TMD
Al-Jebrni et al., 2018	[2]	CNN	CONT	EMB	ACT	-
Khan et al., 2018	[52]	DNC	PRFX	OH	ACT	TSP, TSSC, WK, TMD
Schönig et al. 2018	[85]	LSTM	CONT	OH	ACT, ATTR	-
Mehdiyev et al., 2018	[70]	AE + LR	NGRAM	-	ACT, ATTR	-
Camargo et al., 2019	[22]	LSTM	PRFX	P-EMB	ACT, R	TSP
Lin et al., 2019	[59]	LSTM	CONT	EMB	ACT, ATTR	-
Hinkka et al., 2019	[45]	GRU	PRFX	OH	ACT, ATTR, C-ATTR	-
Theis et al., 2019	[102]	DFNN	TSS	FB	ACT, PM, ATTR	DTPA
Pasqua. et al., 2019	[73]	CNN	PRFX	FB	ACT	TSSC
Mauro et al., 2019	[68]	CNN	PRFX	EMB	ACT	TSP
Philipp et al., 2020	[74]	TRANS	CONT	EMB	ACT	-
Jalayer et al., 2020	[49]	LSTM	PRFX	EMB	ACT	-
Taymouri et al. 2020	[101]	GAN	PRFX	OH	ACT	TSP
Heinrich et al., 2021	[44]	(LSTM) KVP & GCNN	PRFX	ACT	-	
Buksh et al., 2021	[18]	TRANS	PRFX	EMB	ACT	-
Weinzierl et al., 2021	[114]	GGNN	LS	OH	ACT, DFG	-
Venugopal et al., 2021	[109]	GCN	LS	GRPH	ACT, DFG	TSSC, TSP, WK, TMD

Table 2.2: Collected studies based on deep learning for next activity prediction.

encoder-decoder scheme is enhanced with a “modulator”. This component integrates the representations of the activity and its attributes, weighing the significance of each attribute to provide a combined representation.

Certain predictive monitoring methodologies propose novel ways to encode the attributes of the event log. For example, in [85] the authors investigate how can discrete and continuous event features can be used in RNNs. In [22] the resources available in the event log are clustered into roles, then train an LSTM to simultaneously predict the next activity, timestamp, and role. [45] adopts a similar approach, clustering the events based on their attributes and employing these clustering labels as additional input information in an RNN. These types of encodings primarily aim to reduce the complexity of the input by categorizing attributes into fixed groups, at the risk of neglecting relevant information.

Several approaches also employ Convolutional Neural Networks (CNNs). [73] uses CNNs to predict the next activity by rearranging the prefixes in a grid-like fashion. For each event, the authors count the number of occurrences present in the prefix, then feed these prefixes to

a two-dimensional CNN for predicting the next activity. Similarly, [68] adapts the Inception model [97] to forecast the next activity, outperforming LSTMs on certain event logs. [44] introduces Gated Convolutional Neural Networks [27] and Key-Value-Predict [26] Attention Networks. The former merges convolutional networks with a gating mechanism, akin to those used in LSTMs and GRUs, while the latter employs an attention mechanism to learn the correlation between pairs of elements within the input sequence. The primary advantage of convolutional neural networks lies in their computational efficiency relative to RNNs. However, they present increased design complexity due to variations such as stacking pooling layers and determining the appropriate size for the convolution filter.

Additionally, several approaches are testing novel innovative architectures for predictive monitoring. [101] uses Generative Adversarial Networks (GANs) [39] to predict the next activity and timestamp in an ongoing process instance, hypothesizing that GANs could potentially minimize the requirement for huge amounts of training data. [18] and [74] explore the application of Transformer architectures [108] in predictive monitoring. These architectures rely solely on attention mechanisms, hence potentially mitigating the performance degradation experienced by RNNs with long sequences and improving both training and inference speeds. However, despite the promising outcomes, the reliability of these approaches across a wide variety of datasets remains a subject of further research.

Finally, several papers employ explicit process models to assist in prefix encoding. [102] constructs feature vectors by executing a token replay of a prefix over a process model. The resulting vectors encapsulate information about the most recent activation on a Petri net, the time of activation (represented as a decay function), the number of tokens, and the event log attributes. These vectors are subsequently input into a deep feed-forward neural network. [114] explores the use of Gated Graph Neural Networks, with their adjacency matrix either based on the prefix events or the Directly Follows Graph (DFG) extracted from the event log. The graph edges contain information about the prefix. [109] investigates the use of Graph Convolutional Networks (GCNs) for predicting the next activity, focusing on variations in encoding methods for the DFG-based adjacency matrix (binary, weighted, Laplacian matrix, etc.).

2.3 Activity suffix prediction

The vast majority of the techniques for suffix prediction are based on approaches that also predict the next activity [33, 99, 36, 52, 22, 59, 49, 100, 51]. These approaches are trained

to predict the next activity in the sequence, then the prediction of the next activity is repeated until the end-of-case token is predicted or the maximum trace length is reached. The predicted suffix is the sequence of activities predicted by the model. However, there are also approaches that directly predict the suffix [25, 94] which compact the full suffix into a single label, hence bypassing the need for incremental predictions.

Between the approaches that are trained to predict the next activity, the distinction among them lies in *how the next activity is sampled from the predicted probability distribution*. In the case of next activity prediction, this step is straightforward, as the selected activity is always the one with the highest probability. However, using this strategy for suffix prediction can potentially lead to forecasting suboptimal suffixes. This is because the network may become trapped in predicting the same set of activities, hence impeding sufficient diversity during the predictions. Therefore, the taxonomy from Table 2.1 is extended to incorporate the following sampling strategies from the state of the art:

- *Argmax (ARG)*: in this approach, the next predicted activity in the suffix is always selected to be the one with the highest probability.
- *Random (RAND)*: with this method, the next predicted activity in the suffix is chosen randomly, following the predicted probability distribution.
- *Beam (BEAM)*: beam search is a heuristic search algorithm that explores the most promising nodes. In this context, it retains a set (beam) of the top k candidate sequences at each time step and expands them further. It allows for a more diverse set of predictions, increasing the likelihood of finding the most suitable suffix. This technique balances the need for diversity and accuracy in predicted suffixes.
- *One-shot (ONE)*: this strategy compresses the suffix into a single label, with the network being trained to predict it. Unlike other methods, the network does not predict individual activities within the suffix. Instead, it predicts the entire suffix at once, providing a holistic prediction of the process outcome.

Table 2.3 shows the studies that focus on activity suffix prediction. In [99] they relied on the *argmax* selection method, while the approach in [36], based on the architecture of [99], controls the selection of the next activity during the suffix prediction phase by verifying its compliance with a set of mined LTL rules. This approach uses the Beam search strategy to obtain the suffix. The methods from [52, 22, 59, 49] also employ the *argmax* sampling

approach. However, the study by [22] compared the *random* approach with the *argmax* one, noting its behavior depends on the event characteristics of the event log. Specifically, the *random* approach performs better with more complex logs and longer traces, while the *argmax* performs better with simpler logs and shorter traces. [100] proposes an encoder-decoder LSTM trained as a GAN. In this setup, the encoder-decoder serves as a suffix generator aiming to deceive the discriminator, which is another LSTM trained to differentiate between fake suffixes from the generator and real suffixes.

Between the approaches that directly predict the suffix, [25] encodes the entire prefix into a single representation using embeddings, capturing temporal information such as the time between events, the hour, and the day of the month for each event, among other aspects. A feed-forward network is then trained to predict the suffix identifier, enabling a one-shot prediction. In [94], prefixes are replayed in a Petri net process model to identify the set of activated places —behavioral context—, followed by the training of an LSTM to predict the next sequence of attributes given a prefix of attributes —data context—.

Finally, the most similar suffix from the training set is selected based on both its behavioral and data context for a given prefix during inference. Finally, [51] explores multiple architectures for suffix prediction, including LSTMs, recurrent autoencoders (either trained by maximum likelihood estimation or using a GAN approach), Transformers [108], Large Language Models (LLMs) such as BERT [29] or GPT [76], and WaveNet [103] based on CNNs. The study concludes that no architecture significantly outperforms the others across a diverse range of event logs.

2.4 Conclusions

Despite the multitude of deep learning-based approaches in predictive monitoring that endeavor to predict the next activity or sequence of activities, their performance in these tasks still has room for improvement. Furthermore, they fall short of fully exploiting the structural information available in the process models. These approaches overlook the order in which events occur in the event log, and rely predominantly on Directly-Follows Graphs (DFGs), a representation that, although simpler, is much less expressive than Petri nets. Moreover, concerning the approaches for activity suffix prediction, the majority lean heavily on predefining a sampling strategy during inference. This strategy, in most instances, does not prove robust enough to function effectively across all types of event logs. Furthermore, it often predicts

Author, Year	Reference	Network type	Sequence encoding	Event encoding	Input data	Time features	Sampling
Evermann et al., 2017	[33]	LSTM	CONT	EMB	ACT	-	RAND
Tax et al., 2017	[99]	LSTM	PRFX	OH	ACT	TSP, TSSC, WK, TMD	ARG
Francescomarino et al., 2017	[36]	LSTM	PRFX	OH	ACT, LTL	TSP, TSSC, WK, TMD	BEAM
Khan et al., 2018	[52]	DNC	PRFX	OH	ACT	TSP, TSSC, WK, TMD	ARG
Camargo et al., 2019	[22]	LSTM	PRFX	P-EMB	ACT, R	TSP	ARG, RAND
Lin et al., 2019	[59]	LSTM	CONT	EMB	ACT, ATTR	-	ARG
Jalayer et al., 2020	[49]	LSTM	PRFX	EMB	ACT	-	ARG
Taymouri et al., 2021	[100]	GAN (LSTM)	PRFX	OH	ACT	TSP	BEAM
Dalmas et al., 2021	[25]	FFN	PRFX	EMB	ACT	DUR, TSP	ONE
Sun et al., 2021	[94]	LSTM	PRFX	GRPH, OH	ACT, ATTR	DUR, WK, DATE	ONE
Ketyko et al., 2021	[51]	AE, TRANS, LSTM, LLM	PRFX	EMB	ACT	TSP, TSSC	ARG

Table 2.3: Collected studies based on deep learning for activity suffix prediction.

suffixes that are excessively short, as in the case of the *beam search*.

Another conclusion is that the state-of-the-art in predictive monitoring still lacks a standardized methodology for conducting fair comparisons across different approaches and multiple event logs. First, there is a considerable inconsistency in the data partitioning strategies used for evaluation. Many studies do not employ cross-validation or even use the same random seed for splitting the data, leading to potentially skewed or incomparable results. Second, the metrics used for comparison also vary between studies, making it challenging to draw definitive conclusions about the relative effectiveness of different methods. Lastly, there is an absence of robust statistical tests to establish the significance of observed differences between methods. Without such tests, it is difficult to reliably ascertain whether one approach outperforms another or if observed differences are merely due to random fluctuations or methodological discrepancies. This lack of standardization hampers the progress of the field by obscuring which methods are superior and why.

In this Ph.D. dissertation, we first propose a methodology to compare multiple predictive monitoring approaches across various event logs. This methodology is both robust and informative. Its robustness is ensured by supporting a cross-validation procedure and a series of statistical tests, while its informativeness is derived from providing a ranking of the

approaches, coupled with a more fine-grained comparison. This latter comparison takes into account the individual results of cross-validation testing. Leveraging this methodology, we introduce TACO and ASTON. These predictive monitoring models exploit the information contained in both the structure of the process model and its interactions with the activity prefix. This information is extracted through a Graph Recurrent Neural Network, which considers the complete sequence of interactions between the prefix and the process model. Lastly, we present DOGE, a novel predictive monitoring approach that learns the optimal sampling strategy for activity suffix prediction. This approach builds upon and enhances the sampling strategy proposed in ASTON, marking a significant advancement in the field.

CHAPTER 3

BENCHMARKING

*A man who trusts everyone is a fool and a man
who trusts no one is a fool.*

— Lews Therin Telamon, *Wheel of Time*

Evaluation plays a critical role in the development and validation of predictive monitoring approaches, providing both a measure of effectiveness and a means for comparison. A consistent and scientifically robust evaluation methodology is indispensable, not only for ensuring that the results are replicable and reliable but also for facilitating a meaningful comparison across different methods and configurations. In light of this, the present chapter outlines the standardized procedure that will be adopted throughout this thesis for both next activity prediction and activity suffix prediction problems. This procedure encompasses four essential components: (i) the selection and characterization of event logs that may be used for evaluation, providing flexibility by not being restricted to specific logs; (ii) guidelines for splitting the event logs in a manner that ensures that the testing performance accurately reflects the ability of the model to capture the entire behavioral spectrum present in the event log; (iii) the specification of evaluation metrics to provide a quantifiable measure of model performance, and (iv) a framework for utilizing statistical tests in comparing different approaches, allowing for results to be interpreted cohesively, regardless of the quantity of approaches and event logs involved.

Figure 3.1 shows the sequential and interactive nature of these four key components — *Datasets*, *Partitions*, *Metrics*, and *Statistical Tests*— that make up the evaluation framework.

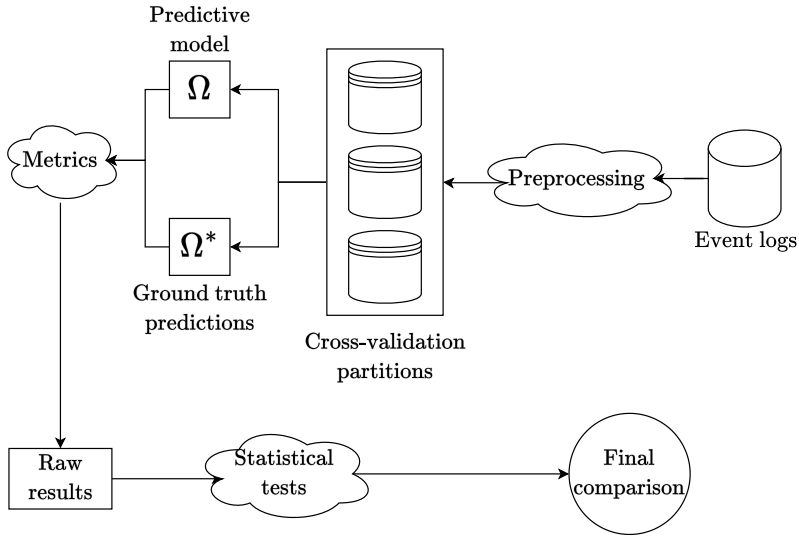


Figure 3.1: Proposed evaluation framework for evaluating predictive monitoring approaches.

Initially, a comprehensive set of event logs is selected to form the datasets for evaluation. Following this, during the preprocessing stage, each dataset is divided using a reproducible cross-validation strategy. This ensures that the partitions created can be replicated by other researchers. Subsequently, the predictions from the neural models are evaluated against the ground truth using one of the metrics defined in this chapter, yielding raw evaluation results. Finally, a suite of statistical tests is applied to these raw results to derive a reliable and statistically-backed comparison among the different predictive monitoring approaches.

Together, these elements form a comprehensive framework designed to reflect the true performance of the predictive monitoring models, offering insights that are both informative and statistically sound. This rigorous evaluation transcends mere counting of performance metrics, incorporating statistical tests to provide solid empirical backing. Unlike post-hoc speculation, where conclusions are formed after observing the data without rigorous testing, this methodology seeks to ensure that the findings are grounded in statistically valid comparisons. Thus, the use of statistical tests prevents the drawing of arbitrary or unwarranted conclusions, providing an analysis that is more dependable and less prone to subjective interpretation. This methodological refinement resonates with a commitment to accuracy and robustness, enhancing the overall credibility of the results presented throughout this disserta-

tion. This framework is publicly available¹ so any researcher can use it to evaluate their own approaches and compare them with the ones presented in this thesis. Additional experimental results of the benchmark are available in the Appendix A

The contents of this chapter are extracted from the following publication:

Efrén Rama-Maneiro, Juan Vidal and Manuel Lama. **Deep Learning for Predictive Business Process Monitoring: Review and Benchmark.** *IEEE Transactions on Services Computing*, 16:739-756, doi: 10.1109/TSC.2021.3139807.

Author's contributions:

Efrén Rama-Maneiro: Methodology, Software, Validation, Formal Analysis, Investigation, Data Curation, Writing — Original Draft.

Juan Vidal: Conceptualization, Resources, Writing — Review & Editing, Visualization, Supervision, Project administration, Funding acquisition, Investigation.

Manuel Lama: Conceptualization, Resources, Writing — Review & Editing, Visualization, Supervision, Project administration, Funding acquisition, Investigation.

3.1 Event logs

The event logs utilized throughout this thesis encompass a diverse array of real-life cases from various domains, each possessing unique characteristics. This selection has been curated to facilitate a robust evaluation of each approach, considering multiple dimensions of the data: the size of event logs, the length of individual traces, the variability in time, and other distinct features. Such a diverse dataset allows for a comprehensive assessment that takes into account the multitude of scenarios that may be encountered in practice. The specific event logs, along with their variations that have been considered in this Ph.D. dissertation, are as follows:

- *BPI Challenge 2012* [107]: this event log represents a loan application process and captures the complete life cycle of applications from a global bank. We consider different logs for the subprocesses *A*, *O* and *W* as well as for the variations that only consider the activities that have a *Completed* lifecycle transition.

¹<https://gitlab.citius.usc.es/efren.rama/pmdlcomparator>

- *BPI Challenge 2013* [91, 92]: this log details the problem management processes related to closed problems at Volvo IT. We consider the closed problems [91] and the full incidents [92] event log.
- *Helpdesk* [110]: this log records the ticketing management process of the helpdesk of an Italian software company. Each case starts with the insertion of a new ticket into the ticketing management system and finishes when the issues is solved and the case closed.
- *Environmental Permit* [17]: originating from the CoSeLoG project, this data details the execution of a building permit application process across five different municipalities in the Netherlands, capturing a wide variety of attributes related to the process.
- *Sepsis* [63]: this medical event log documents the events of sepsis cases in a hospital. Attributes like group responsible for the activity and results of tests are included, and the timestamps have been randomized while keeping the time between events unaltered.
- *Nasa* [56]: this log records method-call level events describing a single run of an exhaustive unit test suite for NASA’s Crew Exploration Vehicle. Each trace represents the execution of a single unit test method. Lifecycle information is captured, reflecting method call and return and illustrating a method-call hierarchy.

Table 3.1 provides a comprehensive statistical summary of the event logs used in this study, offering both structural and temporal perspectives. Structurally, the table displays the number of traces, activities, and events along with the number of different variants, and average and maximum case lengths. These measures show the complexity and variation in process structures across different logs. Temporally, this table depicts the average and maximum durations for individual events and traces, revealing the time dynamics within each process. These two perspectives together provide a multi-dimensional view of the event logs, showcasing the diverse nature of real-world processes captured in this study.

In particular, Table 3.1 highlights the substantial variation in both structural complexity and temporal dynamics across different event logs. This diversity is apparent in several dimensions:

- *Number of Distinct Activities*: the event logs vary significantly in terms of the number of distinct activities they encompass. For instance, the *BPI Challenge 2013 Closed Problems* log has as few as 7 activities, while the *BPI Challenge 2012* log contains as many as 36 activities.

- *Average Case Length*: there are wide disparities in the average case length across the logs. Specifically, the *BPI Challenge 2013 Closed Problems* log has a short average case length of 4.48, while the *NASA* log features much lengthier cases, with an average of 28.7.
- *Number of Unique Process Variants*: the logs exhibit considerable variability in the number of unique process variants they contain, revealing the structural rigidity or flexibility inherent in each. On one end of the spectrum, the *Environmental Permit* log has only 116 unique variants, indicating a more predictable process. On the other end, the *BPI Challenge 2012* log has 4366 unique variants, indicating a more complex process.
- *Temporal Metrics*: the logs differ widely in terms of case and event durations. For example, the average case duration ranges from a quick 5.41 days in the *Environmental Permit* log to a much longer 178.88 days in the *BPI Challenge 2013 Closed Problems* log.
- *Special Cases*: some logs present anomalies in their temporal metrics. For example, the *NASA* log records zero days for all its time measures, attributable to the log recording

Event log	Traces	Activities	Events	Avg. case length	Max. case length	Avg. event duration	Max. event duration	Avg. case duration	Max. case duration	Variants
BPI-2012 [107]	13087	36	262200	20.04	175	0.45	102.85	8.62	137.22	4366
BPI-2012-A [107]	13087	10	60849	4.65	8	2.21	89.55	8.08	91.46	17
BPI-2012-C [107]	13087	23	164506	12.57	96	0.74	30.92	8.61	91.46	4336
BPI-2012-O [107]	5015	7	31244	6.23	30	3.28	69.93	17.18	89.55	168
BPI-2012-W [107]	9658	19	170107	17.61	156	0.7	102.85	11.69	137.22	2621
BPI-2012-W-C [107]	9658	6	72413	7.5	74	1.75	30.92	11.4	91.04	2263
BPI-2013-C-P [91]	1487	7	6660	4.48	35	51.42	2254.84	178.88	2254.85	327
BPI-2013-I [92]	7554	13	65533	8.68	123	1.57	722.25	12.08	771.35	2278
Env-permit [17]	1434	27	8577	5.98	25	1.09	268.97	5.41	275.84	116
Helpdesk [110]	4580	14	21348	4.66	15	11.16	59.92	40.86	59.99	226
Nasa [56]	2566	94	73638	28.7	50	0.00	0.00	0.00	0.00	2513
Sepsis [63]	1049	16	15214	14.48	185	2.11	417.26	28.48	422.32	845

Table 3.1: Statistics of the event logs. Time-related measures are shown in days.

Dataset	Attributes
BPI-2012 [107]	lifecycle:transition, org:resource
BPI-2012-A [107]	org:resource
BPI-2012-C [107]	org:resource
BPI-2012-O [107]	org:resource
BPI-2012-W [107]	lifecycle:transition, org:resource
BPI-2012-W-C [107]	org:resource
BPI-2013-C-P [91]	org:group, org:resource
BPI-2013-I [92]	org:group, resource country, organization involved, org:role, impact product, lifecycle:transition, org:resource
Env-permit [17]	org:group,org:resource
Helpdesk [110]	org:resource
Nasa [56]	apploc:joinpoint, apprun:exthrowtype, apploc:etype, apprun:excatchtype, lifecycle:transition
Sepsis [63]	org:group

Table 3.2: Event attributes used for “data-aware” approaches.

of unit test execution times, which occur at a significantly faster rate than traditional business processes.

This Ph.D. thesis narrows its scope to the utilization of event-level attributes, while consciously excluding trace-level attributes from consideration. The range of attributes, as outlined in Table 3.2, signifies the possibilities offered by different event logs. However, it is worth noting that the availability of these attributes does not need their usage in entirety by any particular approach. The attributes associated with each event log present a diverse set that can be employed as required. An approach can freely choose to incorporate any subset of these attributes depending on the specific requirements of the use case. Alternatively, it may opt to abstain from using any attributes altogether. This flexibility allows for customization and adaptability in the processing and analysis of event logs. Table 3.2 encapsulates the variety of attributes offered by the event logs. The *BPI 2013 Incidents* log, for instance, provides a rich set of attributes like *org:group*, *organization involved*, and *org:role*, among others. At the other end of the spectrum, logs like *BPI 2012 Complete* and *BPI 2012 A* offer a single attribute, *org:resource*.

3.2 Preprocessing

In the realm of machine learning and predictive analytics, how the data is prepared and split plays a pivotal role in ensuring robustness and generalizability of models. Properly segregating data into training, validation, and test sets is of paramount importance to ensure that models are not only learning the underlying patterns in the data but can also generalize to unseen instances. This helps in mitigating overfitting, where a model performs exceptionally well on the training data but fails to do so on new, unseen data. Utilizing techniques such as 5-fold cross-validation further augments this approach. In 5-fold cross-validation, the dataset is partitioned into five subsets; the model is trained on four of these and tested on the remaining one. This process is repeated five times, each time with a different subset as test set. This methodology provides a comprehensive understanding of the performance of the model across different data splits, ensuring reliability in its predictions.

In particular, and regarding the actual splitting of the event log into training-validation-test sets, we have chosen to adopt a variant of nested cross-validation without the hyperparameter tuning loop for evaluation. This decision was carefully considered, taking into account the unique aspects of the problem domain where traditional methods like the k-fold cross-validation applied on pairs of prefixes-targets could lead to biased results due to the specific nature of process mining data. Here, we refer as “targets” as the next activity suffix or simply the next activity, depending on the predictive task. In our case, we are dealing with the full trace variants rather than individual observations of prefixes because this ensures that any trace would be split in both training and test sets, having some prefixes in the training set and some others in the test set. Therefore, the splitting procedure has been carried out on the whole traces and not in individual prefixes, being the percentages we refer to the number of traces in the event log, and not the number of prefixes.

Our evaluation method involves shuffling the traces before splitting, followed by a 5-fold cross-validation, generating 5 different pairs of training and test folds. Then, each training fold is split in a 80%-20% fashion to create a training and validation set, with the validation set used for selecting the best-performing model. We want to highlight that this method provides us with 5 different test sets, ensuring a robust evaluation of our approach and not relying on a single, potentially skewed holdout test set. Consequently, the results we report reflect the behavior of any tested approach on the whole event log, following the same trace variant distribution. It is also important to note that some of the logs used are fairly small, so using a holdout test set would highly bias the results. This method mitigates also these

issues due to testing on the whole event log. Moreover, each trace in the log incorporates a special end-of-case token, signaling trace completion. This inclusion is crucial, especially for the next activity suffix prediction task, as it dictates when to stop neural model inference. For consistency, this token is incorporated in the next activity prediction task as well. Lastly, note that the prefixes considered go from length 1 to the maximum trace length of the event log, so we do not consider the empty prefix as an input to the models.

Finally, it is worth emphasizing that all partitions have been generated in a reproducible manner using a specific seed value. This ensures that any new approach can be precisely compared with the results obtained in this thesis for existing state-of-the-art methods.

3.3 Metrics

Depending on the predictive task—whether next activity prediction or activity suffix prediction—we contemplate the usage of different metrics. These metrics are the following:

- *Next activity prediction:* as the next activity prediction task is a classic multiclass classification problem, we use the *accuracy* metric. This choice is consistent with the results reported by most works in the state of the art. The accuracy measures the proportion of correct classifications relative to the total number of predictions. While other measures such as the Brier score [15], the Matthews correlation coefficient [66], F1 score, precision, and recall were also considered, we found that their results aligned with accuracy and did not provide additional relevant insights.
- *Activity suffix prediction:* when predicting an activity suffix within the context of predictive monitoring, it is vital to recognize that the activities of a process may occur in parallel [99]. Hence, we opt for the normalized Damerau-Levenshtein distance [13] metric instead of the metrics employed for the next activity prediction task. This metric measures the edit distance between two strings without excessively penalizing token transpositions. Within the predictive monitoring framework, such a transposition could signify a pair of parallel activities. These strings represent the forecasted activity suffix for a particular event prefix and its associated actual activity suffix. The Damerau-Levenshtein distance, which quantifies the number of deletions, insertions, substitutions, and transpositions required to transform one string to another, is defined as follows:

Definition 12: Damerau-Levenshtein edit distance

Let a and b be two strings, with $d_{a,b}(i, j)$ denoting the distance function between prefixes of a and b up to positions i and j respectively. The Damerau-Levenshtein distance is recursively defined as:

$$d_{a,b}(i, j) = \min \begin{cases} 0 & \text{if } i = j = 0, \\ d_{a,b}(i-1, j) + 1 & \text{if } i > 0, \\ d_{a,b}(i, j-1) + 1 & \text{if } j > 0, \\ d_{a,b}(i-1, j-1) + 1_{(a_i \neq b_j)} & \text{if } i, j > 0, \\ d_{a,b}(i-2, j-2) + 1_{(a_i \neq b_j)} & \text{if } i, j > 1 \text{ and } a_i = b_{j-1} \text{ and } a_{i-1} = b_j, \end{cases}$$

Each recursive case in the definition above corresponds to one of the operations considered by the Damerau-Levenshtein distance:

- $d_{a,b}(i-1, j) + 1$ means a deletion from a to b .
- $d_{a,b}(i, j-1) + 1$ means an insertion from a to b .
- $d_{a,b}(i-1, j-1) + 1_{(a_i \neq b_j)}$ means either a match or a mismatch between symbols a_i and b_j .
- $d_{a,b}(i-2, j-2) + 1_{(a_i \neq b_j)}$ means a transposition of two adjacent symbols.

To make the metric interpretation independent of both the event log under evaluation and the characteristics of its individual suffixes, the Damerau-Levenshtein metric is normalized as follows:

Definition 13: Normalized Damerau-Levenshtein distance

Let $d_{a,b}$ be the Damerau-Levenshtein distance between strings a and b . Denoting the length of a string as $|\cdot|$, the normalized Damerau-Levenshtein distance is:

$$nd_{a,b}(i, j) = 1 - \frac{d_{a,b}(|a|, |b|)}{\max(|a|, |b|)} \in [0, 1]$$

3.4 Statistical tests

In empirical research and data-driven analyses, statistical tests serve as invaluable tools in discerning genuine patterns from mere coincidences. Their primary role is to assess the likelihood that observed results are not just random occurrences but reflect underlying truths. By providing objective measures of significance, these tests provide confidence in the assertions made based on experimental or observational data. In the domain of machine learning and predictive monitoring, where multiple algorithms or approaches compete to demonstrate superiority, it is paramount to validate any perceived performance differences. Without rigorous statistical tests, claims of one method superiority over another may be built on shaky grounds, vulnerable to scrutiny and challenge. In this Ph.D dissertation, all analysis of the results are supported by statistical tests.

Traditionally, the statistical tests performed in most publications from any discipline are called Null Hypothesis Statistical Tests (NHST). NHST are a prevalent method in scientific research designed to assess the strength of the evidence against a null hypothesis. In essence, the null hypothesis (H_0) indicates that there is no effect or no difference, and it is the default assumption that researchers aim to test against. Researchers will then introduce an alternative hypothesis (H_1), which asserts the presence of an effect or difference. The objective of NHST is to determine the probability, given the observed data, that the results align with the null hypothesis. This probability is commonly referred to as the p-value. A small p-value, typically below a predetermined significance level such as 0.05, suggests that the observed data would be unlikely under the null hypothesis, leading to its rejection in favor of the alternative hypothesis. As noted in [9, 7], the main problems of NHST are the following ones:

- *NHST does not estimate probabilities of hypotheses:* while it is common to interpret p-values as giving the probability of the observed data under the null hypothesis, they do not offer direct insight into the probability of either hypothesis being true. NHST does not quantify the evidence in favor of the alternative hypothesis or provide a direct estimate of the probability of the hypotheses.
- *Point-wise null hypotheses are practically always false:* when subjected to enough data, almost any null hypothesis can be rejected. The precise null hypothesis —like two means being exactly equal— is often an unrealistic representation of reality.
- *The p-value does not separate between the effect size and the sample size:* p-values can

be made arbitrarily small by simply increasing the sample size, even if the observed effect is negligible and possibly irrelevant in practical scenarios.

- *NHST yields no information about the null hypothesis:* rejecting or failing to reject the null hypothesis is not the same as accepting it. NHST does not provide any positive evidence in favor of the null hypothesis, even when it is not rejected.
- *There is no principled way to decide the α level:* the common threshold of 0.05 is arbitrary. There is no foundational rationale for its universal application, and adjusting it can significantly alter the results and conclusions of a test.

Therefore, in this Ph.D dissertation, we resort to Bayesian statistical tests to support the analysis of the results. Unlike NHST, Bayesian statistical tests are based on the Bayesian probabilistic framework that quantifies the uncertainty in parameters using probability distributions. In a Bayesian setting, both the prior distribution and the observed data are combined to update beliefs about the parameters of interest, resulting in a posterior distribution, which provides a continuous measure of evidence, allowing for direct probability statements about parameters and hypotheses. Moreover, Bayesian tests can offer richer insights by evaluating the probability of both the null and the alternative hypotheses, and they are naturally equipped to handle estimation of effect sizes and their uncertainties. This approach is more interpretable and aligns closely with the way humans intuitively understand uncertainty and evidence.

In this Ph.D dissertation, we propose to perform the comparison between the approaches in a two-stage fashion. First, a Bayesian ranking analysis algorithm [20] is applied to discern which approaches have the potential to outperform the others. Following this, we perform pairwise comparisons using a hierarchical Bayesian test [9] among the top-ranking methods. This two-stage method of comparison minimizes the number of pairwise comparisons, streamlining the interpretation of the results. Furthermore, it reduces the likelihood of committing Type I errors, which occur when a true null hypothesis is incorrectly rejected. In simpler terms, a Type I error means falsely claiming an effect or difference when none actually exists. The next subsections describe in detail these two statistical tests.

3.4.1 Bayesian statistical test for ranking

The Bayesian statistical test for ranking is based on the Plackett-Luce model. This model has two interesting properties: (i) the parameters of the model represent the probability of

an algorithm being ranked in the first position; and (ii) it fulfills Luce's axiom [62], which implies that the probability of the ordering between two algorithms will be the same no matter how many algorithms are added to the comparison. The Plackett-Luce model for ranking is defined as follows:

Definition 14: Plackett Luce model [20]

Let $\sigma = (\sigma_1, \dots, \sigma_n)$ be a vector of rankings for a specific dataset, where $\sigma_i = j$ implies that the j -th algorithm is ranked in the i -th position, and $w = (w_1, \dots, w_n)$ be a vector of learnable weights. Then, the probability that the Plackett-Luce model assigns to a ranking is:

$$P_{PL}(\sigma) = \prod_{i=1}^n \frac{w_{\sigma_i}}{\sum_{j=i}^n w_{\sigma_j}}$$

This expression is approximated with a Bayesian model which will enable us to benefit from the advantages of not relying on NHST tests. Thus, according to Bayesian models, we need to define the *sampling function*, which is the distribution from which the data comes—the set of rankings—. In this case, the *sampling function* is the Plackett Luce model. Then, we assume that the parameters of the distribution are unknown, so each parameter will have a probability associated. We also assume that the data has a *prior distribution* which is subsequently modified with the data collected, updating the uncertainty of the model, and obtaining the posterior distribution of the model. This update is performed by means of the Bayes rule:

Definition 15: Bayes rule

Let w be a vector of parameters of the sampling function —weights from the Plackett-Luce model— and R be the set of rankings for every tested dataset. Then, the update procedure is performed using the following expression:

$$P(w|R) \propto P(R|w)P(w)$$

Where:

- $P(w|R)$ is the *posterior distribution*, which represents our updated belief about the

weights after observing the rankings.

- $P(R|w)$ is the *likelihood*, which measures the compatibility of the observed data (rankings) with the given weights.
- $P(w)$ is the *prior distribution*, which represents our initial belief about the weights before observing the data.

The Dirichlet distribution is chosen as the prior in the Bayesian Plackett-Luce model due to its conjugacy with the multinomial distribution. In Bayesian statistics, *conjugacy* means that a prior distribution is the conjugate of a likelihood function if the resulting posterior distribution belongs to the same family of distributions as the prior. In this case, if Dirichlet is selected as a prior distribution and it is updated using a multinomial likelihood, the resulting posterior distribution will also be Dirichlet, simplifying the computational process. On the other hand, the multinomial distribution models the outcomes of multiple categories and their associated probabilities. For instance, consider rolling a biased six-faced die, where each face of the die, representing a category, has a different probability of being rolled. When rolled multiple times, the number of times each face appears follows a multinomial distribution.

In the Plackett-Luce model, choosing the top-ranked item among a set of items can be thought as drawing from a multinomial distribution: each item has a certain *weight*, similar to the probability of each category in the multinomial distribution, which influences its chance of being picked first. In the Plackett-Luce model, the initial action of selecting the top-ranked item among a set can be thought of as a draw from a multinomial distribution. Considering this, the Bayesian Plackett-Luce model for ranking algorithms can be defined as follows:

Definition 16: Bayesian Plackett-Luce model for ranking [20]

Let w be a vector of parameters of the sampling function (weights from the Plackett-Luce model) and R be the set of rankings for every tested dataset. The Bayesian Plackett-Luce model defines the posterior distribution of the weights given the observed rankings as:

$$P(w|R) \propto \text{Dir}(w; \alpha) \prod_{\sigma \in R} P_{PL}(\sigma; w)$$

Where:

- $\text{Dir}(w; \alpha)$ is the Dirichlet prior distribution of the weights, with α being the hyper-

parameters of the prior.

- $P_{PL}(\sigma; w)$ is the Plackett-Luce model, which gives the probability of a ranking σ given the weights w .

Note that the expression does not have a closed form, but it can be approximated using Markov Chain Monte Carlo (MCMC) [42] methods to analyze the distribution of the sampled weights. From the posterior distribution, two significant insights can be derived. The first one is the expected ranking for each of the tested approaches. At each MCMC iteration, a ranking based on the sampled weights can be established. By averaging these rankings over all iterations, a *mean ranking* is obtained, giving a probabilistic estimate of the most likely ordering of the approaches. The second insight is a credible interval for these rankings, which we can compute by obtaining the 5%, 50%, and 95% quantiles of the posterior distribution for the weights. This interval can be plotted into an error bar graph: if the credible intervals for a pair of approaches do not overlap, we can affirm that there are significant statistical differences between them in terms of ranking.

Figure 3.2 and Table 3.3 show the ranking comparison of four different approaches. Figure 3.2 represents the aforementioned error bar graph, while Table 3.3 displays the expected probability and mean ranking of each approach being the best. This table specifically includes two metrics: the expected rank and the probability of each approach being the best. The expected rank offers an aggregate measure of how each algorithm generally performs across multiple event logs, providing an intuitive ranking metric. The probability of being the best gives an estimation of the likelihood of each algorithm outperforming all others, offering another perspective on algorithmic robustness. According to the example in Table 3.3, the best approach would be “B”, with a 50% probability and an average rank of 1.5. Furthermore, Figure 3.2 provides additional insights. According to this figure, since the quantiles of the “B” approach do not overlap with those of the other approaches, we can affirm that “B” is significantly better than the other approaches *in terms of ranking*. A similar conclusion can be made about “C”, affirming that it is significantly worse than the other approaches in terms of ranking. Since the error bars for “A” and “D” overlap, no definitive conclusions can be drawn about the comparison between these two approaches.

It is worth noting that a credible interval in Bayesian statistics is different from a confidence interval in frequentist statistics. A credible interval provides a range of values for the parameter—in this case, the weights or the ranking—within which there is a specified probability—

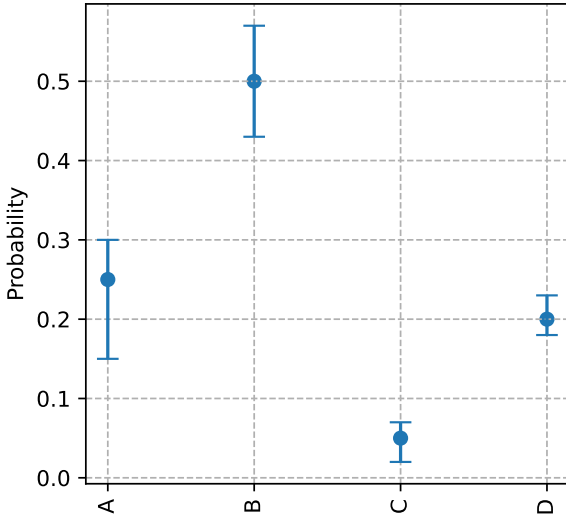


Figure 3.2: Example of the Bayesian Plackett-Luce model for ranking.

	B	A	D	C
Rank	1.5	2.5	2.7	3
Prob. (%)	50	25	20	5

Table 3.3: Example Plackett-Luce rankings and posterior probabilities of four example approaches.

e.g., 90%— that the true value of the parameter lies, given the observed data. This is a direct probability statement about the parameter, which is different from a confidence interval that provides a range of values that would contain the true parameter value in a specified proportion —e.g., 95%— of hypothetical repeated samples of the data.

3.4.2 Hierarchical bayesian statistical test

The Hierarchical Bayesian test is a powerful tool that addresses the limitations of the Bayesian ranking test. While the Bayesian ranking test focuses on the absolute rankings, the Hierarchical Bayesian test takes into account the specific performance differences between the algorithms. This allows it to discern not only whether two pairs of approaches are different or equal, but also

the magnitude of these differences. The Hierarchical Bayesian test is based on a hierarchical probabilistic model, which is an extension of the Bayesian correlated t-test, capable of making inferences on multiple datasets while accounting for all available information: mean, standard deviation, and correlation. More specifically:

- The mean refers to the average difference in performance between the algorithms across all the folds of the cross-validation.
- The standard deviation measures how much this difference varies across the different folds of the cross-validation.
- The correlation refers to the relationship between the performances of the algorithms on the same dataset due to the overlapping training set. This correlation arises because the same dataset is used to train the algorithms, which means that the performance of the algorithms on the same dataset is not independent.

Therefore, this test not only informs of the performance of a pair of approaches in a particular dataset, but also tells us which approach works better on all datasets, overall. It is important to clarify a common misconception at this point. One might expect that the overall probability is simply the mean of the probabilities of the individual datasets. However, this is not the case. In the Hierarchical Bayesian test, the overall probability is a synthesis of both shared properties across datasets and individual variations specific to each dataset. This hierarchical model accounts for the performance patterns shared by all datasets, but also recognizes the unique characteristics of each dataset. This ensures that the final overall probability is a more granular representation of the performance of the algorithm, taking into account both the general trend and specific exceptions. Formally, the hierarchical Bayesian test is defined as follows:

Definition 17: Hierarchical Bayesian Test [9]

Let $x_i = \{x_{i1}, x_{i2}, \dots, x_{in}\}$ be the n cross-validation measures of the i -th dataset, μ_i be the mean difference of accuracies in the single datasets, μ_0 be the average difference of accuracy between the two classifiers on the population of data sets, and σ_i be the standard deviations. The Hierarchical Bayesian test defines the following model [9]:

1. $x_i \sim MVN(\mu_i, \Sigma_i)$: The cross-validation measures of the i -th dataset are jointly

multivariate-normal distributed with the same mean, same variance, and correlation. This means that the measures are correlated since they come from overlapping training sets.

2. $\mu_1, \dots, \mu_q \sim t(\mu_0, \sigma_0, \nu)$: The mean difference of accuracies in the single datasets, μ_i , depends on μ_0 . Using the t distribution provides robustness against potential outliers better than a normal distribution.
3. $\sigma_1, \dots, \sigma_q \sim \text{unif}(0, \bar{\sigma})$: The standard deviations are uniformly distributed between 0 and the average standard deviation.

Delving deeper into the relationships between μ_0 and the individual μ_i , we can discern the hierarchical structure within the Bayesian test. The term μ_0 symbolizes the global difference in performance between the two algorithms over all datasets. Conversely, each μ_i represents the specific performance difference on the i th dataset. Through this hierarchical model, there is a belief that the individual dataset differences, while unique, are generated from a shared distribution centered around the common value, μ_0 . This belief indicates that these performance differences are not entirely isolated but are instead influenced by a global perspective.

The hierarchical Bayesian test estimates the mean difference in performance between the algorithms, the standard deviation of this difference, and the correlation between the performances of the algorithms on the same dataset due to the overlapping training set. It applies shrinkage to these estimates to improve their accuracy and interpretability, i.e., this test pulls individual estimates towards a global central value—like μ_0 —, especially when individual datasets are small or noisy. The test also uses the concept of the Region of Practical Equivalence (ROPE) to determine whether the difference in performance between the algorithms is not only statistically significant but also *practically* significant. A ROPE is a range of values around zero that are considered to be practically equivalent to zero. If the 95% Highest Density Interval of the posterior distribution of the difference between two parameters falls entirely within the ROPE, then the difference is considered to be practically insignificant. Following the example of [9], we set up a ROPE value of 1% for our analyses.

As an example, let us compare two approaches, A and B , on four different datasets $D_1, \dots, D_4$. The results of this example Hierarchical Bayesian Test are highlighted in Table 3.4. This table compares two approaches, A and B , detailing the likelihood of A outperforming B ($A > B$), both being equivalent ($A = B$), or A underperforming relative to B

($A < B$). To assess if A is at least as good as B , one can sum the probabilities from $A > B$ and $A = B$. In this analysis, a probability exceeding 95% is considered statistically significant. In particular, this table shows that approach A is significantly better than approach B in dataset D_1 with a 95% probability, not being significantly worse in dataset D_4 with 95% probability —45% + 50%—. Overall, we cannot affirm that approach A is better than approach B , but we can affirm that approach A is not worse than approach B with 95% probability —72% + 23%—.

	$A > B$	$A = B$	$A < B$
D_1	95%	3%	2%
D_2	55%	35%	10%
D_3	20%	70%	10%
D_4	45%	50%	5%
Overall	72%	23%	5%

Table 3.4: Example hierarchical bayesian tests comparing two approaches, A and B .

3.5 Conclusions

One of the main problems in predictive monitoring research is the heterogeneity in comparing the performance of the different state-of-the-art approaches, which makes it very difficult, if not impossible, to establish a fair comparison between them. In this chapter, we presented a benchmarking methodology that addresses this drawback, supporting the experimental work throughout this Ph.D. dissertation and that is built on four key pillars:

- (i) *Comprehensive selection of event logs*: our wide-ranging choice of event logs, which vary significantly in complexity and characteristics, ensures that the evaluations capture the performance of tested approaches across a broad spectrum of contexts. This helps provide a more holistic understanding of their strengths and weaknesses, and ensures that our findings are broadly applicable.
- (ii) *Dataset preprocessing strategy*: our strategy for preprocessing the event logs takes into account the unique nature of process mining data and employs a tailored cross-validation methodology. By doing so, it offers a more accurate estimation of how an approach will perform on a given event log, thereby reducing the likelihood of misleading results due to inappropriate preprocessing.

- (iii) *Specification of Evaluation Metrics*: to ensure a fair and standardized comparison, we employ a set of standardized metrics. These metrics serve as the basis for quantifying the performance of various approaches, thus providing a concrete, numerical basis for our conclusions. The inclusion of these metrics in the evaluation framework ensures that the comparison is not only robust but also directly interpretable and widely applicable.
- (iv) *Two-stage statistical testing methodology*: our use of a two-stage statistical testing approach provides a robust and comprehensive comparison of the tested methods across our selection of event logs. This methodology not only reduces the likelihood of Type I errors, but also simplifies the interpretation of the results, making it easier to draw meaningful conclusions.

These pillars work together to support an experimental methodology that is both robust and comprehensive, ensuring that the evaluations and comparisons of various approaches throughout this dissertation are both reliable and insightful.

CHAPTER 4

GRAPH-BASED NEXT ACTIVITY PREDICTION

Your next line is...

— Joseph Joestar, *Jojo's Bizarre Adventure*

In this chapter, we present TACO (Recurrent **T** Graph **C**onvolutional **P**rocess Predictor), a deep learning model specifically designed to predict the next activity in an ongoing business process instance. TACO is unique in its ability to leverage not just the information available in the event log, but also the structural properties of the underlying process model and its dynamic interactions with the activity prefix. To accomplish this, TACO takes advantage of Graph Convolutional Networks (GCNs), which are adept at extracting the most relevant features from the process model at each timestep. To allow for the processing of an entire event prefix, rather than a single event, we further enhance the capabilities of GCNs by extending them into Graph Recurrent Neural Networks (GRNNs). This innovative approach enables TACO to capture and exploit the spatio-temporal dependencies between activities over time, thus offering superior predictive performance.

For the sake of clarity and consistency throughout this chapter, we will refer to a single example event log and process model. Specifically, Table 4.1 will serve as our example event log, and Figure 4.1 will represent the corresponding process model. The event log in Table 4.1 showcases two representative traces from the *BPI-2012-W-C* event log. Meanwhile, Figure 4.1 illustrates the process model capturing the behavior of that event log, along with the execution

semantics derived from replaying the trace “214364”.

The contents of this chapter are extracted from the following publication:

Efrén Rama-Maneiro, Juan Vidal and Manuel Lama. **Embedding Graph Convolutional Networks in Recurrent Neural Networks for Predictive Monitoring**. *IEEE Transactions on Knowledge and Data Engineering*, 1-16, ISSN:1041-4347, doi: 10.1109/TKDE.2023.3286017.

Author’s contributions:

Efrén Rama-Maneiro: Methodology, Software, Validation, Formal Analysis, Investigation, Data Curation, Writing — Original Draft.

Juan Vidal: Conceptualization, Resources, Writing — Review & Editing, Visualization, Supervision, Project administration, Funding acquisition, Investigation.

Manuel Lama: Conceptualization, Resources, Writing — Review & Editing, Visualization, Supervision, Project administration, Funding acquisition, Investigation.

Trace ID	Activity	Timestamp	Resource
173712	W_Afhandelen leads (WAL)	01/10/2011 08:45:13	Gordon
173712	W_Completeren aanvraag (WCA)	02/10/2011 09:26:51	Alex
173712	W_Completeren aanvraag (WCA)	03/10/2011 10:39:04	Barney
173712	W_Completeren aanvraag (WCA)	04/10/2011 11:51:33	Adrian
214364	W_Afhandelen leads (WAL)	06/10/2011 09:10:22	Joseph
214364	W_Completeren aanvraag (WCA)	06/10/2011 15:13:22	Joseph
214364	W_Nabellen offertes (WNO)	08/10/2011 10:48:11	Enrico
214364	W_Valideren aanvraag (WVA)	08/10/2011 11:10:32	Harambe

Table 4.1: Example extracted from the BPI-2012-W-C event log.

4.1 Graph Recurrent Neural Networks

Graph Neural Networks (GNNs) are a powerful class of neural networks designed to handle data structured as graphs. Traditional neural networks are not inherently suited for graph data, as they assume a fixed-size input and do not account for the relational structure inherent in graphs. GNNs, on the other hand, are designed to work in non-euclidean domains, being, thus capable of capturing dependencies between nodes in a graph, making them particularly useful for tasks that involve relational or networked data, such as social network analysis or predictive

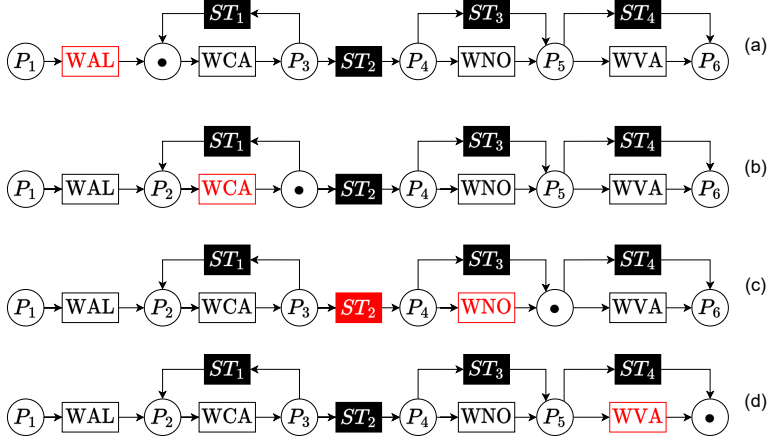


Figure 4.1: Process model for the event log in Table 4.1 and execution semantics —in red— for trace 214624.

monitoring. One specific type of GNN is the Graph Convolutional Network (GCN) [54], which is designed to operate directly on graphs and leverage their structural information.

Definition 18: Graph Convolutional Network (GCN) Operator [54]

Let I_N be an identity matrix, A the adjacency matrix of a graph, and D its degree matrix. Then, the GCN operator is defined as:

$$g_\theta \star x \approx \theta(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}x)$$

Here, $\star$ denotes the convolution operation in the graph domain, aggregating information from the neighbors of a given node. θ is a learnable parameter that scales the result.

To address issues like exploding/vanishing gradients and to ensure stability during training, the *renormalization trick* is introduced, i.e., $\tilde{A} = A + I_N$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. Thus, we achieve the following GCN operator:

$$GCN(A, X) = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta$$

Where Θ is a matrix of learnable filter parameters, and X is the matrix of features of the graph.

For tasks like predictive monitoring, where the goal is to predict future events based on historical data, this model has limitations. Firstly, it overlooks the directionality in the input graph, crucial for understanding causal relationships, due to the usage of the symmetric normalization, $\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$. Secondly, it only considers the last state in a sequence of execution states related to events, leading to potential information loss. This is evident in issues like loop overwriting, where a loop in the process may cause the graph encoding to overwrite information inadvertently.

To address the directionality concern, we adopt the random walk normalized Laplacian, $D^{-1}A$, which inherently preserves the directionality of the graph [84]:

Definition 19: GCN with random-walk normalized laplacian [84]

Let $\tilde{A}$ be the adjacency matrix of a graph with added self-loops, $\tilde{D}$ be its corresponding degree matrix, X be a feature matrix of the graph, and Θ be learnable filter parameters. Then, the Graph Convolutional Network operator with the random-walk normalized laplacian is defined as follows:

$$GCN(A, X) = \tilde{D}^{-1} \tilde{A} X \Theta$$

To solve the second issue, we propose to apply a recurrent operation to the sequence of token replays. We utilize the Gated Recurrent Unit (GRU) [23] as a foundational block. GRUs are favored for their ability to achieve results comparable to Long Short-Term Memory (LSTM) [46] networks while being computationally more efficient.

Definition 20: Gated Recurrent Unit (GRU) [23]

A Gated Recurrent Unit (GRU) is defined as follows:

$$\begin{aligned} z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\ r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \\ \tilde{h}_t &= \tanh(W_h x_t + U_h (r_t \circ h_{t-1}) + b_h) \\ h_t &= z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t \end{aligned}$$

Where:

- σ is the sigmoid activation function.
- x_t is the input at timestep t .
- h_{t-1} is the hidden state of the previous timestep.
- W and U are weight matrices and b are bias terms.

The update gate z_t determines how much of the previous hidden state is retained, while the reset gate r_t controls how much of the previous hidden state influences the new memory content $\tilde{h}_t$.

Graph Recurrent Neural Networks (GRNNs) [81] integrate the strengths of both GCNs and recurrent neural networks, such as GRUs. In a GRNN, the traditional matrix multiplication between learnable weights and inputs—or hidden states—in a GRU is replaced with a graph convolution operation. This fusion allows the network to capture both spatial relationships in the graph structure and temporal dependencies across sequences.

Definition 21: Graph Recurrent Neural Network

Let $GCN(A, X)$ denote the GCN operation using the normalized random-walk laplacian matrix. Then, a Graph Recurrent Neural Network, supported by a GRU, is defined as follows:

$$\begin{aligned} z_t &= \sigma(GCN(x_t, A) + GCN(h_{t-1}, A) + b_z) \\ r_t &= \sigma(GCN(x_t, A) + GCN(h_{t-1}, A) + b_r) \\ \tilde{h}_t &= \tanh(GCN(x_t, A) + r_t \circ GCN(h_{t-1}, A) + b_z) \\ h_t &= z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t \end{aligned}$$

Where:

- σ is the sigmoid activation function.
- x_t is the input at timestep t .
- h_{t-1} is the hidden state of the previous timestep.
- b are bias terms.

By integrating the graph convolution operation directly into the inherent recurrence of the

GRU, GRNNs can simultaneously process spatial and temporal information at each timestep. This integrated approach offers an advantage over a sequential application of GCN followed by a GRU, as it allows the network to more effectively learn spatio-temporal dependencies in the input data. Additionally, by applying the GCN operation to the hidden state of the GRU, the state is made to inherently depend on the process model. In this context, the process model is represented by the normalized random-walk Laplacian matrix. However, it is important to note that the GRNN is tailored for graphs with a fixed structure, given that the underlying process model remains constant.

After applying the GRNN operation, the resulting final hidden state of the GRU is represented as a matrix with dimensions $\mathbb{R}^{|\mathcal{P}| \times H}$, where $\mathcal{P}$ denotes the set of places in the Petri net process model, and H is the hidden size of the GRNN. Given our objective to predict the next activity in an ongoing process, it becomes imperative to assign a unique label to the entire graph, aligning with a graph-level task. This requirement calls for a “readout” operation, which condenses the matrix output from the graph convolution into a single vector, thereby projecting the diverse outputs of the graph into a summarized representation.

The *readout* operation fulfills this role by reducing the graph outputs along the node dimension, producing a vector of dimension $\mathbb{R}^H$. There are several readout methods available, such as maximum, average, sum, or the attention-driven weighted average. In TACO, we have opted for the maximum operation, owing to its simplicity and suitability for classification tasks [116]. The formal definition of this readout operation is as follows:

Definition 22: Maximum Readout

Given a matrix h resulting from a graph convolution operation, the maximum readout operation is defined as:

$$h_G = \max(h_v) \mid v \in G$$

Here, h_G represents the summarized representation of the graph G . The function $\max$ indicates the element-wise max-pooling operation, and h_v is the learned hidden state for node v .

4.2 Encoding

In TACO, we differentiate between two types of features: those specific to each node of the place graph and those inherent to each event of the prefix. The former encapsulates

the spatio-temporal interactions between the prefix and the process model, forming the *node feature matrix*. In contrast, the latter conveys information directly extracted from the event log, constituting an attribute feature matrix.

Definition 23: Node feature matrix

Let L be the maximum length of a trace in the event log, $\mathcal{P}$ the set of places from a Petri net process model, $\mathcal{Q}$ be the set of features that represent a node which include the activated place and transitions fired for a given activity, and $|\cdot|$ be the cardinality operator. Then, a node feature matrix is a matrix of dimensions $\mathbb{R}^{L \times |\mathcal{P}| \times |\mathcal{Q}|}$ that captures the interactions of all events of the prefix with the process model such that each matrix $\mathbb{R}^{|\mathcal{P}| \times |\mathcal{Q}|}$ represents the interactions of an event with the process model. If the number of events in the prefix is less than L , we apply prepadding —padding on the left side of the matrix—.

Note that the number of columns of the node feature matrix depends on the number of transitions that can be fired concurrently in the Petri net, including every possible firing of silent transitions. In our example, only one transition is activated between every observable transition, so a single column, $F2$ is present in the matrix. This number is calculated by replaying the whole log and accounting for the maximum number of possible fired transitions between two observable transitions. Note that the first column, $F1$, which indicates which place has had a token after a transition firing, has the same meaning as each row of the node feature matrix, so it could be thought of as redundant. However, we still retain it because it helps the convergence of our GRNN model.

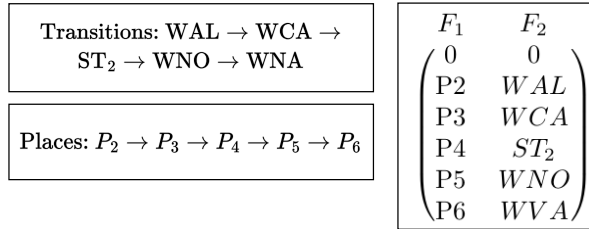


Figure 4.2: Node feature matrix after encoding the prefix shown on Table 4.1 using the process model from Figure 4.1.

Conversely, TACO also employs an attribute feature matrix:

Definition 24: Attribute feature matrix

Let L be the maximum length of a trace in the event log, R be the attribute features that can be derived from the event log, and $|\cdot|$ be the cardinality operator. Then, an attribute feature matrix is a matrix of dimensions $\mathbb{R}^{L \times |R|}$ that encodes comprehensive information about the entire event prefix.

For TACO, the attribute feature matrix encodes the following event features: (i) the event log activity generating a token in the places; (ii) the encoded time bucket since the previous event; (iii) the encoded time bucket since the prefix first event; (iv) the m attributes associated with the event. Similar to the node feature matrix, prepadding is applied when necessary.

Figure 4.3 provides an illustrative example of the attribute feature matrix for the trace 214364 from the log presented in Table 4.1. Specifically, the matrix incorporates the following features: F_1 : time elapsed since the previous event; F_2 : time elapsed since the first event; F_3 : the specific activity from the event log; F_4 : the resource associated with the event.

It is noteworthy that all features in our approach are categorical. For features that are inherently continuous, such as time, we employ a two-step conversion process to categorize them. First, we compute all potential time differences —both between consecutive events and from the start— across the training and validation sets of the event log. Then, these computed time values undergo quantile-based discretization, resulting in equal-sized intervals, which serve as categories for any given time value. The range of these intervals extends from 0 to the highest value observed in the training-validation log. Any value exceeding this range is categorized into the uppermost interval. Thus, for the matrix in Figure 4.3, the intervals for F_1 are $[0, 1000)$, $[1000, 20000)$, $[20000, 150000)$, $[150000, +\infty)$, while those for F_2 are $[0, 50000)$, $[50000, 100000)$, $[100000, 200000)$, $[200000, +\infty)$.

Instead of employing one-hot encoding for these categorical features, we opt for individual embedding matrices for each feature. Opting for individual embedding matrices ensures that the dimensionality of X remains unaffected by the number of activities in the event log, optimizing the memory usage of the neural network. Lastly, it is worth highlighting that the column values of the matrix are cumulative, derived from token replays starting from the initial event from the prefix up to the current one. This accumulation ensures retention of place-related information from all preceding events.

Activities: WAL → WCA → WNO → WVA	
Time since previous event: 0s → 21780s → 156889s → 1341s	
Time since first event: 0s → 21780s → 178669s → 180010s	
Resources: Joseph → Joseph → Enrico → Harambe	
	$\begin{matrix} F_1 & F_2 & F_3 & F_4 \\ \begin{pmatrix} 1 & 1 & \text{WAL} & \text{JOS} \\ 2 & 1 & \text{WCA} & \text{JOS} \\ 4 & 3 & \text{WNO} & \text{ENR} \\ 2 & 3 & \text{WVA} & \text{HAR} \end{pmatrix} \end{matrix}$

Figure 4.3: Attribute feature matrix after encoding the prefix from Table 4.1.

4.3 Architecture

Figure 4.4 shows the data flow of our approach. The procedure begins with an event log and its associated prefix, which form the foundation for subsequent predictions. Using this event log, we apply the Split Miner [5] to derive a collection of process models, each tailored to a unique hyperparameter configuration. From these models, the one that demonstrates optimal fitness with the validation set is selected for further processing. This selected model undergoes a transformation into a place graph, a reduced representation of the Petri net process model that makes it easier to handle in our deep learning model. Formally, a place graph can be defined as follows:

Definition 25: *Place Graph*

A place graph is a directed graph $G = (\mathcal{P}, E)$ obtained from a Petri net $N = (\mathcal{P}, \mathcal{T}, \mathcal{F})$. Here, $\mathcal{P}$ is a set of places and $E \subseteq \mathcal{P} \times \mathcal{P}$ is a set of edges such that every edge connects two places $p_1, p_2 \in \mathcal{P}$. This connection is valid if p_1 and p_2 are interconnected by a transition $t \in \mathcal{T}$ in the Petri net, denoted as $(p_1, p_2) \in E \rightarrow (p_1, t) \in \mathcal{F} \wedge (t, p_2) \in \mathcal{F}$.

In our approach, the place graph is represented as a binary adjacency matrix $A \in \mathbb{R}^{|\mathcal{P}| \times |\mathcal{P}|}$, where $\mathcal{P}$ denotes the set of places within the Petri net. Opting for a place graph offers a dual advantage. Firstly, it encapsulates the interactions between a Petri net activities, relying only on the tokens that are either consumed or produced in the places, since each event is intrinsically tied to a snapshot of the Petri net execution state. Secondly, it cuts down on memory usage by roughly half. This reduction is significant, especially considering that the matrix dimensions are directly proportional to the number of nodes in the graph. Following

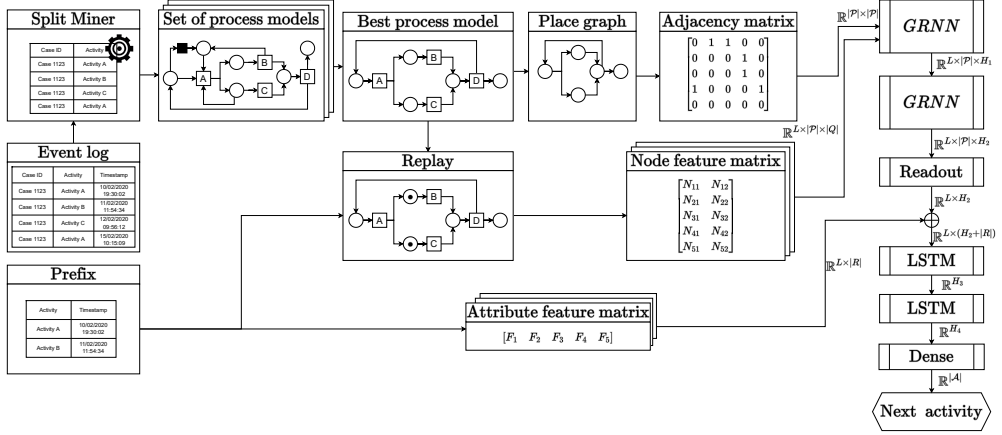


Figure 4.4: Pipeline of the data flow in TACO. The sizes of the matrices are highlighted at each step. Here, $\mathcal{P}$ is the set of places from the Petri net process model, L is the length of the longest trace of the event log, H denotes hidden size, R is the set of attribute features, $\mathcal{A}$ is the set of activities from the event log, and Q is the set of node features.

this, the place graph undergoes normalization, transforming into a random-walk normalized Laplacian, which then serves as an input for the GRNN. Given that every log prefix shares the same process model, the discovery of this model is only performed prior to the start of any prefix replay.

Following the discovery of the process model, its prefix is replayed, generating a node feature matrix that captures the activations of the places and transitions specific to that prefix. Concurrently, an attribute feature matrix is constructed, encompassing supplementary details like executed activities, temporal data related to events, and other attributes. Consequently, the GRNN receives two inputs: (i) the adjacency matrix, and (ii) the node feature matrix, which details the sequence of activations in the Petri net after replaying the whole prefix in the process model. The synthesized output of the GRNN, when combined with the attribute feature matrix, is fed into an LSTM. Subsequently, this LSTM forwards its outputs to a softmax classifier, which is tasked with predicting the next activity.

To ease the comprehension of our approach, we present Algorithm 1, which outlines the preprocessing steps to extract the process model and derive the necessary process metrics for constructing the feature matrices.

The initial step, as depicted in line 1, involves mining the best model in terms of fitness

Algorithm 1: Preprocessing algorithm for TACO**Input:** l - whole process log, l_{train} - training set of the event log**Output:** $bestModel$ - best model mined, $initialMarking$ - initial marking, N - number of places of the best process model, T_{max} - maximum fireable transitions in the process model, k - maximum trace length in the event log, max_{attr} - maximum number of attributes in the event log, $normalizedLaplacian$ - normalized laplacian of the place graph.

- 1 $bestModel, initialMarking = modelHyperparameterSearch(l_{train});$
- 2 $N = getNumberPlaces(bestModel);$
- 3 $T_{max} = getMaxTransitions(bestModel, l_{train});$
- 4 $k = getMaxTraceLength(l);$
- 5 $max_{attr} = maxAttributes(l);$
- 6 $placeGraph = buildPlaceGraph(bestModel);$
- 7 $adjacencyMatrix = getAdjMatrix(placeGraph);$
- 8 $normalizedLaplacian = normalize(adjacencyMatrix);$

using the Split Miner process discovery algorithm [5]. This is achieved through a grid search, inspired by the methodology in [102]. The search optimizes the frequency threshold ϵ , which controls the filtering process, and the parallelism detection parameter η . An important aspect of TACO is its versatility. It can use any discovery algorithm and does not mandate a perfect fitness for the process model. Consequently, if a prefix fails to replay due to trace misalignment, tokens are appended to the input places of the relevant transition, enabling it to fire. Upon deriving the optimal model from Split Miner, lines 2 to 5 compute essential metrics for constructing the node and attribute feature matrices. These metrics include the number of places, the maximum transitions, the longest trace length, and the maximum attribute count. Lastly, line 6 focuses on constructing the place graph. This graph is then represented as an adjacency matrix in line 7. The final step, depicted in line 8, transforms this matrix into a random-walk normalized Laplacian.

Algorithm 2 constructs the feature matrices needed to train the neural model. The process begins by initializing an empty node matrix with dimensions determined by the maximum number of transitions and the number of places, as shown in line 1. Subsequent lines, 2 and 3, initialize the lists that will store the sequences of node and attribute matrices. The current marking of the Petri net is set to its initial state in line 4. As the algorithm progresses from lines 5 to 10, it iterates over each event in the prefix. For every event, both the node matrix and the attribute matrix are built from lines 6 to 8.

Algorithm 2: Algorithm for building the feature matrices of TACO**Input:** outputs of Algorithm 1**Output:** *attributeMatrix* - List of attribute feature matrices, *nodeMatrixList* - List of node feature matrices

```

1  nodeMatrix = newMatrix( $N$ ,  $1 + T_{max}$ );
2  nodeMatrixList = [];
3  attributeMatrixList = [];
4  marking = initialMarking;

5  for  $e$  in  $p$  do
6      nodeMatrix, marking = buildNodeMatrix( $e$ , bestModel, nodeMatrix, marking);
7      nodeMatrixList.add(nodeMatrix);
8      attributeMatrix = buildAttributeMatrix( $e$ ,  $max_{attr}$ );
9      attributeMatrixList.add(attributeMatrix);
10 end

11 Function buildNodeMatrix( $e$ , bestModel, nodeMatrix, marking) :
12      $act_S$ ,  $act_T$ , marking = replayEvent(bestModel,  $e$ , marking);
13     nodeMatrix[ $act_N$ , 1] =  $act_N$ ;
14     for  $i = 1$ ;  $i < len(act_T)$ ;  $i = i + 1$  do
15         nodeMatrix[ $act_N$ ,  $1 + i$ ] =  $act_T[i]$ ;
16     end
17     return nodeMatrix, marking;

18 Function buildAttributeMatrix( $e$ ,  $max_{attr}$ ) :
19     attributeMatrix = newMatrix( $3 + max_{attr}$ );
20     timeSincePrev, timeSinceStart = getTimeFeatures( $e$ ["timestamp"]);
21     qTimeSincePrev = discretizeTime(timeSincePrev);
22     qTimeSinceStart = discretizeTime(timeSinceStart);
23     attributeMatrix[1] =  $e$ ["activity"];
24     attributeMatrix[2] = qTimeSincePrev;
25     attributeMatrix[3] = qTimeSinceStart;
26      $i = 1$ ;
27     for attribute in  $e$  do
28         attributeMatrix[ $3 + i$ ] =  $e$ [attribute];
29          $i += 1$ ;
30     end
31     return attributeMatrix;

```

The function defined in line 11 implements the node matrix construction, taking as input an event, the best-mined process model, the preceding node matrix, and the current Petri net marking. Within this function, line 12 replays the event on the previously mined process model using the current net marking. This action yields important information: the activated places, transitions, and the net marking after the event replay. These data are then used to update the node matrix between lines 13 to 16. The function concludes its operation by returning both the updated node matrix and the new marking, as indicated in line 17.

In parallel, the function defined in line 18 focuses on building the attribute matrix. It requires as inputs the event and the previously determined maximum number of attributes. An initial step in this function, line 19, creates an empty vector whose size is the maximum number of attributes plus three. Time features, crucial for the matrix, are computed and subsequently transformed into categorical values, as seen in lines 20 to 22. The latter part of the function, spanning lines 23 to 30, fills the matrix with data about activities, time, and attributes.

4.4 Loop overwriting

It might be tempting to merge both the attribute feature matrix and the node feature matrix into a single matrix, where each matrix element, X_{ij} , would represent the j^{th} feature from place i . This would allow the GCN model to process both node and event-level information concurrently. However, this approach presents a significant challenge: node features can be overwritten when a loop occurs within a set of places. This issue becomes particularly pronounced when the loop comprises a single activity. In such cases, the information loss is maximum, due to the limited number of places involved in this type of loops.

Figure 4.5 illustrates this problem using the replay process of trace 173712 from the *BPI-2012-W-C* event log —see Table 4.1—. The place graph is also depicted in this figure. As observed, the information in the matrix X_3 is overwritten by that in X_4 due to a loop in the activity *W_Completeren_aanvraag*. If we consider a GCN model that processes only the last replay state without fully accounting for the evolution of the trace, the only inputs would be the normalized random-walk Laplacian matrix and the X_4 feature matrix. Consequently, the resource information available in X_3 would be inaccessible to the GCN model.

A potential solution to this encoding challenge is to expand the feature matrix along the feature dimension Q up to the maximum loop size present in the event log ($\mathbb{L}$). This would account for older events occurring in the same place, resulting in a feature matrix X

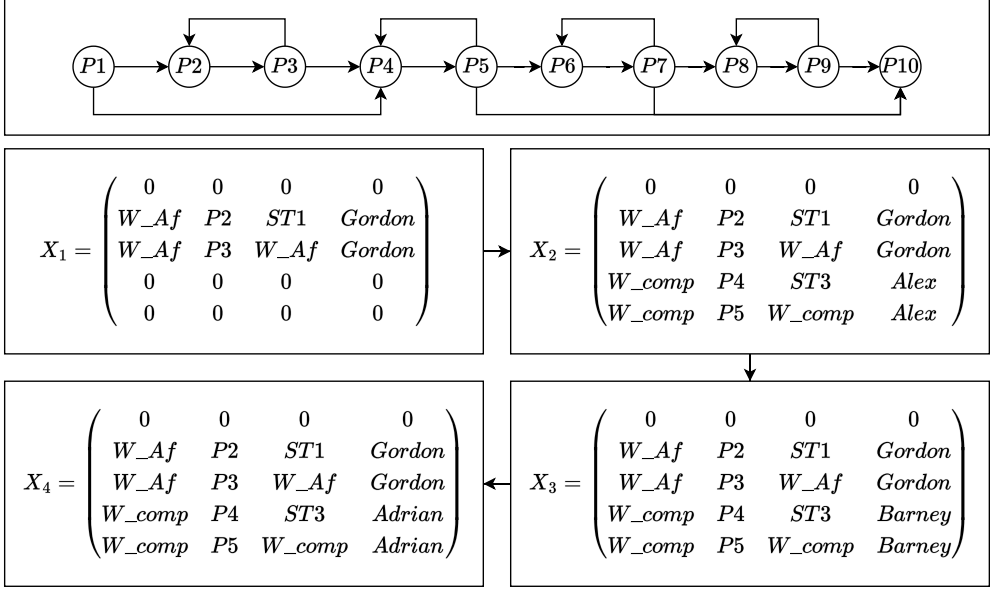


Figure 4.5: Loop overwriting problem exemplified. Encoding for places $P6$ to $P10$, temporal and attribute features omitted for the sake of clarity.

with dimensions $\mathbb{R}^{P \times (Q \cdot \mathbb{L})}$. However, some event logs contain loops of such length that this approach becomes computationally infeasible.

A straightforward solution to the loop overwriting issue would involve applying the GCN operation independently to each node matrix in the sequence $\langle X_1, \dots, X_n \rangle$, for every prefix event. Subsequently, a readout operation would be performed after each convolution operation, with the resulting information fed into an LSTM to capture temporal dependencies. However, this approach introduces another challenge: the structural properties of the input—the sequence of node feature matrices—and its temporal properties—the sequence of attribute feature matrices—are processed separately. As a result, certain spatio-temporal interactions, such as dependencies within a loop, might be overlooked. To address this problem, our approach leverages a GRNN to simultaneously process the spatial and temporal features derived from the replay of the prefix over the process model.

4.5 Experimentation

4.5.1 Experimental setup

To evaluate TACO we followed the methodology described in Chapter 3. We benchmarked our approach against 10 state-of-the-art approaches [22, 33, 45, 52, 68, 73, 99, 102, 109, 18]. However, we opted to exclude the *Nasa* and *Sepsis* event logs from our evaluation. The rationale behind this decision is that the approach from [22] relies on the presence of the "org:resource" attribute, which these logs lack. Additional experimental results are available in the Appendix B.

For all tested event logs, we fixed our approach hyperparameters. We utilized 256 hidden units for both the LSTM and the GRNN. The optimization was performed using the Adam optimizer [53] with a learning rate of 0.001, complemented by a cosine annealing warm restart scheduler [61]. Time features were quantized into 10 buckets, and the embedding dimension was set at 32. Training spanned 100 epochs, incorporating an early stopping criterion based on validation accuracy: training stops if the validation accuracy does not show improvement for 10 consecutive epochs after reaching its peak. Our approach resilience across a diverse range of hyperparameters negated the need for extensive hyperparameter tuning. All experiments were conducted on a server equipped with an Intel Xeon Gold 6326, with 64 cores, 128 GB of RAM and 1 Nvidia A100 graphics card. Our implementation was developed in Pytorch 1.8.1. For event log preprocessing, we leaned on *PM4PY* [11], and *scmamp* [21] was employed for statistical testing.

4.5.2 Results and discussion

Table 4.2 presents the evaluation results. TACO achieves the best results in 7 out of the 10 tested event logs, the second-best result in one, and the third-best in the remaining two. Notably, the most significant margins over the next best approach are observed in the *BPI-2013-I*, *BPI-2013-C-P*, and *BPI-2012-W* logs. In contrast, the performance advantage in logs like *BPI-2012*, *BPI-2012-C*, and *Env-permit* is more subtle. However, TACO performance lags in the *BPI-2012-A*, *BPI-2012-O*, and *BPI-2012-W-C* logs.

Table 4.3 delineates the rankings derived from the Bayesian Plackett-Luce model. TACO secures the top rank and exhibits the highest probability of being the best approach, a substantial lead of 24.8 percentage points over its nearest competitor. The credible intervals, represented by the 5% and 95% quantiles, along with the expected probability of each approach emerging

	BPI-2012	BPI-2012-A	BPI-2012-C	BPI-2012-O	BPI-2012-W	BPI-2012-W-C	BPI-2013-C-P	BPI-2013-I	Env-Permit	Helpdesk
Camargo	83.28	75.98	77.93	81.35	76.4	68.95	54.67	66.68	85.78	82.93
Evermann	59.33	75.82	62.38	79.42	75.37	67.53	58.83	66.78	76.19	83.66
Hinkka	86.65	81.19	80.64	87.23	84.78	70.54	63.47	74.69	84.43	83.08
Khan	42.9	74.9	47.37	66.08	60.15	52.22	43.58	51.91	83.59	79.97
Mauro	84.66	79.76	80.06	82.74	85.98	68.64	24.94	36.67	53.59	31.79
Pasqua.	83.25	74.12	74.6	78.88	81.19	68.34	47.45	46.03	86.69	83.93
Tax	85.46	79.53	80.38	82.29	85.35	69.79	64.01	70.09	85.71	84.19
Theis (w/o attr.)	82.89	65.5	75.26	78.38	86.22	80.06	59.48	59.41	86.29	78.77
Theis (w/ attr.)	80.96	65.67	75.75	76.89	86.86	83.84	54.65	51.5	85.12	79.69
Venugopal	54.69	54.88	63.75	67.87	53.9	64.8	48.44	49.62	69.58	78.7
Zararah	85.31	79.87	77.43	81.15	85.19	68.19	63.29	68.94	86.06	83.96
TACO	87.08	79.78	80.85	82.95	88.34	74.2	67.53	77.72	87.66	85.2

Table 4.2: Mean accuracy of the 5-fold cross-validation. Best, second-best, and third-best approaches are highlighted in green, orange and yellow, respectively.

	TACO	Tax	Hinkka	Zararah	Camargo	Theis (w/o)	Theis (w/)	Pasqua.	Evermann	Mauro	Venugopal	Khan
Rank	1.03	2.59	2.83	3.92	5.07	6.81	7.31	7.62	8.27	10.02	11.14	11.4
Prob. (%)	39.7	14.9	13.6	9.2	6.3	3.8	3.4	3.1	2.6	1.6	1	0.9

Table 4.3: Plackett-Luce rankings and posterior probabilities of the approaches.

as the winner, are illustrated in Figure 4.6. It is worth noting that when two approaches do not have overlapping intervals, they are statistically distinct. The non-overlapping credible interval of TACO underscores its differences in ranking compared to the other approaches.

Table 4.4 summarizes the performance differences between TACO and other state-of-the-art approaches. A positive or negative difference indicates TACO superior or inferior performance, respectively, in comparison to another method. To ensure consistency with hierarchical statistical tests, we employ the ROPE strategy, treating differences of less than 1% accuracy as negligible, effectively deeming the compared approaches equivalent. Predominantly, the differences lean favorably towards TACO. Specifically, TACO surpasses other state-of-the-art methods in 96 instances, matches them in 10, and falls short in just 4. These

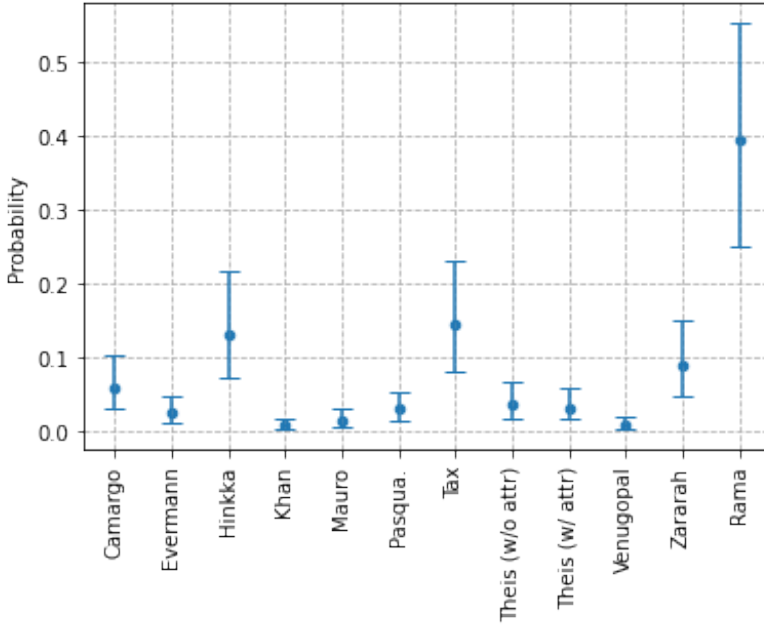


Figure 4.6: Credible intervals —5%, 95% quantiles— and the expected probability of winning for the tested approaches —TACO is denoted as “Rama” in this diagram—.

four instances pertain to the methods of Theis (w/ attr) [102] and Hinkka [45]. While Theis (w/ attr) bests TACO in *BPI-2012-W-C* by 9.64%, TACO outshines it in all other datasets, with the most pronounced margins in *BPI-2013-I* (26.22%) and *BPI-2012-A* (13.41%). Similarly, Theis (w/o attr) outperforms TACO in *BPI-2012-W-C* by 5.86%, but lags in all other datasets, especially in *BPI-2013-I* (18.31%) and *BPI-2012-A* (13.58%). In contrast, while TACO outperforms Hinkka [45] in 6 event logs, it matches the results in *BPI-2012* and *BPI-2012-C*, but is outperformed in *BPI-2012-A* and *BPI-2012-O*.

TACO typically outperforms other methods, although there are exceptions in event logs characterized by simpler process models, specifically *BPI-2012-A* and *BPI-2012-O*. In such scenarios, models based on recurrent networks are sufficient to capture the interdependencies between events. These results emphasize the idea that graph-based approaches excel in identifying intricate behavioral patterns, which are often challenging for shallower models like LSTMs. This capability enhances the prediction accuracy for the next activity, especially when the event logs exhibit loops or parallels.

	BPI-2012	BPI-2012-A	BPI-2012-C	BPI-2012-O	BPI-2012-W	BPI-2012-W-C	BPI-2013-C-P	BPI-2013-I	Env-permit	Helpdesk
Camargo	3,8	3,1	2,92	1,6	11,94	5,25	12,86	11,04	1,88	2,27
Evermann	27,75	3,26	18,47	3,53	12,97	6,67	8,7	10,94	11,47	1,54
Hinkka	0,43	-2,11	0,21	-4,28	3,56	3,66	4,06	3,03	3,23	2,12
Khan	44,18	4,18	33,48	16,87	28,19	21,98	23,95	25,81	4,07	5,23
Mauro	2,42	-0,68	0,79	0,21	2,36	5,56	42,59	41,05	34,07	53,41
Pasqua.	3,83	4,96	6,25	4,07	7,15	5,86	20,08	31,69	0,97	1,27
Tax	1,62	-0,45	0,47	0,66	2,99	4,41	3,52	7,63	1,95	1,01
Theis (w/o attr)	4,19	13,58	5,59	4,57	2,12	-5,86	8,05	18,31	1,37	6,43
Theis (w/ attr)	6,12	13,41	5,1	6,06	1,48	-9,64	12,88	26,22	2,54	5,51
Venugopal	32,39	24,2	17,1	15,08	34,44	9,4	19,09	28,1	18,08	6,5
Zarahah	1,77	-0,79	3,42	1,8	3,15	6,01	4,24	8,78	1,6	1,24

Table 4.4: Accuracy cross-validation differences of TACO against other approaches. Highlighted in bold differences greater than 1%.

For a more fine-grained understanding of the differences between the methods, we employed a hierarchical Bayesian test, as described in Chapter 3, whose results are presented in Table 4.5. We executed this test to contrast our method with the top two contenders according to the Plackett-Luce ranking, mirroring our approach in the analysis of absolute differences in accuracy. These tables report the probabilities for each dataset as well as the overall probability that one approach is superior.

The statistical tests reveal that TACO significantly outperforms the other approaches in five datasets, as shown in Table 4.5. It matches the performance in two datasets —*BPI-2012* and *BPI-2012-C*— and falls short in two others, *BPI-2012-A* and *BPI-2012-O*. In the *BPI-2013-C-P* dataset, the superiority of TACO is not statistically significant, which differs from the results presented in Table 4.4. This discrepancy arises from the pronounced variability in cross-validation results for this dataset. Consequently, the only definitive conclusion is that TACO performance is on par with Hinkka [45]. Broadly speaking, while TACO does not statistically surpass Hinkka [45], it still shows a 91% likelihood of achieving superior accuracy.

Conversely, against Tax approach [99], TACO emerges as the superior method in five datasets, as highlighted in Table 4.5. It equals Tax performance in *BPI-2012-C*, but definitive conclusions cannot be made for the remaining datasets. Notably, TACO performance is nearly

	TACO vs. Hinkka			TACO vs Tax		
	$A > H$	$A = H$	$A < H$	$A > T$	$A = T$	$A < T$
BPI-2012	0.305%	99.68%	0.01%	98.02%	1.97%	0.0136%
BPI-2012-A	0.028%	2.168%	97.80%	24.45%	72.85%	2.70%
BPI-2012-C	1.627%	99.82%	0.02%	2.34%	97.62%	0.036%
BPI-2012-O	0.072%	0.185%	99.74%	51.99%	44.55%	3.45%
BPI-2012-W	<u>100%</u>	0%	0%	99.98%	0.019%	0%
BPI-2012-W-C	<u>99.88%</u>	0%	0.111%	99.62%	0.358%	0.026%
BPI-2013-C-P	81.16%	14.54%	4.30%	99.87%	0.12%	0.012%
BPI-2013-I	<u>99.29%</u>	0.658%	0.053%	99.85%	0.137%	0.017%
Env-permit	<u>99.02%</u>	0.898%	0.08%	93.03%	6.8%	0.166%
Helpdesk	<u>99.55%</u>	0.439%	0%	54.58%	45.39%	0.031%
Overall	91%	1.2%	7.8%	99.3%	0.2%	0.6%

Table 4.5: Hierarchical bayesian tests comparing TACO (A) against Hinkka et al. (H) and Tax et al. (T). Underlined values signal statistical significance ($> 95\%$).

on par with Tax in the *Env. permit* dataset, trailing by less than 2%. For the other three datasets, the only interesting fact is that TACO performance is comparable to that of Tax. Nevertheless, TACO still commands a 99.3% probability of outperforming [99], which is statistically significant.

4.5.3 Ablation study

This section delves into the results from two distinct ablation studies. The first study harnesses the full spectrum of available information —activities, token-replay, time features, and event attributes— to individually assess each component impact on the overall performance. The second study maintains the architecture of TACO but varies the information it utilizes. To ease the experimentation and clarify the interpretation of the results, this study concentrates on evaluating the performance of TACO when it uses: (i) only transition activations and place marking; and (ii) exclusively time features and event attributes. We refrain from conducting tests focused solely on activity usage as it would make the GRNN module of TACO unnecessary.

The main goals of the first ablation study are (i) to assess the individual performances of the GRNN and LSTM; and (ii) to explore the GRNN functionality without incorporating recurrence. This essentially involves evaluating how well the GCN operator performs when limited to processing only the latest accumulated replay from the entire prefix. This evaluation

aims to quantify the performance enhancement achieved by circumventing the loop overwriting issue. Table 4.6 showcases the results, highlighting the GCN performance is consistently inferior to that of the GRNN across all logs. This performance disparity is particularly pronounced in logs with intricate process models, as underscored in Table 4.6.

Furthermore, every event log has a fair number of loops, which further degrades the performance of the GCN. When comparing the LSTM with TACO, the most significant differences are observed in the logs highlighted in bold, where the LSTM is unable to capture the interactions between the events; so the utilization of structural information from the process model is highly advantageous. However, it is worth noting that the LSTM consistently outperforms the GRNN in nearly every log. This is because the GRNN is restricted to processing the information from the token replay, whereas the LSTM has access to a broader range of information, including activities, time features, and attributes.

	GCN	GRNN	LSTM	TACO
BPI-2012	<u>53.43</u>	82.83	86.71	87.08
BPI-2012-A	<u>79.49</u>	79.49	79.77	79.78
BPI-2012-C	<u>75.12</u>	77.26	80.31	80.85
BPI-2012-O	<u>72.53</u>	80.82	82.64	82.95
BPI-2012-W	<u>31.35</u>	65.17	88.06	88.34
BPI 2012 W C.	<u>67.29</u>	68.07	73.75	74.2
BPI-2013-C-P	<u>57.56</u>	59.57	65.29	67.53
Helpdesk	<u>83.47</u>	83.75	79.99	85.2
BPI-2013-I	<u>60.44</u>	67.15	75.84	77.72
Env-permit	<u>84.25</u>	84.60	85.90	87.66

Table 4.6: Ablation study comparing the individual performance (accuracy) of the different components of TACO. Underlined values show results of the GCN model where it heavily underperforms with respect to the GRNN, while **bold** values show results where TACO heavily outperforms the LSTM model.

The results of the second ablation study are displayed in Table 4.7. This table illustrates the impact of different information components on performance, compared against the results from TACO. Each initial from the column of the table represents a different type of information component: *A* for activities, *N* for activated places, *T* for activated transitions, *I* for time features, and *R* for log attributes. As indicated in the table, the difference of performance when using only activated transitions (*T*), the information of the marking (*N*), or a combination of both is marginal for these datasets. Nonetheless, it is not guaranteed that exclusively using

the marking of the nodes or the transition activation would consistently yield this across all potential event logs. Consequently, TACO incorporates both information types to maximize its generality. Furthermore, the integration of time features (*I*) proves to be advantageous in the majority of scenarios, as shown in the underlined values of the table 4.7. This is particularly pronounced in datasets like *BPI-2012-O* and *BPI-2012-C*. The incorporation of event attributes (*R*) also enhances performance in most instances, with the most significant enhancements highlighted in bold in Table 4.7. It is worth noting that the *BPI-2012-A* dataset does not exhibit any performance boost from the addition of supplementary information. This exemplifies a simple log where the token replay data alone suffices to extract the optimal results achievable with this architecture.

	ANT	AN	AT	ANTI	ANTIR
BPI-2012	85.42	85.40	85.42	85.70	87.08
BPI-2012-A	79.49	79.49	79.49	79.78	79.78
BPI-2012-C	77.61	77.59	77.57	<u>80.21</u>	80.85
BPI-2012-O	81.14	81.18	81.16	<u>83.00</u>	82.95
BPI-2012-W	85.36	85.31	85.36	<u>86.14</u>	88.34
BPI-2012-W-C	68.46	68.49	68.56	<u>68.91</u>	74.2
BPI-2013-C-P	61.88	62.49	62.76	<u>63.42</u>	67.53
Helpdesk	84.08	83.96	83.99	<u>84.20</u>	85.20
BPI-2013-I	69.03	68.68	68.62	<u>70.22</u>	77.72
Env-permit	86.24	86.21	86.28	<u>86.00</u>	87.66

Table 4.7: Ablation study showing the performance (accuracy) of our approach using different inputs, represented by a different initial, being “A” activities, “N” activated places, “T” activated transitions, “I” time features, and “R” log attributes.

4.6 Conclusions

In this chapter, we introduced TACO, a novel approach to the next activity prediction problem. Unlike many existing methods, TACO leverages both the information contained in event log traces and the insights distilled from the process model. We developed an architecture that synergizes the structural nuances of the process model with the rich data present in the events from the event log. This was achieved by adapting the GRU model, enabling concurrent processing of the state of the Petri net using GCNs for each prefix event. As a result, our architecture learns from event log prefixes in a bidimensional manner: spatially, as represented

by the directed graph of Petri net places, and temporally, as represented by the event sequence within the prefix. A notable advantage of our approach over GCNs is its resilience against information loss during process loops. TACO adeptly captures the dynamic interplay of event replays within the process model, all while maintaining computational efficiency and simplicity.

Our evaluation is comprised of 10 real-world event logs, comparing TACO against 10 state-of-the-art methodologies. The findings underscore the consistency of TACO and its superior predictive performance relative to its counterparts. Our experiments prove the suitability of GNNs for predictive process monitoring, given their intrinsic capability to optimize the utilization of process model information. This assertion is further bolstered by our results, where TACO consistently outshines other approaches, with the exception of logs characterized by simpler process models.

CHAPTER 5

ENCODER-DECODER GRAPH-BASED ACTIVITY SUFFIX PREDICTION

Prepare for unforeseen consequences.
— G-Man, *Half Life 2*

In this chapter, we introduce ASTON (Activity Suffix predicTiOn based on eNcoder-decoder), a novel approach for activity suffix prediction that integrates knowledge from the process model. Drawing inspiration from the insights gained during the development of TACO, ASTON refines the process of learning from the model. Specifically, ASTON introduces a novel method for incorporating process model information into an encoder-decoder framework, enhances the graph encoding technique used in TACO, and introduces an innovative sampling strategy. This strategy stands out for its simplicity, efficacy, and adaptability to the diverse characteristics inherent in event logs. As we delve further into this chapter, we will uncover the details of ASTON, using a typical event log from a hospital setting to clarify its concepts and methodologies.

For the sake of simplicity, let Table 5.1 be the example event log that will be used to illustrate the concepts described in this chapter. This event log provides insights about a hospital operational workflow. It records a sequence of activities for various cases, outlining the progression of patient care from registration to various medical procedures and, ultimately, discharge. Each entry in the log not only contains details about the specific activity but also includes timestamps, offering insights into the temporal dynamics of the process. Additionally,

Table 5.1: Potential example of an event log from a hospital.

Case ID	Activity	Timestamp	Resource
Case1934	Registration	15-02-2021 08:34:20	Max
Case1934	CT scan	16-02-2021 12:35:12	Lewis
Case1934	Consultation	20-03-2021 11:23:53	Nicholas
Case1934	Blood analysis	21-03-2021 10:54:12	Sergio
Case2032	Registration	17-02-2021 12:57:25	Charles
Case2032	Blood analysis	17-02-2021 13:05:24	Carlos
Case2032	Discharge	17-02-2021 18:46:26	Fernando

the log identifies the resource or medical personnel responsible for each activity.

The contents of this chapter are extracted from the following publication:

Efrén Rama-Maneiro, Pablo Monteagudo-Lago, Juan Vidal and Manuel Lama. **Exploiting Recurrent Graph Neural Networks for Suffix Prediction in Predictive Monitoring.** abs/2211.16106.

Author's contributions:

Efrén Rama-Maneiro: Methodology, Software, Validation, Formal Analysis, Investigation, Data Curation, Writing — Original Draft.

Pablo Monteagudo-Lago: Software, Writing — Review & Editing.

Juan Vidal: Conceptualization, Resources, Writing — Review & Editing, Visualization, Supervision, Project administration, Funding acquisition, Investigation.

Manuel Lama: Conceptualization, Resources, Writing — Review & Editing, Visualization, Supervision, Project administration, Funding acquisition, Investigation.

5.1 ASTON architecture for training

ASTON leverages an encoder-decoder recurrent neural model equipped with an attention mechanism, as illustrated in 5.1. ASTON is built upon three key components:

- An encoder, which learns an effective representation of the input prefix and its interactions with the process model.
- A decoder, which predicts the suffix using the information obtained from the encoder.

- An attention mechanism, which computes the relevance of reach decoder hidden state to the input prefix, aiding prediction accuracy.

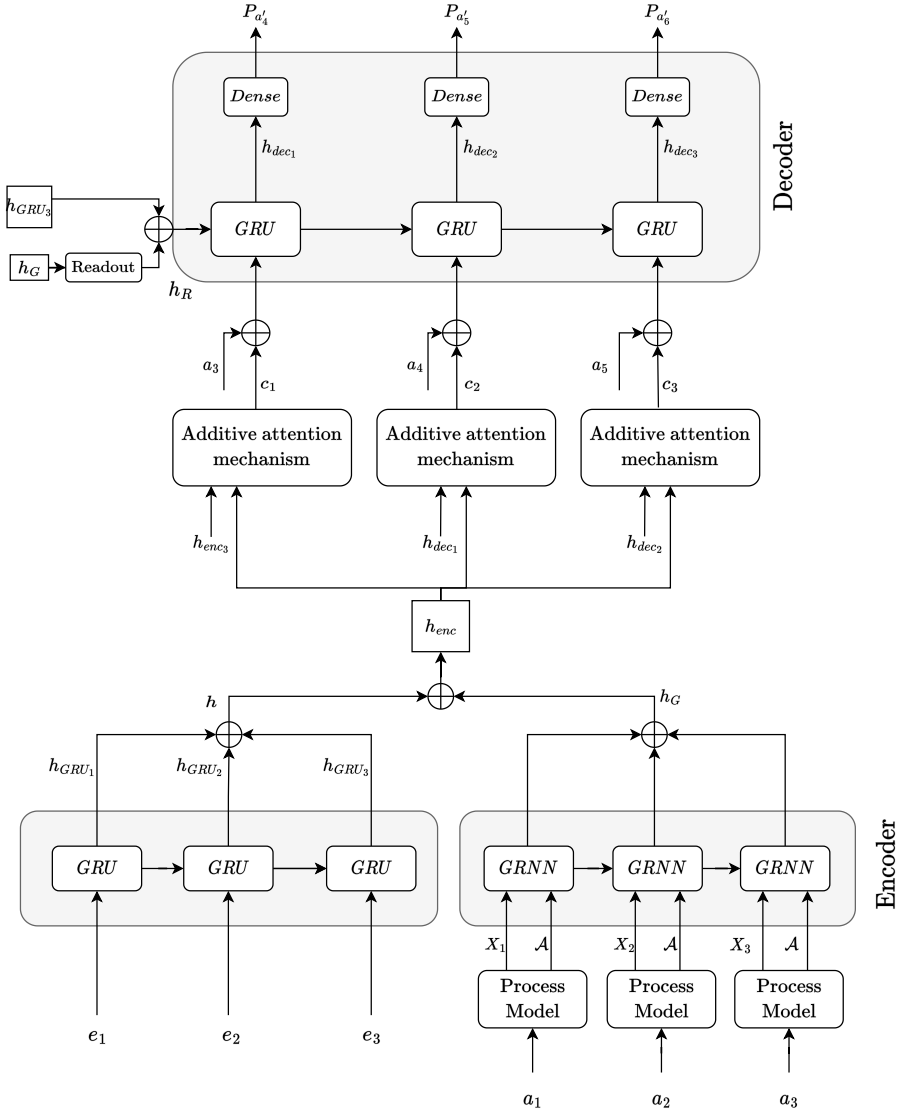


Figure 5.1: ASTON architecture during the training phase.

Diving into each module, the encoder consists of two distinct parts. The first component is a GRU [23]. It learns from the input event prefix, which contains information about activities — a_i , as shown in Figure 5.1—, the time and resources for each event (e_i), and the replay of activities (X_i) over the process model —represented as an adjacency matrix A_i —. The second part is a GRNN, which is trained using the replay of the activity prefix within the process model. The output of the decoder is a sequence of hidden states, h_{enc} , each representing an event in the prefix.

This sequence of hidden states serves two main purposes. Firstly, it generates a tensor which is used to initialize the hidden state of the decoder. This tensor comprises two different parts: one half derives from the final hidden state of the GRU, h_{GRU_3} , while the other half is derived from a readout operation that summarizes all the hidden states produced by the GRNN along the prefix. This design ensures that the decoder has direct access to both the event log information and the process model. Notably, the readout operation is performed over the full sequence of hidden states, h_G , instead of the last hidden state, so that the information since the beginning of the prefix is better propagated towards the summarized tensor. Formally, this readout operation is defined as follows:

Definition 26: Bidimensional readout

Let k be the length of the prefix, $|\mathcal{P}|$ be the number of places of the Petri net, $h_G \in \mathbb{R}^{k \times |\mathcal{P}| \times H_G}$ be the sequence of hidden states, F be the number of features, and H_G be the number of hidden features learned by the GRNN. Then, the max-pooling operation over the spatial and temporal dimensions, $h_R \in \mathbb{R}^{H_G}$, is:

$$h_R = \max_{(1 \leq i \leq k, 1 \leq j \leq |\mathcal{P}|)} X^{(i,j,h)} \quad \forall h \in H_G$$

Secondly, it provides the sequence of hidden states, h_{enc} , used by the decoder and the attention mechanism for the activity suffix prediction task. This sequence, similar to the tensor used to initialize the decoder, comprises half of its information from the event log and half from the process model. Therefore, the attention mechanism will also have direct access to the information of the process model. Also, the information used to initialize the hidden state of the decoder will be balanced with respect to the information it gets from the context vector created by the attention mechanism.

The decoder, implemented as a GRU, outputs a sequence of hidden states, h_{dec_n} , which are subsequently projected into the probability distribution of the next activity of the suffix, Pa'_n . The input at each timestep is a concatenation of the context vector —computed using the attention mechanism— and the preceding activity. To speed up the convergence and training of the predictive model, *teacher forcing* [38] is employed, where each ground-truth activity a_i is used at each timestep instead of the predictions a'_i of the decoder. To predict the first event of the suffix, the decoder is fed with the last event of the prefix — a_3 in Figure 5.1—.

The attention mechanism then computes a context vector c_i that represents the relevance of each hidden state of the decoder, h_{dec_n} , with respect to the full sequence of hidden states of the encoder, h_{enc_n} . For the first event of the decoder, as there are no prior decoder hidden states available, the last state of the encoder is used — h_{enc_3} in Figure 5.1—. Most notably, since the GRNN outputs are directly feeded to the decoder, the attention mechanism has direct access to the information from the process model, instead of being diluted if we also used the GRU to preprocess it. In this paper, we employ an additive attention mechanism [6], whose details will be further elaborated in subsequent sections.

Regarding the training procedure of ASTON, as the activity suffix prediction can be viewed as a multiclass prediction problem of a sequence of activities, ASTON is trained using the cross-entropy loss function in each predicted activity of the suffix:

Definition 27: Activity suffix loss function

Let a be a ground-truth activity index, $P_{a'}$ represent its predicted probability, and $|AC|$ be the number of different activities in the event log, and the cross-entropy loss for next activity prediction be:

$$l(a, a') = -\log\left(\frac{\exp(P_{a'})}{\sum_{i=1}^{|AC|} \exp(P_i)}\right)$$

Then, the loss function for activity suffix prediction is:

$$L(\text{tl}^k(\pi_{AC}(\sigma)), \text{tl}^k(P_{\pi_{AC}(\sigma')})) = \sum_{k=1}^K l(a_k, P_{a'_k})$$

Where $\text{tl}^k(\pi_{AC}(\sigma))$ is the ground-truth activity suffix, $\text{tl}^k(P_{\pi_{AC}(\sigma')})$ is the sequence of probability distributions of the suffix, K is the length of the suffix, a_k is the activity index

k , and $P_{a'_k}$ is the probability distribution of the predicted activity k .

Note that both $\text{tl}^k(\pi_{AC}(\sigma))$ and $\text{tl}^k(P_{\pi_{AC}(\sigma')})$ are padded with end-of-case tokens to the maximum length of the log trace, so they are guaranteed to have the same length during training. However, these pad values are not masked in the loss function, since in the experimentation predictions were more stable in this way.

5.2 Encoding

ASTON encodes each event as a tensor — e_i in Figure 5.1— resulting from the concatenation of the activity, with the time features computed from the timestamps, and the resources assigned to each activity. Activities are first encoded into integer identifiers and then into embeddings, represented by a vector assigned to each activity. Embeddings are used because they are less memory intensive than a one-hot encoding representation in event logs with a high number of activities or resources.

On the other hand, the timestamps of the event cannot be used directly as inputs to the neural network, so the following time features are computed from these timestamps: the time elapsed between one event and its predecessor; the time elapsed between one event and the beginning of the trace; the time elapsed between one event and the midnight of the same day; and the month, the day of the week, and the hour of the day in which the event has happened. In addition, the time elapsed between one event and its predecessor and the time elapsed between one event and the start of the trace are standardized using the z-score technique [88]. This standardization is applied after a logarithmic transformation since there exists a high degree of variability in the time features; so applying the logarithm amplifies the differences in the values near zero, which in turn helps to capture the dependencies among the activities [22]. The other time-related variables are normalized using the maximum value for each type of variable. For example, the time since midnight is normalized by dividing it by 86 400, i.e., the number of seconds in a day.

Figure 5.2 shows an example of the tensor that encodes the information of a prefix. In this tensor, columns represent the prefix events, while rows represent the features computed for each prefix event. Thus, for a given column, i.e., for an event, the feature F_1 refers to the activity; F_2 indicates, if available, the resource; and F_3, F_4, F_5, F_6, F_7 , and F_8 are the normalized time-related features calculated from the event timestamp.

In terms of the inputs to the GRNN module — X_i and $\mathcal{A}$ in Figure 5.1—, we construct an input feature matrix of dimensions $\mathbb{R}^{L \times |\mathcal{P}| \times k}$, where L stands for the length of the prefix, $|\mathcal{P}|$

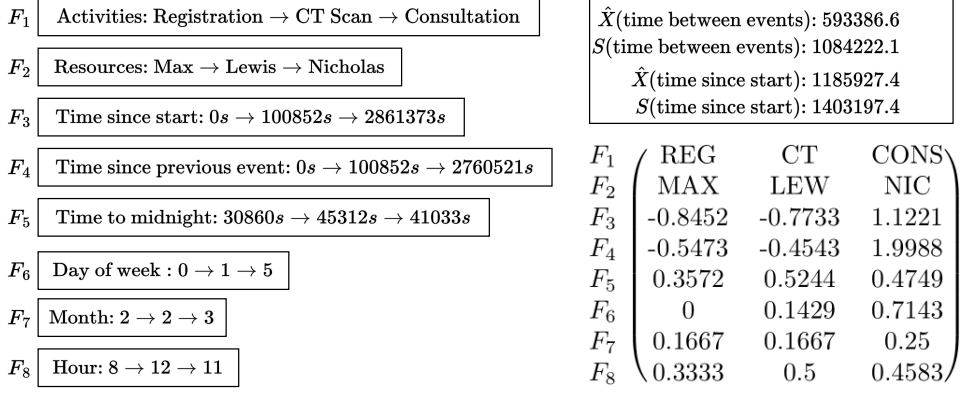


Figure 5.2: Encoding of the prefix corresponding to the three first events of the trace *Case1934* from the log shown in Table 5.1. For the sake of clarity, categorical features are shown before the embedding phase.

is the number of places in the Petri net process model, and k is an hyperparameter represents maximum number of recent place movements considered at the same time in the matrix. In particular, for each event, we record the k most recent movements through the places in the Petri net. Each of these movements is represented in a separate column of the feature matrix, with each column corresponding to an individual place.

This approach is an improvement over TACO, where all information from previous events is accumulated on the same column of the feature matrix. This accumulation can lead to a situation where, from the GCN point of view, many places seem activated simultaneously in the feature matrix, particularly towards the end of the prefix, when, actually, only a single token movement has been performed when a event is replayed. This could pose a challenge for the GCN part of the GRNN in identifying which places where most recently activated for a given event. By enforcing sparsity on the feature matrix and recording on the same matrix previous token activations, we enhance the propagation through the GCN while helping the GRNN to preserve the model information along the full prefix.

By focusing, on each timestep, in encoding the k most recent place movements, aims to facilitate the convergence of the recurrent part of the GRNN. This, in turn, could help the network pay more attention to its GCN part and the extraction of characteristics from the process model. Furthermore, by recording each place activation individually in each column,

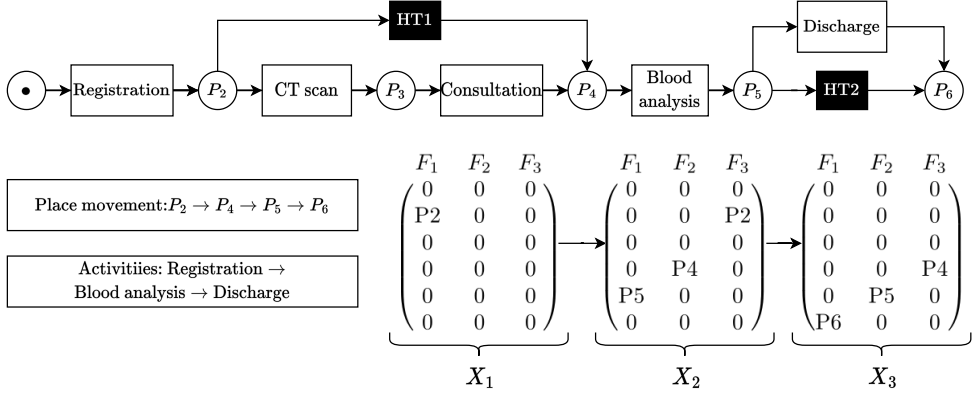


Figure 5.3: Encoding of the prefix corresponding to the trace *Case2032* from the log shown in Table 1.1 under its process model.

the GCN can explicitly account for the temporal aspect of token movement within the Petri net, alleviating the computation from the RNN part of the GRNN.

Let Figure 5.3 be an example of this encoding. This figure displays the feature matrix generated after replaying the trace identified as *Case2032*, using $k = 3$. The feature matrix corresponding to each event in the trace forms a sequence, denoted as X_1, X_2, X_3 , which would serve as the input to the GRNN. It is noteworthy that even if the activation of P_2 is absent from X_3 , the information about P_2 is still propagated through the prefix. This is because the GRNN processes the entire prefix, rather than just the last activations. In our experiments, we empirically found that setting $n = 5$ offers a good balance, performing well without significantly impacting computational efficiency or neural model complexity.

5.3 Attention mechanism

As explained in previous sections, ASTON uses an attention mechanism to tie together the encoder and the decoder modules of the architecture. The motivation behind this is that using only the last hidden state of the encoder might not be enough to obtain an accurate prediction because we would be assuming that this vector perfectly synthesizes the execution context of the prefix. Therefore, an attention mechanism is introduced to help leverage the complete information from the encoder.

Generally, an attention mechanism is performed in three steps to calculate a *context vector*, c_t , which summarizes the entire information of the encoder with respect to the current state of the decoder. These steps are: *applying a scoring function*, *calculating the attention scores* for each hidden state of the encoder, and *computing the context vector*.

Definition 28: Attention Score Function (Alignment function)

Given a hidden state of the decoder, $h_{dec_{t-1}}$, and the sequence of hidden states of the encoder, h_{enc} , the attention score used in ASTON measures the alignment between the encoder and decoder as follows:

$$s_{t,i} = w^T \tanh(W_1 h_{dec_{t-1}} + W_2 h_{enc_i})$$

Where W_1 , W_2 , and w^T are learnable parameters; $h_{dec_{t-1}}$ is the previous hidden state of the decoder; h_{enc_i} is the i_{th} hidden state of the encoder; and $s_{t,i}$ is the calculated score.

Definition 29: Attention Scores

Given a set of alignment scores, s_t , the attention scores, $\{\alpha_{t,1}, \dots, \alpha_{t,n}\}$ are calculated as follows:

$$\alpha_{t,i} = \text{softmax}(s_{t,i}) = \frac{\exp(s_{t,i})}{\sum_{j=1}^{h_{enc}} \exp(s_{t,j})}$$

Where h_{enc} is the sequence of hidden encoder states.

Definition 30: Context vector

Given the attention scores α and the encoder hidden states h_{enc} , the context vector c_t is calculated as follows:

$$c_t = \sum_{i=1}^E \alpha_{t,i} h_{enc_i}$$

Finally, once the context vector c_t is calculated, the decoder input I_t at each timestep t is computed as follows:

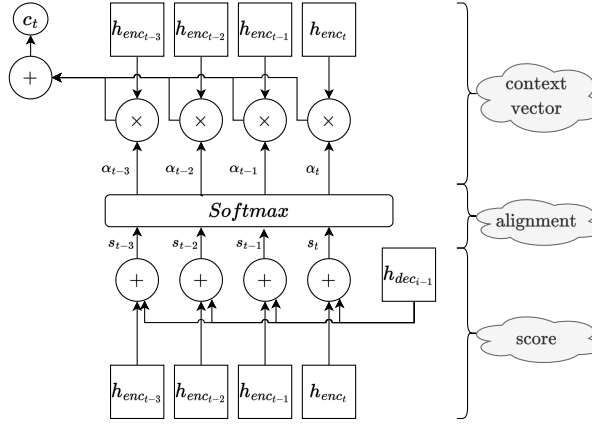


Figure 5.4: Attention mechanism of ASTON when processing an input prefix. The circles with a + inside represent the addition operation, and the circles with a $\times$ inside represent the product between two vectors. *tanh* function omitted for clarity.

$$I_t = c_t \oplus a'_{t-1}$$

Where $\oplus$ is the concatenation operator, and a'_{t-1} is the predicted activity for the previous timestep—or the last activity of the prefix if $t = 0$ —.

Figure 5.4 visually outlines the process of creating the context vector, c_t , as used in the attention mechanism in ASTON. This process involves the hidden states of the encoder ($h_{enc_{t-3}}, \dots, h_{enc_t}$), the last state of the decoder ($h_{dec_{t-1}}$), and the context vector (c_t) itself. The lower part of the picture depicts the first state of the attention mechanism, where scores are computed by adding the hidden states of the encoder and the last state of the decoder in a pairwise manner. The middle part of the figure then demonstrates how these calculated scores are aligned by applying a softmax operation. Lastly, the top part of the figure shows the production of the final context vector, c_t , achieved by multiplying each of the hidden states of the encoder with the aligned scores and summing the results.

5.4 ASTON architecture for inference

Once the network has been trained, the output at each step in the sequence is actually a probability distribution over the set of activities, $\text{tl}^k(P_{\pi_A(\sigma')})$, from the decoder, from which the predicted activity a'_t must be selected during an inference phase. In this inference phase,

the decoder cannot use the ground truth activities as inputs, unlike during the training phase, as Figure 5.5 shows. Instead, it substitutes these with the predicted activities, $\text{tl}^k(\pi_A(\sigma'))$, except for the first activity, which is the last activity of the input prefix. This introduces an *exposure bias* that comes from the difference between the training inputs —ground truth— and the inference inputs —predicted outputs—. This makes the selection of the next activity of the suffix during inference critically important.

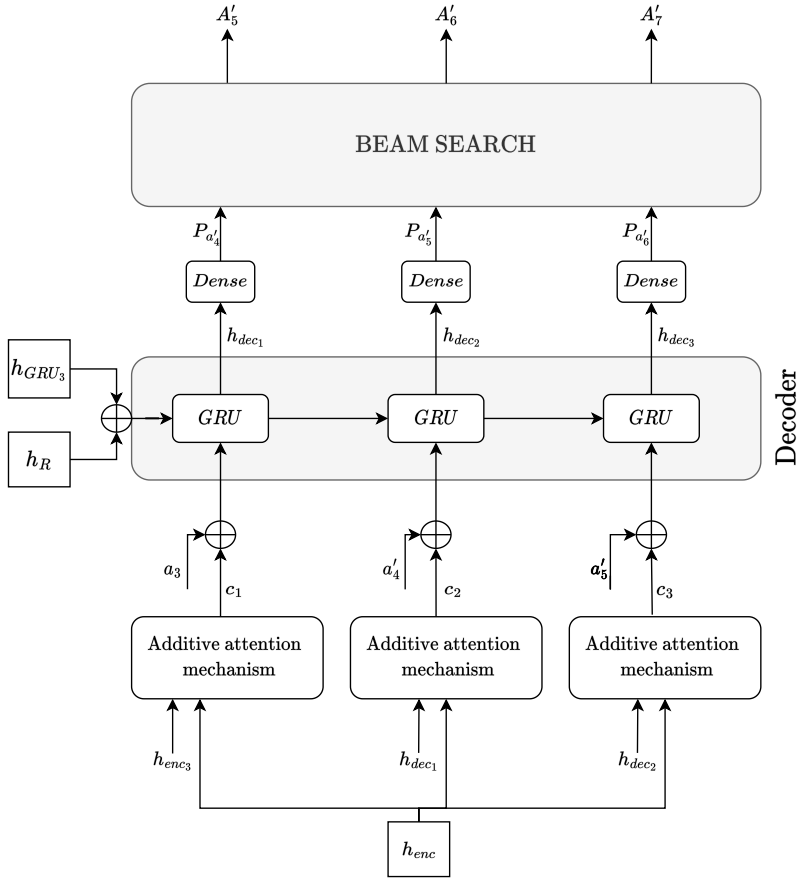


Figure 5.5: ASTON architecture for the inference phase, where the encoder is omitted for clarity.

Furthermore, it is worth mentioning that Figure 5.6 demonstrates the possible output probabilities that would be generated by the decoder after predicting the next two activities

for a given prefix. In predictive process monitoring, two strategies for activity selection are commonly used:

- *Argmax* [22, 99], where the most likely activity is always selected at each prediction step. For instance, following this strategy in Figure 5.6, the activities “A” and “A” would be selected, producing the suffix “AA” with a joint probability of $0.5 \cdot 0.6 = 0.3$. Typically, *argmax* works reliably in event logs with short traces while underperforming in logs with long ones.
- *Random sampling* [22, 33], where the next activity is randomly sampled from the probability distribution of the next activities. Returning to the example in Figure 5.6, we might select “B” as the next activity, followed by “A” as the final prediction, yielding the predicted suffix “BA” with a joint conditional probability of $0.1 \cdot 0.8 = 0.08$. This strategy, unlike *argmax*, performs better in logs with long traces while underperforming in logs with short ones.

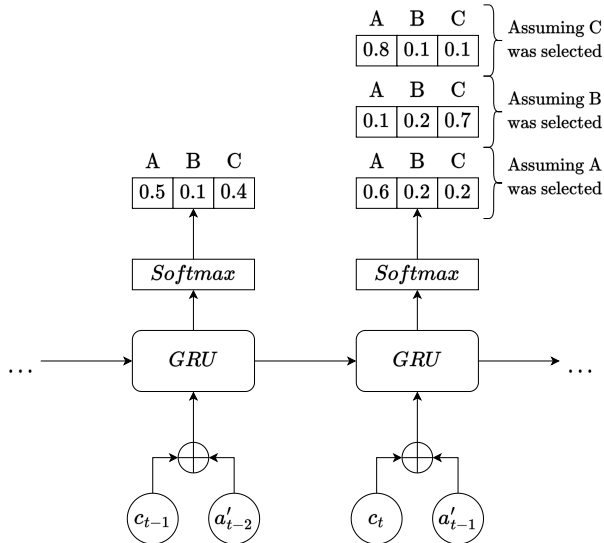


Figure 5.6: Output probabilities for the next two activities of a given prefix, where the numbers in the top of the figure represent the probability of a given activity at a certain timestep. Depending on which activity is selected on timestep $t-2$ —A, B, or C—, three different probability distributions for timestep $t-1$ can be found out.

However, both strategies might not select the suffix with the highest overall probability (“CA”). The *argmax* strategy lacks exploration, potentially leading to the selection of suboptimal solutions or even the repeated selection the same set of activities, never stopping. On the contrary, the *random sampling* strategy can predict suboptimal suffixes due to its overly broad exploration. Both strategies only take into account the probabilities of the current prediction and do not consider the probabilities of the entire suffix.

To address these limitations, ASTON uses the beam search [10], an optimization of exhaustive search that significantly reduces both memory and time requirements while maintaining good solution quality. The time complexity of the beam search algorithm is $(O)(B_w \cdot |A| \cdot k)$, where B_w is the beam width —representing the number of alternative suffixes kept at each decision point—, $|A|$ is the number of different activities in the log, and k is the maximum length of the suffix. Thus, the beam search operates by maintaining only the B_w most promising sequences at each decision point and discarding the rest. This contrasts with the exhaustive search, which would keep all possibilities open, quickly leading to an infeasible number of alternatives as the length of the sequence increases.

As an example, let us refer to Figure 5.7, which demonstrates the operation of beam

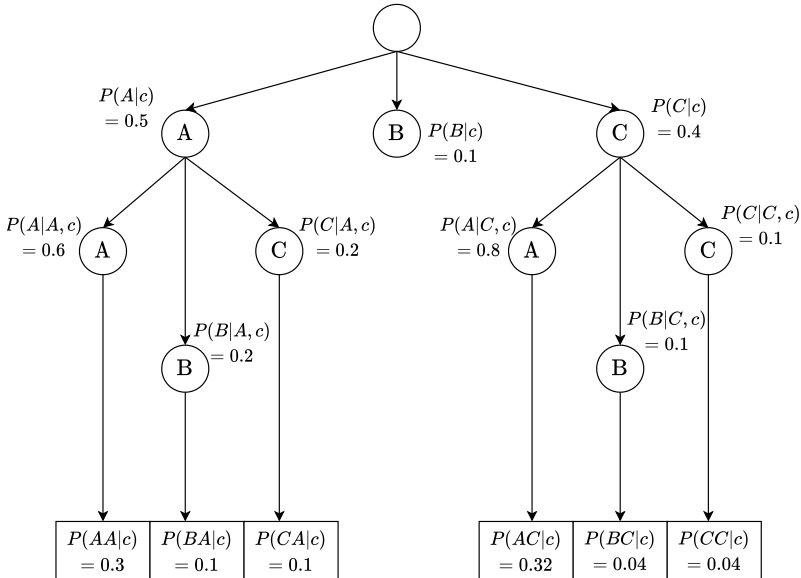


Figure 5.7: Probability tree for a suffix of two activities and three possible predictions at each step.

search given the output probabilities shown in Figure 5.6. For a beam width of $B_w = 2$, only the two most probable alternatives are kept at each step. In the first step, the activities “A” and “C” are chosen for further exploration since they have the two highest probabilities—0.5 and 0.4, respectively— while activity “B” is discarded as it has the lowest probability—0.1—. The chosen activities are expanded in the order of their probability; in this case “A” is expanded before “C”. In the second step, the search stops as it has reached the end of the suffix prediction. The suffix with the highest joint-conditioned probability is then selected from the remaining activities. In this example, that would be “CA”. In particular, the joint-conditioned probability of a partial suffix is formally computed during the beam search through the following expression:

Definition 31: Beam search scoring function

Let $hd^k(\sigma)$ be the prefix that conditions the prediction; a'_i be the i_{th} activity, with $i \in \{0, \dots, N - 1\}$, then the beam search scoring function is defined as follows:

$$P(a'_i, \dots, a'_0 | hd^k(\sigma)) = \prod_{j=0}^i P(a'_j | a'_{j-i}, \dots, a'_0, hd^k(\sigma))$$

Taking the logarithm on both sides and considering that the logarithm of a product is the sum of the logarithms, we get:

$$\log P(a'_i, \dots, a'_0 | hd^k(\sigma)) = \sum_{j=0}^i \log P(a'_j | a'_{j-i}, \dots, a'_0, hd^k(\sigma))$$

Note that the logarithm operation is applied to deal with the numerical instability associated with consecutive products of floating-point numbers in the interval $[0, 1]$ and because the sum operation is less computationally intensive than the product operation.

However, the previous expression tends to favor short suffixes since a negative log probability is added for each predicted event, yielding a lower score for the predicted suffix. Therefore, we have adopted the same normalization strategy as in [115] to fairly compare suffixes of different lengths, resulting in the following expression:

Definition 32: Normalized beam search scoring function

$$\log P(a'_i, \dots, a'_0 | \text{hd}^k(\sigma)) = \frac{(5+1)^\gamma}{(5+i)^\gamma} \sum_{j=0}^i \log P(a'_j | a'_{j-i}, \dots, a'_0, \text{hd}^k(\sigma))$$

Where i is the current prediction step, and γ is a factor which is set to 0.65 in our experimentation.

This suffix normalization is what distinguishes the beam search implemented in ASTON from the beam search proposed by [36] and [100]. Thus, [36] does not perform any kind of normalization to the output probabilities, whereas [100], instead of normalizing, stops the prediction if the suffix length exceeds a certain threshold related to the average trace length of the log. This stopping strategy is too restrictive since it does not allow the prediction of suffixes longer than this threshold.

5.5 Experimentation

5.5.1 Experimental setup

We evaluated ASTON against five other state-of-the-art approaches [100, 99, 33, 36, 22] under the experimental conditions delineated in Chapter 3. Notably, the approach from [22] offers two distinct operational modes, leading to two separate results. Throughout our experiments, ASTON consistently employed the same hyperparameters across all event logs as our findings indicated that its architecture remains robust against hyperparameter variations. Specifically, the neural network was trained for 150 epochs with a batch size of 64, an embedding dimensionality of 32, a 2-layer GRU with a hidden dimension of 32, and a 0.1 dropout in both the encoder and decoder. We utilized the Adam optimizer with a learning rate of 0.005. During the inference phase, we set a beam width of 5 for suffix predictions based on its empirical performance on the validation sets of the examined event logs.

For ASTON, this hyperparameter configuration remained consistent across all logs, demonstrating its robustness. Moreover the approach from [22] underwent tuning via random search, as documented in the paper, whereas the other methods were executed with the set of hyperparameters outlined in their respective source codes. Despite including the “Nasa” and “Sepsis” datasets in our evaluation, the approach from [22] could not be executed on them due

to the lack of the resource attribute. Consequently, any statistical tests involving this approach excluded results obtained from these two datasets. In particular, for the ranking statistical test, we disregarded the results from these datasets, even for the other approaches.

5.5.2 Results

Table 5.2 shows the results of the experimentation. ASTON obtains the best result in 8 out of 12 event logs, the second best result in 3 out of 12, and the third best result in 1 out of 12. These results seem to indicate that ASTON is highly consistent in both event logs with shorter and longer traces: specifically, for the 8 logs in which it obtains the best results, there exists a difference in performance —ranging from 0.0049 until 0.0924— when comparing with the second-best approach in each log; whereas for the other 4 logs, it obtains results —ranging from 0.0111 until 0.0577— that are comparable to best approach in each log —Camargo/argmax, and Taymouri—.

However, it is difficult to evaluate which is the best approach overall, since it does not only depend on the number of event logs in which each approach is the best but also on the quantitative differences between the results obtained in each event log. Thus, to rank the approaches based on their performance, a Bayesian Plackett-Luce model is applied, the results

Event log	Camargo/ argmax	Camargo/ random	Evermann	Francesco- marino	Tax	Taymouri	ASTON
BPI-2012	0.1851	0.3891	0.1986	0.1321	0.1409	0.1867	0.3947
BPI-2012-A	0.6536	0.6441	0.5847	0.2871	0.4597	0.5420	0.6371
BPI-2012-C	0.2218	0.4495	0.2693	0.0883	0.1717	0.3557	0.4544
BPI-2012-O	0.6845	0.6042	0.5544	0.4591	0.4972	0.5150	0.6268
BPI-2012-W	0.1941	0.3110	0.2800	0.1410	0.0975	0.2493	0.3993
BPI-2012 W C.	0.0501	0.3191	0.3372	0.1126	0.0789	0.3201	0.3622
BPI-2013 C. P.	0.6641	0.5337	0.6416	0.5276	0.5824	0.4358	0.6932
BPI-2013-I	0.2607	0.5294	0.4730	0.3607	0.3336	0.2452	0.6218
Env-permit	0.8440	0.7595	0.5713	0.3924	0.8163	0.4271	0.8469
Helpdesk	0.9110	0.8524	0.8354	0.4619	0.8695	0.6298	0.8999
Nasa	-	-	0.1218	0.0842	0.2320	0.4844	0.4529
Sepsis	-	-	0.2693	0.0742	0.1158	0.3217	0.3966

Table 5.2: ASTON Results of the 5-fold cross-validation for the activity suffix prediction task in terms of the Damerau-Levenshtein similarity. The best, second-best, and third-best results are highlighted in green, orange, and yellow, respectively.

	ASTON	Camargo/ random	Evermann	Camargo/ argmax	Taymouri	Tax	Francesco- marino
Rank	1.02	2.33	2.99	3.93	5.25	5.57	6.92
Prob. (%)	50.88	17.35	12.72	8.50	4.78	4.10	1.67

Table 5.3: Plackett-Luce rankings and posterior probabilities of the tested approaches against ASTON.

of which are shown in Table 5.3. ASTON obtains the best overall ranking and the highest probability of being the best approach —50.88%—, with a difference of 33.53 percentage points over the second approach —Camargo/random—, with 17.35%. Furthermore, Figure 5.8 shows the credible intervals —quantiles 5% and 95%— and the expected probability of being the best approach. In this figure, ASTON does not overlap any other approach, showing that its results are statistically different from those of the other tested approaches in terms of overall ranking. Furthermore, no approach clearly emerges as the second-best in terms of ranking, as each has credible intervals that overlap with at least one other approach.

To better quantify the differences between ASTON and the second and third-best approaches of the Plackett-Luce ranking —Camargo/random and Evermann— a hierarchical Bayesian test has been performed, as described in Chapter 3. In this test, it is considered that a

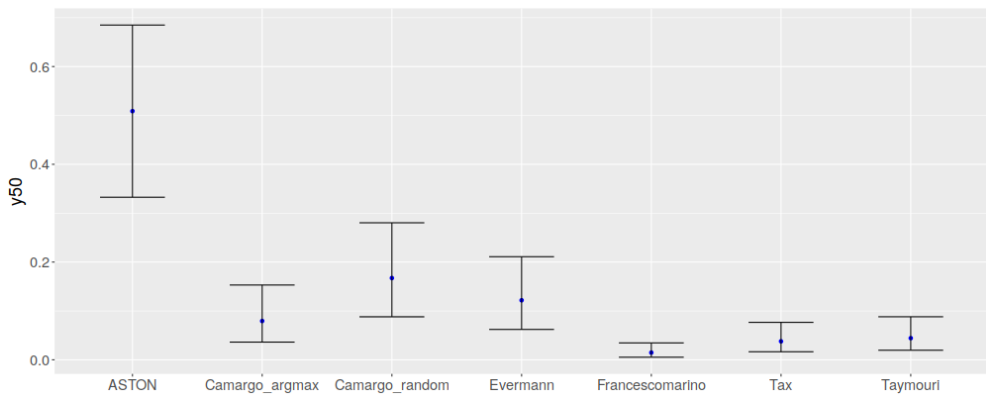


Figure 5.8: Credible intervals and the expected probability of winning for the tested approaches against ASTON.

Event log	$A > C$	$A = C$	$A < C$
BPI-2012	48.70%	2.64%	48.65%
BPI-2012 A	15.05%	7.20%	77.75%
BPI-2012-C	29.71%	6.35%	69.66%
BPI-2012-O	89.90%	0.35%	9.74%
BPI-2012-W	<u>97.82%</u>	0.43%	1.74%
BPI-2012-W-C	<u>99.26%</u>	0.05%	0.68%
BPI-2013-C-P	<u>99.27%</u>	0.15%	0.58%
BPI-2013-I	<u>99.98%</u>	0.02%	0%
Env-permit	<u>99.89%</u>	0.02%	0.09%
Helpdesk	<u>99.99%</u>	0%	0.01%
Overall	<u>99.4%</u>	0%	0.6%

Table 5.4: Hierarchical Bayesian tests per dataset: ASTON (A) vs Camargo/random (C). Underlined values show statistical significance.

probability greater than 95% indicates statistical significance. The results of this hierarchical Bayesian test are shown in Table 5.4 and 5.5:

- Table 5.4 shows that ASTON significantly outperforms Camargo/random in six event logs—*BPI-2012-W*, *BPI-2012-W-C*, *BPI-2013-C-P*, *BPI-2013-I*, *Env-permit*, and *Helpdesk*—, being close to be significantly better in *BPI-2012-O*.—with 89.90% of probability—. For the other three event logs, there is a high degree of uncertainty to make any conclusion: Camargo/random does not significantly outperform ASTON in any event log, being close in *BPI-2012-A* and *BPI-2012-C*—with 77.75% and 69.66% of probability, respectively—. These results indicate that ASTON outperforms Camargo/random in event logs that have shorter traces, whereas it does not underperform Camargo/random when traces are longer, showing that ASTON is overall more robust to the underlying characteristics of the event log. In general, ASTON significantly outperforms the approach from Camargo/random, by a 99.4%.
- Table 5.5 shows that ASTON significantly outperforms the Evermann approach for every event log, except for *BPI-2012-W-C*, in which ASTON is close to achieve statistical significance with respect to Evermann—87.51% of probability—. Therefore, an encoder-decoder learning approach with an attention mechanism is an effective method for the activity suffix prediction due to less accumulation of errors during inference.

Event log	$A > E$	$A = E$	$A < E$
BPI-2012	<u>99.98%</u>	0%	0.02%
BPI-2012-A	<u>99.98%</u>	0%	0.02%
BPI-2012-C	<u>100%</u>	0%	0%
BPI-2012-O	<u>99.98%</u>	0%	0.02%
BPI-2012-W	<u>99.92%</u>	0.03%	0.05%
BPI-2012-W-C	87.51%	1.04%	11.44%
BPI-2013-C-P	<u>98.48%</u>	0.22%	1.3%
BPI-2013-I	<u>100%</u>	0%	0%
Env-permit	<u>100%</u>	0%	0%
Helpdesk	<u>99.99%</u>	0%	0.01%
Nasa	<u>100%</u>	0%	0%
Sepsis	<u>99.98%</u>	0%	0.02%
Overall	<u>100%</u>	0%	0%

Table 5.5: Hierarchical Bayesian tests for ASTON (A) vs Evermann (E). Underlined values show statistical significance.

Considering all the logs, ASTON consistently outperforms Evermann with a 100% probability.

5.5.3 Inference strategy comparison

We have compared the performance of ASTON using different inference strategies: selecting the most likely activity —*argmax*—, randomly sampling the next activity following the probability distribution —*random*—, using an unnormalized beam search strategy —*beam*—, and normalizing the beam search log probabilities according to the length of the predicted suffix —*beam norm.*—. The evaluation encompasses both the length of the generated suffixes, providing insights into how the sampling strategy performs concerning the tested event log. Tables 5.6 and 5.7 show the Damerau-Levenshtein edit distance performance and the average predicted suffix length for each strategy, respectively.

The results indicate that the *argmax* strategy tends to have a good performance in logs with shorter traces, such as *BPI-2012-A*, *BPI-2012-O*, *BPI-2013-C-P*, *Env-Permit*, *Helpdesk*, with an average difference of 0.06. In contrast, the *random* strategy performs better in logs with longer traces like *BPI-2012*, *BPI-2012-C*, *BPI-2012-W*, *BPI-2012-W-C*, *Nasa*, *Sepsis*, having an average difference of 0.1158 in these logs. Furthermore, Table 5.7 reveals that the *argmax* strategy predicts significantly longer suffixes than the ground truth for logs with longer traces.

This rigidity of *argmax* is counterbalanced by the flexibility of the *random* strategy, which tends not to always select the same activity and has the potential to complete suffix predictions by introducing the end-of-case token.

The *unnormalized beam* strategy generally shows consistent performance across all the logs, outperforming *argmax* in logs with longer traces and rivaling the *random* strategy. In shorter trace logs, it matches the performance of *argmax* while surpassing the *random* strategy. However, the unnormalized beam strategy predicts suffixes much shorter than the ground truth for several logs with longer traces —Table 5.7—, namely *BPI-2012*, *BPI-2012-W*, *BPI-2012-W-C*, and *Sepsis*, which hampers its overall performance in these logs.

Lastly, the *normalized beam* strategy significantly outperforms the *unnormalized beam* strategy. The length normalization of the log probabilities in the beam search alleviates the issue of predicting excessively short suffixes and enhances the overall quality of the activity suffix predictions. As shown in Table 5.6, this strategy offers the most consistent performance across all tested event logs, surpassing all other strategies except for *BPI-2013-C-P* and *Helpdesk*, where *argmax* and *beam* strategies, respectively, perform better. Additionally, the *normalized beam* strategy outperforms the *unnormalized beam* strategy in all tested event logs in terms of average predicted suffix length (Table 5.7). This demonstrates that the normalization strategy effectively mitigates the issue of predicting shorter suffixes.

Event log	Argmax	Random	Beam	Beam norm.
BPI-2012	0.1767	0.3841	0.3774	0.3947
BPI-2012-A	0.5767	0.5660	0.6362	0.6371
BPI-2012-C.	0.2340	0.4204	0.4201	0.4544
BPI-2012-O	0.6240	0.5503	0.6254	0.6268
BPI-2012-W	0.1514	0.3609	0.3502	0.3993
BPI-2012-W-C	0.1083	0.3074	0.2844	0.3622
BPI-2013-C-P	0.6961	0.6301	0.6909	0.6932
Helpdesk	0.8998	0.8331	0.9000	0.8999
Sepsis	0.3921	0.3724	0.3456	0.3966
BPI-2013-I	0.5395	0.5852	0.6119	0.6218
Env-permit	0.8470	0.7655	0.8468	0.8469
Nasa	0.4475	0.4298	0.4210	0.4529

Table 5.6: Normalized Damerau-Levenshtein performance of ASTON under several inference strategies.

Event log	Truth	Argmax	Random	Beam	Beam norm.
BPI-2012	19.44	141.12	16.51	4.93	19.08
BPI-2012-A	2.21	2.70	2.36	1.32	1.34
BPI-2012-C	10.90	72.18	9.93	3.36	5.55
BPI-2012-O	3.42	2.44	3.34	2.31	2.38
BPI-2012-W	15.55	123.90	12.81	3.20	10.43
BPI-2012-W-C	6.77	63.85	5.95	0.98	4.87
BPI-2013-C-P	2.86	1.26	2.24	1.11	1.14
Helpdesk	1.98	1.64	2.08	1.63	1.64
Sepsis	11.11	18.26	9.86	4.02	6.49
BPI-2013-I	7.18	28.47	6.22	2.76	3.18
Env-permit	2.88	2.49	2.87	2.47	2.47
Nasa	14.40	13.62	13.86	9.36	14.14

Table 5.7: Average predicted suffix length for each inference strategy compared to the ground truth average suffix length.

5.6 Conclusions

This chapter presented ASTON, a predictive monitoring approach that focuses on predicting the next activity suffix given a running case. Unlike most state-of-the-art approaches, ASTON addresses this problem using an encoder-decoder architecture with an attention mechanism. Thus, our model is trained to predict the whole suffix, accumulating fewer errors during inference and focusing only on predicting those elements that are relevant to the task at hand, which, in this paper, are the activities of the event log. Our neural model is enriched with information from the process model, which helps during the learning phase of the neural network. In addition, our beam search strategy successfully balances the exploration and exploitation of candidate suffix selection during the post-training phase. Moreover, the proposed normalization for the beam search improves the performance in logs with long traces, while not degrading the performance in logs with shorter traces.

We evaluated ASTON using 12 real-life event logs against 6 state-of-the-art approaches. The results underscore that ASTON significantly outperforms all other considered approaches from a statistical perspective, securing the top result in 8 event logs, the second-best in 3, and the third-best in one. Importantly, ASTON exhibits robustness to the underlying characteristics of the event log, delivering consistent performance across a wide variety of process logs.

CHAPTER 6

REINFORCEMENT LEARNING-BASED ACTIVITY SUFFIX PREDICTION

All knowledge has its price

— Hermaeus Mora, *The Elder Scrolls V: Skyrim*

In the previous chapter, we underscored the critical importance of the sampling strategy employed in the inference phase for activity suffix prediction. This assertion is consistent with recent findings in Natural Language Processing [71], which highlight the decoding strategy as a key determinant of model efficacy. It is also worth noting that the effectiveness of a strategy can be task-specific, underscoring the lack of a universally optimal solution. In this regard, our previous chapter focused on strategies commonly used in predictive monitoring, such as argmax, random, and beam search. Although we introduced an improved strategy, the normalized beam search, which consistently performed well across various event logs, our experimentation did not extend to alternative strategies such as top-k sampling [34] or nucleus sampling [47]. These sampling strategies could potentially provide alternative solutions, although their suitability may vary due to our requirement for exact matching with the ground truth suffix, in contrast to the more flexible, multifaceted and ambiguous nature of language.

Considering this, it becomes increasingly evident that the choice of a sampling strategy, along with its hyperparameters, essentially introduces another hyperparameter in the predictive model itself. The goal of this chapter is to develop a methodology that allows the sampling strategy to be learned during the training phase. Such an approach would not only ease the

inference phase but also reduce the need for manual tuning of hyperparameters like beam width and the length normalization factor.

In this chapter, we argue that the disparities in performance across the multiple sampling strategies stem from the training methodologies of neural networks. Traditional approaches, apart from those based on GANs, rely on Maximum Likelihood Estimation (MLE) and are trained to maximize the likelihood of the next token given the previous tokens. However, it is not synonymous with maximizing the likelihood of the entire sequence, which is our primary goal. To address this discrepancy, we propose a novel approach for activity suffix prediction grounded in Reinforcement Learning (RL). RL offers a framework that eases the training of a model to maximize the reward of the entire suffix sequence, rather than individual tokens. This aligns with our hypothesis that the ideal sampling strategy can be discerned during the training phase of the model, simplifying the neural network design and ensuring that the most probable activity is consistently selected.

Formally, our learning objective can be defined as follows:

Definition 33: *Optimal sampling strategy approximation*

Let $p'(a_i|a_1, \dots, a_{i-1})$ be the predicted probability distribution of the next activity a_i by a given predictive model trained with MLE, and λ^* be the optimal but potentially intractable or computationally expensive sampling strategy for activity suffix prediction from the universe of all possible sampling strategies $\Lambda = \lambda_1, \dots, \lambda_n$ such that it would give the optimal suffix prediction given a MLE trained model. Then the objective of the optimal sampling strategy approximation is to find a probability distribution $p^*(a_i|a_1, \dots, a_{i-1})$ such that:

$$\arg \max(p^*(a_i|a_1, \dots, a_{i-1})) \approx \lambda^*(p'(a_i|a_1, \dots, a_{i-1}))$$

In the previous expression, the operator λ^* returns the activity that maximizes the prediction of the whole suffix, and it does not have to be the most probable activity from the probability distribution p' . Therefore, the suffix predicted by selecting at each timestep according to $p^*(a_i|a_1, \dots, a_{i-1})$ would be equivalent to applying the optimal sampling strategy, λ^* , to the probability distribution learned through MLE. In this Ph.D. dissertation, RL serves as a computational framework for approximating λ^* . In the following sections, we describe how RL aids in approximating this ideal behavior.

6.1 Reinforcement learning

RL [96] is a type of machine learning where an agent learns how to interact with an environment to perform a certain task, where the environment represents the context or situation in which the agent operates. At each timestep, the agent observes the state of the environment, chooses an action to perform based on its current policy, and then receives feedback from the environment in the form of a reward and the next state. The sequence of states, actions, and rewards from the start to the termination is called an *episode*. In each episode, the agent starts from an initial state and continues to interact with the environment until it reaches a terminal state, signaling the end of the episode. The goal of the agent is to learn a policy, which is a mapping from states to actions, that maximizes the expected cumulative discounted reward over many episodes. Formally, RL can be described as follows:

Definition 34: Reinforcement Learning

Let (S, A, P, R, γ) be a tuple where:

- S represents the state space, which is the set of all possible states the agent can be in.
- A is the action space, denoting all possible actions the agent can take.
- $P : S \times A \times S \rightarrow [0, 1]$ is the transition probability, indicating the likelihood of moving from one state to another given an action.
- $R : S \times A \times S \rightarrow \mathbb{R}$ represents the reward function, which provides a scalar feedback signal when transitioning from one state to another.
- $\gamma \in [0, 1]$ is the discount factor, which measures the degree of importance of both immediate and future rewards. A value closer to 0 makes the agent short-sighted, giving more importance to immediate rewards, while a value closer to 1 makes it prioritize long-term rewards.

The primary objective in RL is to find an optimal policy $\Pi^* : S \rightarrow A$ that maximizes the expected cumulative discounted reward:

$$\max_{\Pi} E_{\Pi} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \right]$$

where $s_t \in S$ represents the state at time t , and $a_t \in A$ denotes the action taken at time t .

One of the challenges in RL is the balance between exploration and exploitation. While traditional methods like Q-learning [111] and SARSA [82] have been effective, they sometimes suffer from slow convergence and can be sensitive to hyperparameters.

In this Ph.D dissertation, we adopted the Proximal Policy Optimization [86] (PPO) algorithm, which optimizes the policy directly by using gradient ascent, in such a way that it ensures that the policy is not too different from the previous one. Thus, it provides stability to the training procedure by avoiding too large updates. PPO makes use of the concept of *advantage function* [96], which can be defined as follows:

Definition 35: Advantage function

The advantage function measures how much better or worse is taking a specific action a in a given state s compared to the average action in that state. It can be calculated as the difference between the Q-value [72] of the action a , $Q(s, a)$, and the value function for the state s , $V(s)$:

$$A(s, a) = Q(s, a) - V(s)$$

In this case the value function V measures the expected cumulative reward starting from state s and following a particular policy Π , whereas the Q-function represents the expected cumulative reward starting from state s , and taking action a , and the following policy Π .

In practice, the advantage function $A(s, a)$ is computationally expensive to compute directly. Therefore, it is often approximated using the Temporal Difference (TD) error. In this approximation, the TD error serves as an estimate for $A(s, a)$ and is calculated based on the current state s , the action a , the received reward r , and the next state s' .

Definition 36: Temporal Difference error

Let s be the current state of the agent, a the action taken, r the reward received, and s' the next state. Then, the TD error, TD_{error} is used to approximate the advantage function A as follows:

$$A(s, a) \approx TD_{\text{error}} = r + \gamma V(s') - V(s)$$

where γ is the discount factor.

PPO belongs to a family of RL algorithms called *Actor-Critic models*. In this type of algorithms, the agent is composed of two different neural networks that are trained simultaneously: the actor, which proposes the next action to take given the current state, and the critic, which estimates the expected return or value from a given state under the current policy. In particular, these models can be defined as follows:

Definition 37: Actor-Critic models

An actor-critic model is composed of two different components:

- **Actor:** given a state $s \in \mathcal{S}$, the actor, denoted as $\Pi(a|s; \Theta_{\Pi})$, outputs a probability distribution over the action space $\mathcal{A}$. The policy of the actor is parametrized by Θ_{Π} and aims to select actions that maximize the expected rewards. In a deep learning model, Θ_{Π} would correspond to the weights of the neural network that is responsible for the policy.
- **Critic:** the critic, denoted as $V(s|\Theta_V)$, estimates the expected return of being in state s . It is parametrized by Θ_V and provides a baseline against which the actions of the actor can be evaluated. In a deep learning model, Θ_V would correspond to the weights of the neural network that is responsible for estimating the state value.

An actor-critic model proceeds as follows in order to find an optimal policy Π^* that maximizes the expected cumulative reward:

- The agent observes the current state s and proposes an action a based on its current policy.

- The agent takes action a , transitions to a new state s' , and receives a reward r .
- The critic estimates the value of the states s and s' , and approximates the advantage function $A(s, a)$ using the temporal difference error.
- Both the actor and the critic are updated using this advantage. The actor aims to maximize the expected reward by adjusting its policy in the direction of the actions with higher advantages, while the critic updates its value estimates to better predict future rewards.

Taking this into account, PPO can be formally described as:

Definition 38: Proximal Policy Optimization

Let $(S, A, P, R, \gamma, \Pi_\theta)$ be a tuple, where $\Pi_\theta : S \rightarrow A$ is a policy parametrized by θ . Then, PPO aims to find an optimal policy that maximizes the following objective function:

$$L(\theta) = \hat{E}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right]$$

where:

- $\hat{E}_t$ represents the empirical expectation over the data collected at time step t . In this context, it is calculated as the average value of the objective function based on the set of observed samples at time step t . The empirical expectation serves as an approximation of the true expected value, particularly useful when the true distribution is unknown or not easily tractable.
- $r_t(\theta) = \frac{\Pi_\theta(a_t|s_t)}{\Pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the ratio of the probability of action a_t under the current policy to the probability of the action under the old policy.
- $\hat{A}_t$ is an estimator of the advantage function at time step t .
- The operator $\text{clip}(x, a, b)$ bounds or clips x between a and b .

The main caveat of PPO is its ability to maintain a balance between exploration and exploitation without the need for intricate hyperparameter tuning. By using the clipping mechanism on the ratio of new to old probabilities, $\text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)$, PPO ensures that policy updates remain relatively small, leading to more stable and consistent learning.

6.2 Architecture

In this section, we describe our RL-based approach for activity suffix prediction, namely the “**Deep Reinforcement Learning Predictor**” or “**DOGE**”. Within DOGE, the event log serves as the environment, each episode consists of a sequence of states, actions and rewards corresponding to predict an activity suffix. Every activity predicted by the agent represents a timestep within that episode.

Figure 6.1 shows the dataflow of DOGE during training. Here, we distinguish between two types of dataflows: the ones that are executed once per episode —filled lines— and the ones that are executed for every action taken by the agent —dotted lines—. As shown in the aforementioned figure, the operations of suffix sampling and edit distance calculation are done once per episode, while the policy network is executed for every action taken by the agent and the predicted suffix is accordingly updated. In our implementation, the policy network is parametrized by a feed-forward neural network.

Figure 6.2 shows how the observation matrix is created for the policy network in the case of the example log from Table 5.1, where the dotted lines indicate operations performed for each timestep of the agent. This matrix consists of four different pieces of information obtained from the event prefix sampled at the beginning of the episode: the sequence of

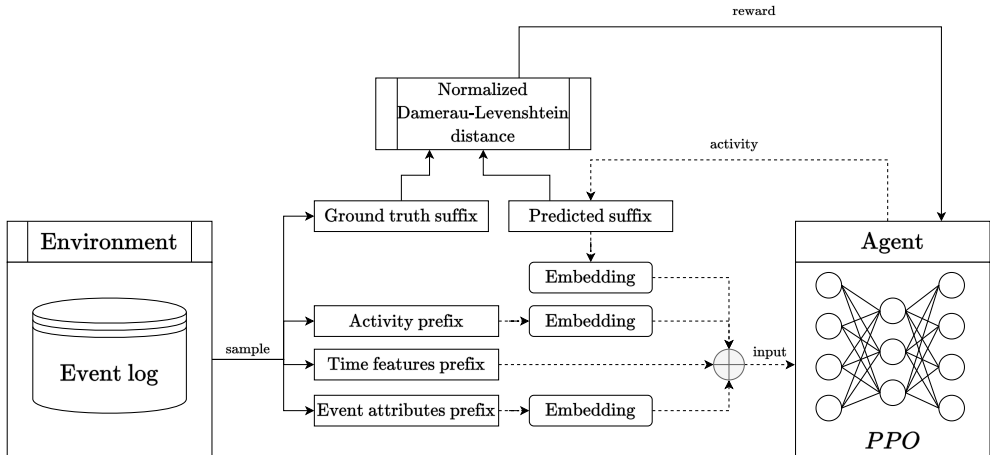


Figure 6.1: Pipeline of DOGE. Dotted lines depict dataflow from each timestep, while filled lines represent dataflow executed once per episode.

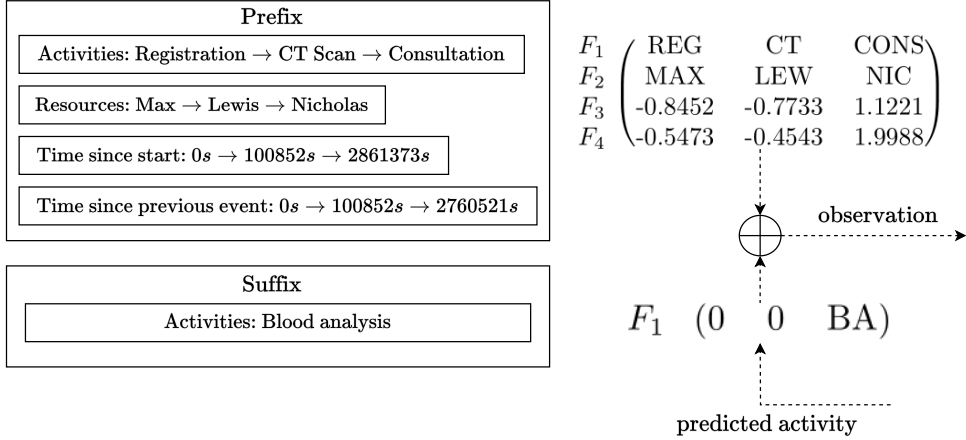


Figure 6.2: Example of the observation encoding procedure. $\oplus$ denotes the concatenation operator. Details of the calculus of the time features can be found in Section 5.2.

activities; the sequence of times elapsed since the start of the case; the sequence of times between two consecutive events; and, for each attribute, its corresponding sequence of values. Each sequence of attributes and the sequence of activities are embedded using distinct and separate embedding matrices, while the time features are simply normalized using z-score normalization. Furthermore, this information from the prefix is concatenated with the sequence of predicted activities by the agent. At each timestep —activity prediction—, this sequence is updated according to the suffix predicted so far by appending the predicted activity to the end of the list, so the concatenation of the prefix with the suffix must be performed at each timestep. Thus, by separating the event prefix and the predicted suffix into different matrices, we ensure that the policy network does not “forget” the current prefix, and we also emulate the encoder-decoder scheme of ASTON.

Algorithm 3 shows the training procedure of DOGE. The process starts with the initialization of the parameters of the policy network (line 1). Three key variables are configured in line 2: the current number of activities predicted by the agent (*curr_timesteps*), the best edit distance similarity achieved on the validation set (*best_dl*), and the count of validation iterations without any improvement. Contrary to standard practices in supervised learning, the training loop in DOGE is determined by a fixed number of timesteps rather than epochs (line 3). This approach aligns with the *dynamic* nature of the learning procedure in reinforcement

learning, where the agent continually interacts with the environment at each timestep. In contrast, supervised learning generally operates in a *static* manner, iterating over the entire dataset for each epoch. This dynamic aspect is further evidenced by the sampling of a prefix (P) and its corresponding ground truth suffix (S_{gt}) from the environment in line 4. Each new sampling effectively resets the environment and initiates a new episode in the learning process.

Lines 5-6 initialize three essential variables: the predicted suffix (S_{pred}), a tensor (T) containing tuples of the input and the corresponding action taken at each timestep. The main training loop spans from lines 6 to 26. Line 6 shows the stop condition for the agent predictions: either the end-of-case token (e_{end}) is predicted or the sum of the length of the sampled prefix ($|P|$) and the length of the predicted suffix ($|S_{pred}|$) is greater than the maximum trace length of the event log $|L_{max}|$.

The input tensor for the policy network is created in line 7. This tensor concatenates the embedded activity sequence of the prefix ($\pi_{AC}(P)$), the temporal features of the prefix ($\pi_{TI}(P)$), the sequence the attributes from the prefix ($\pi_{D_1}(P) \oplus \dots \oplus \pi_{D_n}(P)$), and the activities predicted by the agent up to that point (S_{pred}). In particular, S_{pred} is the only dynamic component updated with each prediction, making DOGE operation similar to ASTON in terms of “encoding-decoding” its inputs and outputs. In line 8, the policy network stochastically samples an action a_t from its probability distribution $\Pi(I_t, \Theta)$. This action is then appended to the predicted suffix S_{pred} (line 9), the tuple (I_t, a_t) is added to the trajectory tensor T (line 10), and the timestep counter increments by one (line 11). The inherent stochasticity in the sampling facilitates the agent exploration of potential suffixes during training.

Lines 12 to 25 detail the validation procedure, which is triggered whenever the total number of timesteps is divisible by the `EVAL_FREQ` variable. Contrary to conventional RL practices, the agent is evaluated on the entire validation set, which is different from the training environment (line 13), instead of only on a subset of the environment. Another noteworthy distinction lies in the mode of action selection during this phase. Instead of relying on the stochastic action selection employed during training, the agent operates deterministically in the validation stage, selecting the most probable action always according to the current policy. This deterministic or “greedy” action selection ensures that the predicted suffix is always the same for any given prefix. This procedure is applied in both validation and testing phases.

After the validation step, the algorithm assesses the performance of the current policy against the best observed performance on the validation set. If the current policy achieves a better normalized Damerau-Levenshtein (NDL) distance value (val_dl) than the best value

Algorithm 3: Training procedure of DOGE

Input: L_{train} - Training event log, L_{val} - Validation event log, L_{max} - Maximum trace length
 L_{max} , Policy network Π with parameters Θ , γ - Discount factor, MAX_TIMESTEPS -
Maximum number of timesteps to train, PATIENCE - Number of evaluations without
improvement before early stopping, EVAL_FREQ - Frequency of evaluations

Output: Trained policy network Π

```

1  init( $\pi, \Theta$ );
2  curr_timesteps, best_dl, eval_without_improving = 0, 0, 0;
3  while curr_timesteps < MAX_TIMESTEPS do
4       $P, S_{gt} = \text{sample\_prefix}(L)$ ;
5       $S_{pred}, T, \text{action} = \emptyset, \emptyset, \emptyset$ ;
6      while action  $\neq e_{end}$  and ( $|S_{pred}| + |P|$ ) <  $L_{max}$  do
7           $I_t = \pi_{AC}(P) \oplus \pi_{TI}(P) \oplus \pi_{D_1}(P) \cdots \oplus \pi_{D_n}(P) \oplus S_{pred}$ ;
8           $a_t = \text{sample}(\Pi(I_t, \Theta))$ ;
9          append( $S_{pred}, a_t$ );
10         append( $T, (I_t, a_t)$ );
11         curr_timesteps += 1;
12         if curr_timesteps % EVAL_FREQ == 0 then
13             val_dl = evaluate( $\pi, L_{val}$ );
14             if val_dl > best_dl then
15                 best_dl = val_dl;
16                 eval_without_improving = 0;
17                 save( $\Pi, \Theta$ )
18             end
19             else
20                 eval_without_improving += 1;
21             end
22             if eval_without_improving > PATIENCE then
23                 return best( $\Pi, \Theta$ );
24             end
25         end
26     end
27      $r = \text{NDL}(S_{pred}, S_{gt})$ ;
28     foreach ( $I_t, a_t$ ) in  $T$  do
29          $A_t = \text{compute\_advantage}(\Pi(I_t, \Theta), r)$ ;
30         update_policy( $\Pi(I_t, \Theta), A_t$ );
31     end
32 end

```

previously observed ($best_dl$, in line 14), the algorithm updates the best observed value, resets the counter tracking the number of evaluations without improvement, and saves the current policy parameters, ensuring that the best policy is retained (lines 15 to 17). However, if there is

no improvement in the NDL distance value, the counter tracking evaluations without improvement is incremented (lines 19-20). If this counter exceeds a predefined patience threshold (line 19), the algorithm terminates early, returning the best observed policy parameters. This early stopping mechanism ensures computational efficiency and prevents potential overfitting.

After the validation and potential early stopping checks, the algorithm returns to the main training loop. Once an episode —i.e., a prediction of a suffix for a given prefix— is complete, the agent computes the reward based on the NDL distance between the predicted and the ground truth suffixes (line 27). This reward serves as a feedback mechanism, guiding the learning of the agent. For each timestep within the episode, the algorithm computes the advantage of the taken action using the observed reward (lines 28-29). This advantage measure captures the relative benefit of the chosen action compared to the average action in that state. The policy is then updated using this advantage (line 30), guiding the agent towards actions that are expected to yield better rewards in the future. This loop continues until the agent has processed every input-action pair in the trajectory.

Finally, note that the choice of the NDL distance as the reward function empowers DOGE with some advantages:

- *Parallel branch consideration:* the inherent nature of the NDL distance allows it to account for potential parallel branches in the business process, ensuring a more holistic evaluation of the predicted suffix.
- *Gradient stabilization:* as the NDL distance is normalized to lie between 0 and 1, it can potentially stabilize gradient updates, a crucial aspect when employing PPO for training.
- *Avoiding premature termination:* by opting not to reward the agent at every timestep using the NDL distance, we prevent the agent from being incentivized to continually predict any activity other than the end-of-case token. This design choice ensures that the agent does not prematurely accumulate rewards.
- *Promotion of long-term strategy:* rewarding only at the conclusion of an episode promotes a long-term strategic outlook in the agent. This approach discourages the agent from myopically focusing on the immediate correctness of the current activity prediction which would be the same strategy employed by MLE.

6.3 Experimentation

6.3.1 Experimental setup and implementation details

We have compared DOGE with the same approaches as those in Chapter 5 [99, 22, 36, 100, 33], plus ASTON, using the same methodology described in Chapter 3. In the case of [22], the approach can not be executed in the event logs *Sepsis* and *Nasa* due to them not having resources, which are mandatory to run the approach. Again, we support all our experiments with the bayesian statistical test methodology highlighted in Chapter 3.

For DOGE, the choice of library to use for the implementation of the policy network is an important decision. Indeed, some RL studies have highlighted that code-level optimizations—augmentations of the algorithm that are present in the code but not described in the paper—have a crucial impact on the final behavior of the agent [32]. Due to the high number of of code-level optimizations for PPO [3], we opted for using the implementation available in Stable-Baselines3 [77]. This library offers reliable implementations that closely mirror original papers, easing our experimentation process.

Finally, another key consideration in our experiments was determining the number of timesteps for training the agent. We dynamically adjusted this number based on the characteristics of the event log. Given our validation-controlled model selection, we leaned towards longer training durations, mitigating overfitting risks and leveraging early stopping to prevent excessive training. The number of timesteps is computed as:

Definition 39: Number of RL timesteps for training

Let L_{train} be the training split of an event log L , L_{max} the maximum trace length of the event log, and $|AC|$ be the number of activities in the event log. Then, the number of timesteps to train the agent is calculated as:

$$4 \cdot |hd^{1 \cdots L_{\text{max}}}(L_{\text{train}})| \cdot L_{\text{max}} \cdot (|AC| + 1)$$

where the expression $hd^{1 \cdots L_{\text{max}}}(L_{\text{train}})$ denotes the number of possible prefixes that can be obtained from the training split of the event log.

The aforementioned expression has been determined empirically, based on preliminary experiments to ensure optimal training duration and performance. The constant factor “4” ensures that the agent is given enough timesteps to effectively learn the policy for any event

log. Furthermore, the expression takes into account multiple factors that contribute to the complexity of an event log, namely the maximum trace length and the number of different activities. By making the number of timesteps a function of these variables, our approach provides a training regimen that can be applied to any event log. This is a paradigm shift from traditional RL approaches that manually fix the number of timesteps for a given environment.

6.3.2 Results

Table 6.1 shows the results for the DOGE evaluation. DOGE obtains the best result overall in 8 of the 12 tested event logs, and the third best result in the 4 remaining ones. In general, DOGE obtains very similar results to ASTON, except in the event logs *BPI-2012*, *BPI-2012-C*, *BPI-2012-W-C*, *Nasa*, and *Sepsis*, where it outperforms ASTON by 1.14%, 1.67%, 2.07%, 4.52%, and 3.19%, respectively. However, it underperforms it in the event logs *BPI-2012-O* and *Env-permit* by a 1.34% and a 1%.

Table 6.2 depicts the overall rankings of each approach derived from the Bayesian Plackett-Luce model and the corresponding probability of achieving the best ranking, where DOGE achieves the highest overall ranking among all the evaluated methods. ASTON is the second-best approach with a probability that is close to that of DOGE, —with a difference of 6.39%—

Table 6.1: Results of the 5-fold cross-validation for the activity suffix prediction task in terms of the Damerau-Levenshtein similarity. The best, second-best, and third-best results are highlighted in green, orange, and yellow, respectively.

Event log	Camargo/ argmax	Camargo/ random	Evermann	Francesco- marino	Tax	Taymouri	ASTON	DOGE
BPI-2012	0.1851	0.3891	0.1986	0.1321	0.1409	0.1867	0.3947	0.4061
BPI-2012-A	0.6536	0.6441	0.5847	0.2871	0.4597	0.5420	0.6371	0.6379
BPI-2012-C	0.2218	0.4495	0.2693	0.0883	0.1717	0.3557	0.4544	0.4711
BPI-2012-O	0.6845	0.6042	0.5544	0.4591	0.4972	0.5150	0.6268	0.6134
BPI-2012-W	0.1941	0.3110	0.2800	0.1410	0.0975	0.2493	0.3993	0.4000
BPI-2012-W-C	0.0501	0.3191	0.3372	0.1126	0.0789	0.3201	0.3622	0.3829
BPI-2013-C-P	0.6641	0.5337	0.6416	0.5276	0.5824	0.4358	0.6932	0.6933
BPI-2013-I	0.2607	0.5294	0.4730	0.3607	0.3336	0.2452	0.6218	0.6230
Env-permit	0.8440	0.7595	0.5713	0.3924	0.8163	0.4271	0.8469	0.8369
Helpdesk	0.9110	0.8524	0.8354	0.4619	0.8695	0.6298	0.8999	0.8989
Nasa	—	—	0.1218	0.0842	0.2320	0.4844	0.4529	0.4981
Sepsis	—	—	0.2693	0.0742	0.1158	0.3217	0.3966	0.4285

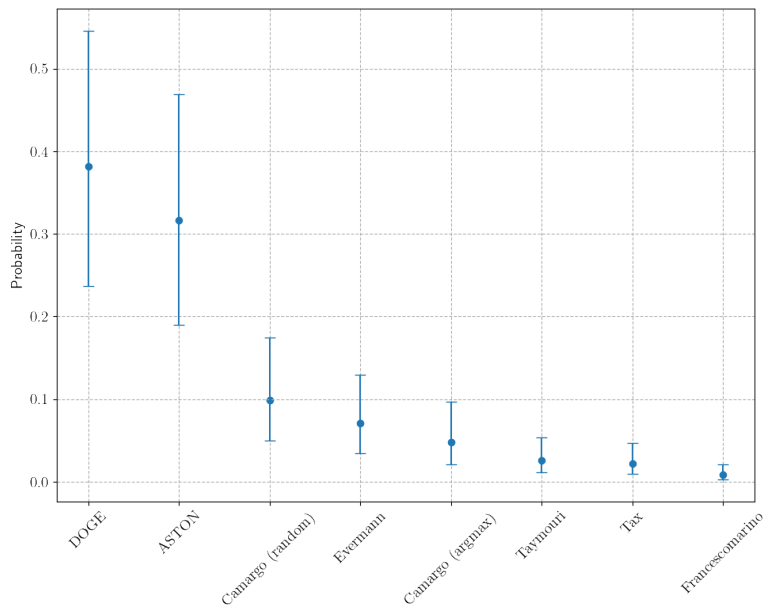


Figure 6.3: Credible intervals —quantiles 5% and 95%— and probability of winning for each of the evaluated approaches.

	DOGE	ASTON	Camargo (random)	Evermann	Camargo (argmax)	Taymouri	Tax	Francescomarino
Rank	1.35	1.67	3.33	4.01	4.89	6.25	6.57	7.92
Probability	38.51%	32.12%	10.37%	7.53%	5.19%	2.84%	2.45%	0.99%

Table 6.2: Ranking and probability of each approach to be the best in terms of ranking.

. Figure 6.3 illustrates the credible intervals for each evaluated approach, along with the winning probability. This figure also highlights that DOGE statistically outperforms all other state-of-the-art methods in terms of ranking since there are non-overlapping pairs with these approaches; except for ASTON, where DOGE shows a significant overlapping of its credible intervals.

The performance of DOGE was compared with ASTON and Camargo/random [22] using hierarchical Bayesian statistical tests. Table 6.3 shows the results of this comparison. DOGE demonstrates a clear advantage over the approach of Camargo/random across a majority of datasets. As Camargo/random relies on a random sampling, its performance is heavily dependant on the type of event log used. The strengths of DOGE become evident in its ability to adapt to the underlying characteristics of the event log. When facing ASTON, DOGE is not inferior— $55.2+40.7 = 95.9 > 95$ —, which underlines the robustness of DOGE. Furthermore, during the inference phase, the effectiveness of DOGE is apparent. DOGE consistently opts for the most probable action—the activity—, thereby eliminating the need for heuristic search methods such as beam search. In fact, DOGE also simplifies the design of the architecture as it does not require the tuning of any hyperparameters like the beam width, which is required in beam search algorithms. This not only simplifies the decision-making process but also enhances execution efficiency.

However, it is noteworthy that these results do not imply that the problem of suffix prediction is fully resolved. On half of the datasets, both DOGE and ASTON achieved less than 50% performance, and only in two cases did the performance exceed 70%. This suggests that while DOGE represents a significant advancement in the field, there is room for further research and optimization. In terms of its practical application, the advantages of DOGE extend to the inference phase. Unlike methods that require heuristic search strategies such as beam search, DOGE directly selects the most probable action, hence simplifying the architecture, reducing the need for hyperparameter tuning, and improving the efficiency during inference.

6.4 Conclusions

In this chapter, we presented DOGE, a RL-based solution for the activity suffix prediction problem. This innovative approach diverges from traditional predictive monitoring techniques, emphasizing the importance of the sampling strategy in determining the efficacy of the model. Through our exploration, we have underscored the significance of the decoding strategy employed in sequence generation, particularly in the context of predicting remaining activities. In this chapter we presented DOGE, a reinforcement learning based solution for the activity suffix prediction problem.

DOGE foundation in RL offers a new perspective on the activity suffix prediction problem. Traditional neural network training methodologies, which rely heavily on MLE, often fall

	DOGE vs ASTON			DOGE vs Camargo		
	$D > A$	$D = A$	$D < A$	$D > C$	$D = C$	$D < C$
BPI-2012	48.14%	47.71%	4.15%	<u>96.78%</u>	3.21%	0.01%
BPI-2012-A	0.60%	<u>99.22%</u>	0.18%	17.70%	75.39%	6.91%
BPI-2012-C	88.65%	11.20%	0.14%	<u>96.58%</u>	3.37%	0.06%
BPI-2012-O	0.04%	6.36%	93.60%	57.45%	41.21%	1.34%
BPI-2012-W	23.72%	66.79%	9.50%	<u>99.03%</u>	0.79%	0.18%
BPI-2012-W-C	71.20%	27.07%	1.73%	<u>99.88%</u>	0.19%	0.02%
BPI-2013-C-P	32.29%	53.79%	13.92%	<u>99.89%</u>	0.09%	0.02%
BPI-2013-I	13.99%	81.48%	4.53%	<u>99.94%</u>	0.04%	0.01%
Env-permit	1.12%	61.44%	37.44%	<u>99.70%</u>	0.25%	0.05%
Helpdesk	0.13%	<u>99.77%</u>	0.11%	<u>99.98%</u>	0.01%	0.01%
Nasa	92.59%	6.86%	0.54%	-	-	-
Sepsis	<u>95.52%</u>	4.27%	0.21%	-	-	-
Overall	55.2%	40.7%	4.1%	<u>99.5%</u>	0%	0.5%

Table 6.3: Hierarchical bayesian tests comparing DOGE (D) against ASTON. (A) and Camargo (random) (C) (22). Statistical significant results (greater than 95%) are underlined.

short in maximizing the likelihood of the entire sequence. RL, on the other hand, provides a framework that naturally prioritizes the reward of the entire suffix sequence over individual tokens. This aligns with our objective of simplifying neural network design and the sampling strategy used.

Furthermore, the use of the Proximal Policy Optimization (PPO) algorithm within DOGE showcases the potential of policy gradient methods in RL for process mining tasks. The ability of PPO to maintain a balance between exploration and exploitation without hyperparameter tuning is particularly important. The choice of the NDL distance as the reward function further enhances the capabilities of DOGE by taking into account potential parallel branches in business processes, stabilizing gradient updates, and promoting a long-term strategic outlook in the agent.

However, it is essential to acknowledge the limitations of DOGE. The training of a policy network for predicting the activity suffix sequence demands substantial computational resources, especially when dealing with event logs that have many activities. Additionally, the network exhibits a high sensitivity to the specific inputs from the log utilized. Therefore, using the process model as an input of the neural network, as seen in methodologies like ASTON and TACO, becomes unfeasible. Nevertheless, the adaptability inherent to the RL framework offers a promising future research line. By incorporating insights about the process model into the reward function in order to *guide* the agent, there is potential to ease the training phase.

CHAPTER 7

CONCLUSIONS

Well...it's a problem that will be solved by wiser heads than mine.

— Tasslehoff Burrfoot, *The Dragonlance Chronicles*

Given the omnipresence of business processes across multiple industries and organizations, it is vital to be able to predict valuable insights about the ongoing execution of processes. This dissertation emphasizes the importance of leveraging the information available in process models to enhance the predictive performance of predictive monitoring models. Neglecting this kind of predictive models would lead to only relying on post-mortem analysis of business processes, which is insufficient to anticipate any potential problems in a business process.

Nowadays, many solutions have been proposed to address this problem based on a plethora of deep learning architectures. While numerous deep learning-based approaches have been developed to predict upcoming activities, a common oversight is their limited utilization of the rich structural information embedded within process models. Some of these methods tend to simplify the representation of event sequences, often resorting to Directly-Follows Graphs (DFGs). While DFGs offer a more straightforward representation, they lack the expressiveness inherent in Petri nets. Concerning the prediction of the activity suffix, there is an over-reliance on predefined sampling strategies. Such strategies, unfortunately, do not consistently deliver across diverse event logs and often result in predictions that are suboptimal. Given the limitations of the state-of-the-art main approaches, we hypothesized

that the performance of predictive monitoring models could be improved by fully leveraging the information available in the process models and by applying better samplings strategies during inference. Furthermore, the field also struggles with the absence of a consistent methodology for comparing various predictive monitoring approaches across different event logs, thereby limiting the development of the field as it is not possible to evaluate the effectiveness of neural architectures or input data encoding, to name two examples. These three drawbacks were addressed in this doctoral thesis:

- We have proposed a methodology for comparing multiple predictive monitoring approaches in a fair and robust way. This methodology spans a comprehensive selection of event logs, allowing the evaluation of any predictive approach in a wide range of scenarios. Central to this methodology is the meticulous preprocessing strategy tailored for process mining data, ensuring accurate estimations and reducing the likelihood of misleading results. Furthermore, our two-stage statistical testing approach facilitates a clearer interpretation of results. Collectively, these elements ensure that our evaluations throughout the dissertation are both reliable and insightful, and pave the way for other researchers that want to compare their predictive monitoring approaches with the state-of-the-art.
- We have introduced TACO, a novel approach to next activity prediction that, in addition to the event log data, also leverages insights from the process model. By proposing the GRNN model that integrates the GRU neural network with GCNs, TACO concurrently processes the Petri net state for each prefix event, capturing both the spatial representation of the Petri net places and the temporal sequence of events. This bidimensional learning approach not only ensures resilience against information loss during process loops but also adeptly captures the dynamic interplay of event replays within the model. The consistency and superior performance of TACO were demonstrated through a comparative analysis against ten leading approaches, utilizing 10 real-world event logs. The results show that TACO statistically outperforms the other state-of-the-art approaches, highlighting the potential of GNNs in predictive process monitoring, especially in more complex process models.
- We have proposed ASTON, a predictive monitoring approach designed to predict the next activity suffix for an ongoing case. Distinctively, ASTON employs an encoder-decoder architecture with an attention mechanism, enabling it to predict the entire suffix

while minimizing errors during inference. By integrating process model information, ASTON enhances the neural network learning phase. Furthermore, our beam search strategy, complemented by a novel normalization technique, allows robust predictions of both long and short trace logs. Through evaluations on 12 real-life event logs 6 main state-of-the-art approaches, ASTON statistically outperforms the other approaches and showcases its adaptability across diverse event logs.

- We have described DOGE, a reinforcement learning-centric solution tailored for the activity suffix prediction. This approach stands apart from conventional predictive monitoring methods, placing a pronounced emphasis on the central role of the sampling strategy in shaping the model effectiveness. DOGE introduces a novel perspective into the activity suffix prediction problem, contrary to those neural approaches that rely on Maximum Likelihood Estimation for their training procedure. Thus, DOGE prioritizes the cumulative reward of the entire suffix sequence, aligning perfectly with the objective of refining neural network configurations and consistently pinpoint the most probable activity. The incorporation of the Proximal Policy Optimization algorithm within DOGE underscores the promise of policy gradient techniques in the realm of reinforcement learning for process mining. Notably, DOGE design choices, such as the normalized Damerau-Levenshtein edit distance for the reward function, amplify its efficacy by accounting for potential parallel pathways in business processes and bolstering a forward-looking strategy in the agent. The results of our evaluations on 12 real-life event logs against 6 state-of-the-art approaches—including ASTON—, show that statistically outperforms the other ones, except ASTON, against which ranks higher. Even though these results show promise, the overall results of the activity suffix prediction problem are still improvable, indicating that there is still room for better results.

With the proposed methodology and the three approaches described in this dissertation, we have successfully achieved all the objectives introduced in Chapter 1. Specifically, we have established a robust methodology for evaluating predictive monitoring techniques, introduced TACO and ASTON that innovatively leverage event log data and process model insights for activity prediction, and presented DOGE, a reinforcement learning solution that emphasizes the role of sampling strategy and the potential of policy gradient methods in process mining. Collectively, these contributions underscore our commitment to enhancing the predictive

performance in process monitoring by holistically integrating process model information. This work holds potential implications across diverse sectors, including healthcare, finance, and manufacturing.

Future work

While the present research has done significant progress in the field of predictive monitoring, there are still several limitations that need to be addressed in future research:

- *Reinforcement Learning and DOGE*: the DOGE approach for activity suffix prediction, rooted in reinforcement learning, faces challenges due to the sparsity of rewards. While this design ensures that the model does not become too focused on individual activities by emphasizing long-term rewards, it encounters difficulties when the model faces event logs with long traces. The learning phase can be hampered, as rewards are only received at the episode conclusion. Additionally, as the number of activities increases, there is a corresponding surge in the computational complexity of the training procedure, primarily attributed to the increased complexity of the action space. A potential solution lies in integrating the process model into the reward function of the agent as opposed to relying on the imperative model discussed in this dissertation. For example, a declarative model expressed through a set of LTL constraints that could either be satisfied or violated depending on the actions of the agent in each timestep. This approach could mitigate reward sparsity by introducing partial rewards based on model constraints. Furthermore, by leveraging the information from the process model, the action space could be further reduced by preventing the agent from undertaking actions outside the allowed behavior of the model.
- *Expanding Process Model Perspectives*: the process models described in this dissertation primarily focus on the control flow perspective, neglecting other important perspectives like the organizational or case perspectives. Notably, the models do not consider interdependencies between cases that occur concurrently. Resources from the event log are treated as another event attribute. An enhancement could involve extending the encoding to support the replay of multiple cases, leading to the automatic extraction of inter-case dependencies. Additionally, applying social network analysis to resources might reveal roles and resource models. These could then be processed similarly to the control-flow perspective discussed in this research.

- *Refining the Graph Recurrent Neural Network Model:* the Graph Recurrent Neural Network model introduced in this dissertation combines the Graph Convolutional Network operation with the input and hidden state of a recurrent model. However, it processes only graph information, leaving the extraction of event log features to another neural network. A potential improvement is the addition of attention mechanisms across spatial and temporal dimensions. This would enhance the model capability, allowing it to process features from both the process model and the event log at the same time. Furthermore, the transformer [108] architecture, renowned for its efficiency and scalability, could be adopted as an alternative to RNNs for the temporal feature extraction, potentially offering improvements in computational speed and interpretability.
- *Improvements over the Methodology:* the methodology proposed in this Ph.D. dissertation serves as a pivotal step toward a standardized, comparative evaluation of predictive monitoring techniques in process mining. Nonetheless, several opportunities for improvement are already present. Firstly, expanding the breadth and depth of the datasets under study would not only add statistical robustness to the evaluations but also ensure broader generalizability across various domains. Secondly, including a wider array of predictive monitoring techniques—ranging from classical machine learning models to newer neural network architectures—could give a more complete view of the predictive monitoring landscape. Furthermore, the utility of this benchmarking procedure could be improved through the development of an intuitive graphical interface. Such an interface would automate the complex processes involved in running the approaches and provide a more user-friendly experience, hence inviting participation from those who may not be experts in predictive monitoring or process mining.

Summarizing, this research has opened several lines of work that promise to further evolve the domain of predictive monitoring. While significant strides have been made in establishing robust methodologies, extracting deep knowledge from process models, and developing innovative models for predictive performance, there remain areas ripe for exploration. The challenges posed by the sparsity of rewards in the DOGE approach, the need to broaden the perspectives of process models, and the potential enhancements to the Graph Recurrent Neural Network model all present opportunities for future research. These avenues not only build upon the foundation laid in this dissertation but also have the potential of making significant contributions that advance the field of process mining.

APPENDIX A

SUPPLEMENTARY MATERIAL FOR THE BENCHMARKING CHAPTER

A.1 Code adaptations

Some of the approaches have undergone some code adaptations:

- **Pasquadibisceglie et al. [73]**. We adapted the code to support more use-cases than in the original paper. Moreover, we optimized the memory usage by changing the datatype of the encoding from 64-bit to 16-bit floating point, and by avoiding loading the entire dataset in RAM to support bigger logs than originally intended.
- **Mauro et al. [68]**. No major changes were made.
- **Tax et al. [99]**. We refactored the code and updated the python version used from 2.7 to 3.6.
- **Evermann et al. [33]**. There are two code versions of the paper provided by the authors: the original one, implemented in Tensorflow 0.12, and an updated version implemented in TensorFlow 1.0. The first one has the disadvantage of requiring the log in a text format and the difficulty of porting it to a newer version of Tensorflow. The second one is tightly coupled with a graphical interface, making it unsuitable to use in the command line. Since the first implementation is based on the Tensorflow RNN tutorial (as stated in the comments of the code), we reimplemented the approach using a newer version

of the RNN tutorial¹, which uses the same training procedure as the implementation of Evermann. We used the same hyperparameters as in the original approach.

- **Khan et al. [52].** The code provided originally by the authors does not contain the preprocessing and encoding stage for any log since it loads the already preprocessed datasets. As far as we know, the code also does not support the prediction of activity suffixes and remaining time, even though in the original paper these results are reported. Therefore, the results reported in this paper are only concerned with the next activity and timestamp. We used the same preprocessing as in Tax et al. [99]. We truncated the traces to a maximum of the last 25 events (in the cross-validated version) due to the excessive time used in the training with the full sequence and by recommendation of the authors. The single train-val-test version uses the full sequence.
- **Theis et al. [102].** Running the code for each log is done in two stages. In the first stage, a process model is mined using the Split Miner [5] by performing a grid search using the same search space, as reported in the original paper [102]. The best model is selected as the mined model with the highest fitness. In the second stage, for generating the training sets and training the neural network, a reimplementation made by the authors of the paper is used², instead of the original implementation of the paper³. The reason for this change is that the original implementation uses a cross-validation procedure that is difficult to circumvent. There is also a ProM plugin⁴ which also generates cross-validation folds, but it was not used since the plugin is already compiled and is very hard to modify. Furthermore, we reported the results for two variants of the implementation: one that uses resources “(w/ resources)” and other that does not use them “(w/o resources)”.
- **Camargo et al. [22].** We reported results for two different methods of sampling the next activity when predicting an activity suffix of activities: always selecting the highest probability activity “(argmax)” and randomly sampling the next activity following the probability distribution of the neural network output “(random)”. We performed a random search hyperparameter optimization procedure following the search space provided in the code and training 20 models per dataset. The architecture used is fixed to the “Shared categorical” architecture reported in the paper [22]. In this architecture,

¹https://www.tensorflow.org/tutorials/text/text_generation

²<https://github.com/Julian-Theis/PyDREAM>

³<https://github.com/Julian-Theis/DREAM-NAP/>

⁴<https://prominentlab.github.io/ProM-DREAM/>

a shared layer integrates the information for the activity and role prefixes, whereas the time prediction is taken separated from the shared layer.

- **Hinkka et al. [45]**. Apart from changing the data split procedure and implementing an adapter to convert logs from CSV to JSON, no major changes were made.
- **Francescomarino et al. [36]**. The execution of this approach involves two main steps: mining the LTL rules from each dataset, which are a series of temporal restrictions that the activities of the log shall follow, and predicting the activity suffix using those rules. We restrict the experimentation to only a “Strong” rule set since it seems to have better results reported in the original paper. To mine the rules we rely on the Rule Mining Tool⁵, and, to avoid leaking information from the testing set, we use the whole validation set with a “minimum constraint support” of 10% and vacuity detection enabled. The templates used for mining are “existence” ($\Diamond A$) and “response” ($\Box(A \rightarrow \Diamond B)$). After mining the rules, we apply the LTL Checker to verify how many rules are satisfied in the validation event log. However, the count reported by the tool counts the number of events that the rule fulfills and not the number of traces in which is fulfilled. Furthermore, a same rule can fulfill a trace multiple times. Thus, instead of taking the rules that are satisfied in more of the 50% of the validation set, we take the rules that have a number of fulfillments more than 50% of the activations ordered by the number of fulfillments up to a maximum of three different rules.
- **Taymouri et al. [100]**: no major changes were made.
- **Venugopal et al. [109]**: no major changes were made. The reported results correspond to the ones of the “weighted” variant in the paper due to them being the most consistent ones across all datasets.
- **Buksh et al. [18]**: no major changes were made.

A.2 Additional next activity experimentation

Some of the approaches for next activity prediction reported additional metrics that could also be used for the evaluation (every approach except [18, 109]). In this section we supply additional measures that are a byproduct of our experimentation. Table A.1 shows the standard

⁵<https://sep.cs.ut.ee/Main/RuM>

deviations of the accuracy measures reported in the paper. Table A.2 shows the brier score measure, Table A.4 shows the Matthews Correlation Coefficient, Table A.3 shows the weighted f1 score, Table A.5 shows the weighted precision, and Table A.6 shows the weighted recall.

- **Brier score** [16]: the Brier score is similar to the accuracy measure, but it also gives a sense of how well the predictions are calibrated with respect to the ground truth. This is specially useful if we are interested in sampling from the probability vector, which is the case of the activity suffix prediction task. The Brier score is calculated as the square of the differences between the predicted probabilities for a given item and the ground truth. More formally, in a multiclass setting, the Brier score is calculated as follows:

$$BS = \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^R (f_{ti} - o_{ti})^2 \in [0, 2] \quad (\text{A.1})$$

Where N is the total number of evaluation samples, R is the number of classes, f is the predicted probabilities for the item i and o is the ground truth outcome for that item (a one hot vector). Note that the formulation of the brier score for a multiclass setting gives a score in the interval $[0, 2]$ whereas the binary formulation gives a score in the interval $[0, 1]$.

- **Matthews correlation coefficient** [67]: the Matthews correlation coefficient (MCC) is a balanced measure that shows the correlation between the predictions of a classifier and the ground truth labels. In a binary classifier, the MCC is calculated from the confusion matrix as follows:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \quad (\text{A.2})$$

This metric can be further extended for a multiclass scenario [40] using the confusion matrix $C \in \mathbb{R}^{K \times K}$:

$$MCC = \frac{\sum_k^K \sum_l^K \sum_m^K C_{kk} C_{lm} - C_{kl} C_{mk}}{\sqrt{\sum_k^K (\sum_l^K C_{kl}) (\sum_{k' \neq k}^K C_{k'l'})} \sqrt{\sum_k^K (\sum_l^K C_{lk}) (\sum_{k' \neq k}^K C_{l'k'})}} \cdot \frac{1}{\sqrt{\sum_k^K (\sum_l^K C_{kl}) (\sum_{k' \neq k}^K C_{k'l'})} \sqrt{\sum_k^K (\sum_l^K C_{lk}) (\sum_{k' \neq k}^K C_{l'k'})}} \quad (\text{A.3})$$

In this latter case, the MCC is not comprised in the interval $[-1, 1]$. Instead, its minimum value varies between 0 and -1 and its maximum value is always 1.

The main advantage of the MCC metric is that takes into account imbalanced classification problems in which one class clearly dominates over the rest.

- **F1 score:** the F1 score is another measure of the accuracy of a classifier and its defined as the harmonic mean of the precision and recall of a classifier. Let P be the precision and R be the recall, then the F1 score is defined as follows:

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \quad (\text{A.4})$$

In this paper we report results in a weighted f1 score fashion: we calculate the F1 score for each class and then average these metrics weighted by the number of samples of each class.

- **Precision:** the precision metric is a measure of the number of correct positive predictions made, i.e, the accuracy for the minority class. This metric is defined as follows:

$$Prec = \frac{TP}{TP + FP} \quad (\text{A.5})$$

We use a weighted precision measure following the same reasoning as in the F1 score.

- **Recall:** the recall metric measures the number of correct positive predictions of every positive prediction made, i.e, it measures the coverage of the positive class. This metric is defined as follows:

$$Rec = \frac{TP}{TP + FN} \quad (\text{A.6})$$

We use a weighted recall metric following the same reasoning as in the F1 score.

A.3 Experimentation with 64/16/20 split

Tables A.7, A.8, and A.9 show the results for some of the approaches evaluated on a different splitting setup. In this experimentation, the traces of the event logs are ordered by its first event timestamp and then split out in train-validation-test sets, with a trace distribution 64%-16%-20%. This split procedure is used since a cross-validation could dramatically increase the training time for some approaches.

	BPI-2012	BPI-2012-A	BPI-2012-C	BPI-2012-O	BPI-2012-W	BPI-2012-W-C	BPI-2013-C-P	BPI-2013-I	Env Permit	Helpdesk
Camargo	±0.69	±1.84	±0.24	±0.92	±7.11	±4.67	±2.71	±0.19	±1.19	±0.82
Evermann	±1.4	±0.53	±0.17	±0.04	±0.24	±0.7	±2.39	±0.15	±1.8	±0.71
Hinkka	±0.11	±0.18	±0.15	±0.17	±0.15	±0.52	±3.34	±0.53	±0.93	±0.78
Khan	±34.57	±0.56	±13.34	±18.44	±11.84	±15.25	±5.53	±6.65	±1.02	±0.89
Mauro	±2.08	±0.19	±0.18	±0.39	±0.28	±0.67	±1.22	±1.43	±1.37	±8.94
Pasqua.	±0.55	±0.4	±0.57	±0.6	±0.61	±0.9	±2.83	±0.37	±0.7	±0.86
Tax	±0.38	±0.85	±0.28	±1.22	±0.33	±0.58	±0.49	±0.37	±0.73	±0.7
Theis (w/o attr.)	±0.93	±0.51	±0.33	±0.24	±2.02	±0.41	±2.75	±0.45	±0.65	±1.05
Theis (w/ attr.)	±0.48	±0.56	±0.28	±0.55	±0.74	±0.61	±3.13	±0.97	±0.59	±0.95

Table A.1: Standard deviation of accuracy of the 5-fold cross-validation

	BPI-2012	BPI-2012-A	BPI-2012-C	BPI-2012-O	BPI-2012-W	BPI-2012-W-C	BPI-2013-C-P	BPI-2013-I	Env Permit	Helpdesk
Camargo	0.219	0.277	0.302	0.224	0.329	0.442	0.581	0.448	0.226	0.269
Evermann	0.613	0.317	0.574	0.281	0.389	0.456	0.533	0.472	0.447	0.286
Hinkka	0.143	0.323	0.213	0.316	0.101	0.200	0.653	0.342	0.380	0.449
Khan	0.921	0.307	0.800	0.516	0.597	0.667	0.709	0.706	0.270	0.313
Mauro	0.209	0.247	0.276	0.222	0.207	0.422	0.866	0.902	0.777	1.018
Pasqua.	0.226	0.321	0.358	0.277	0.282	0.443	0.656	0.700	0.235	0.243
Tax	0.186	0.251	0.271	0.224	0.209	0.400	0.472	0.382	0.233	0.248
Theis (w/o attr.)	0.229	0.424	0.339	0.279	0.222	0.295	0.560	0.506	0.224	0.327
Theis (w/ attr.)	0.258	0.422	0.339	0.297	0.211	0.232	0.621	0.661	0.228	0.309

Table A.2: Mean brier score of the 5-fold cross-validation

Appendix A. SUPPLEMENTARY MATERIAL FOR THE BENCHMARKING CHAPTER

	BPI-2012	BPI-2012-A	BPI-2012-C	BPI-2012-O	BPI-2012-W	BPI-2012-W-C	BPI-2013-C-P	BPI-2013-I	Env Permit	Helpdesk
Camargo	0.811	0.705	0.736	0.809	0.735	0.660	0.523	0.614	0.851	0.797
Evermann	0.504	0.690	0.534	0.775	0.668	0.611	0.505	0.585	0.739	0.800
Hinkka	0.852	0.788	0.772	0.877	0.816	0.676	0.571	0.730	0.825	0.797
Khan	0.425	0.690	0.458	0.637	0.609	0.502	0.369	0.434	0.822	0.753
Mauro	0.830	0.755	0.765	0.808	0.831	0.652	0.100	0.362	0.524	0.284
Pasqua.	0.807	0.687	0.702	0.766	0.775	0.638	0.427	0.374	0.849	0.799
Tax	0.842	0.750	0.767	0.797	0.833	0.682	0.590	0.684	0.842	0.806
Theis (w/o attr.)	0.793	0.572	0.702	0.753	0.815	0.775	0.534	0.547	0.845	0.731
Theis (w/ attr.)	0.781	0.579	0.708	0.739	0.834	0.835	0.510	0.489	0.834	0.756

Table A.3: Mean weighted f1-score of the 5-fold cross-validation

	BPI-2012	BPI-2012-A	BPI-2012-C	BPI-2012-O	BPI-2012-W	BPI-2012-W-C	BPI-2013-C-P	BPI-2013-I	Env Permit	Helpdesk
Camargo	0.825	0.733	0.763	0.785	0.743	0.589	0.383	0.529	0.836	0.784
Evermann	0.572	0.730	0.592	0.766	0.729	0.584	0.498	0.590	0.720	0.796
Hinkka	0.861	0.786	0.793	0.851	0.834	0.626	0.481	0.651	0.821	0.785
Khan	0.415	0.722	0.437	0.576	0.575	0.415	0.266	0.315	0.804	0.728
Mauro	0.840	0.777	0.786	0.801	0.845	0.594	-0.007	0.258	0.518	0.239
Pasqua.	0.825	0.714	0.724	0.758	0.792	0.588	0.362	0.339	0.842	0.790
Tax	0.848	0.775	0.789	0.799	0.838	0.619	0.561	0.627	0.835	0.802
Theis (w/o attr.)	0.821	0.627	0.730	0.751	0.847	0.727	0.430	0.494	0.831	0.688
Theis (w/ attr.)	0.800	0.627	0.735	0.731	0.854	0.782	0.361	0.399	0.816	0.700

Table A.4: Mean MCC of the 5-fold cross-validation

	BPI-2012	BPI-2012-A	BPI-2012-C	BPI-2012-O	BPI-2012-W	BPI-2012-W-C	BPI-2013-C-P	BPI-2013-I	Env Permit	Helpdesk
Camargo	0.828	0.672	0.768	0.871	0.738	0.675	0.608	0.611	0.864	0.793
Evermann	0.461	0.650	0.509	0.834	0.638	0.601	0.462	0.533	0.728	0.785
Hinkka	0.874	0.808	0.804	0.919	0.815	0.694	0.600	0.738	0.826	0.770
Khan	0.536	0.668	0.542	0.669	0.724	0.651	0.403	0.450	0.830	0.738
Mauro	0.845	0.753	0.793	0.839	0.837	0.655	0.063	0.379	0.532	0.260
Pasqua.	0.827	0.696	0.727	0.821	0.758	0.627	0.562	0.487	0.845	0.792
Tax	0.861	0.741	0.783	0.850	0.839	0.715	0.620	0.685	0.847	0.807
Theis (w/o attr.)	0.821	0.544	0.741	0.842	0.787	0.783	0.540	0.553	0.841	0.725
Theis (w/ attr.)	0.811	0.560	0.748	0.820	0.811	0.846	0.518	0.496	0.827	0.763

Table A.5: Mean precision of the 5-fold cross-validation

	BPI-2012	BPI-2012-A	BPI-2012-C	BPI-2012-O	BPI-2012-W	BPI-2012-W-C	BPI-2013-C-P	BPI-2013-I	Env Permit	Helpdesk
Camargo	0.833	0.760	0.779	0.814	0.764	0.690	0.547	0.667	0.858	0.829
Evermann	0.593	0.758	0.624	0.794	0.754	0.675	0.588	0.668	0.762	0.837
Hinkka	0.867	0.812	0.806	0.872	0.848	0.705	0.635	0.747	0.844	0.831
Khan	0.429	0.749	0.474	0.661	0.602	0.522	0.436	0.519	0.836	0.800
Mauro	0.847	0.798	0.801	0.827	0.860	0.686	0.249	0.367	0.536	0.318
Pasqua.	0.833	0.741	0.746	0.789	0.812	0.683	0.475	0.460	0.867	0.839
Tax	0.855	0.795	0.804	0.823	0.854	0.698	0.640	0.701	0.857	0.842
Theis (w/o attr.)	0.829	0.655	0.753	0.784	0.862	0.801	0.595	0.594	0.863	0.788
Theis (w/ attr.)	0.810	0.657	0.758	0.769	0.869	0.838	0.547	0.515	0.851	0.797

Table A.6: Mean recall of the 5-fold cross-validation

	Helpdesk	BPI-2012	BPI-2012-C	BPI-2012-W	BPI-2012-W-C	BPI-2012-O	BPI-2012-A	BPI-2013-C-P	BPI-2013-I	Env permit
Pasqua.	65.84	82.59	74.55	81.59	66.14	77.51	71.47	24.35	31.10	91.47
Tax el al.	75.06	85.20	79.39	84.90	67.80	81.22	77.75	65.57	67.50	89.24
Camargo	76.51	83.41	79.22	83.29	65.19	85.13	78.92	60.62	68.01	91.38
Hinkka	77.90	86.05	79.76	83.52	67.24	85.51	79.27	61.14	77.95	89.46
Khan	69.13	82.93	75.50	86.69	75.91	84.48	75.62	55.57	64.34	89.78
Evermann	70.07	60.38	63.37	75.22	65.38	79.20	74.44	55.66	68.15	84.33
Mauro	74.77	84.56	78.72	85.11	65.01	81.52	78.09	56.97	71.09	87.29
Theis (w/o attr)	67.80	77.64	73.10	85.77	76.97	81.52	66.23	52.31	57.65	91.32
Theis (w/ attr)	66.25	64.23	65.21	76.16	72.52	73.56	65.12	47.69	63.51	82.35

Table A.7: Accuracy measures in % for the prediction in a 64/16/20 split.

	Helpdesk	BPI-2012	BPI-2012-C	BPI-2012-W	BPI-2012-W-C	BPI-2012-O	BPI-2012-A	BPI-2013-C-P	BPI-2013-I	Env permit
Pasqua	0.658	0.228	0.356	0.269	0.460	0.314	0.354	0.904	0.903	0.179
Tax el al.	0.387	0.196	0.286	0.220	0.435	0.257	0.272	0.476	0.418	0.183
Camargo	0.372	0.209	0.287	0.244	0.471	0.182	0.251	0.567	0.418	0.162
Hinkka	0.273	0.113	0.163	0.127	0.228	0.255	0.316	0.642	0.272	0.223
Khan	0.461	0.217	0.335	0.178	0.341	0.214	0.295	0.632	0.420	0.177
Evermann	0.416	0.627	0.576	0.401	0.497	0.302	0.327	0.483	0.484	0.417
Mauro	0.371	0.201	0.295	0.217	0.468	0.255	0.269	0.489	0.358	0.228
Theis (w/o attr)	0.520	0.319	0.368	0.210	0.326	0.263	0.412	0.629	0.534	0.172
Theis (w/ attr)	0.492	0.540	0.484	0.379	0.430	0.385	0.431	0.673	0.505	0.258

Table A.8: Brier score measures for the next activity prediction in a 64/16/20 split.

Appendix A. SUPPLEMENTARY MATERIAL FOR THE BENCHMARKING CHAPTER

	Helpdesk	BPI-2012	BPI-2012-C	BPI-2012-W	BPI-2012-W-C	BPI-2012-O	BPI-2012-A	BPI-2013-C-P	BPI-2013-I	Sepsis	Env permit	Nasa
Evermann	0.807	0.198	0.268	0.277	0.332	0.543	0.582	0.671	0.479	0.279	0.572	0.127
Tax	0.812	0.121	0.175	0.096	0.062	0.565	0.454	0.486	0.233	0.098	0.878	0.060
Camargo (argmax)	0.880	0.163	0.183	0.149	0.027	0.687	0.634	0.726	0.300	-	0.889	-
Camargo (random)	0.837	0.380	0.453	0.339	0.333	0.595	0.642	0.513	0.551	-	0.756	-
Francesco.	0.199	0.119	0.090	0.101	0.258	0.348	0.238	0.434	0.274	0.077	0.176	0.053

Table A.9: Activity suffix prediction Damerau Levenshtein distance in a 64/16/20 split.

APPENDIX B

ADDITIONAL EXPERIMENTS FOR GRAPH-BASED NEXT ACTIVITY PREDICTION

B.1 Process model statistics

Table B.1 shows the average statistics of the mined models for the cross-validation train and validation sets. The reported metrics for the process models, from left to right in the Table, are the average number of loops, edges, transitions, places, and length and maximum length value of the loops that contain exclusively the same activity. This Table shows that the mined process models vary in complexity. The most simple process models are the ones mined from the event logs *BPI-2012-A* and *BPI-2012-O* since they have the lowest number of loops, transitions, and places; while the most complex event logs are the *BPI-2012*, *BPI-2012-C* and *Env-permit*. Note that the *Complete* (C) variants of the *BPI-2012* logs have more loops than their non-complete counterparts since we treat each combination of *lifecycle:transition* and activity as a separate activity.

B.2 Loop behavior

Table B.2 shows some detailed statistics about the presence of loops in the event logs, as well as the performance of TACO when compared to the LSTM in the ablation study. Furthermore, Tables B.3 and B.4 show a fine-grained list of the presence of loops in the event logs. For each event log, two columns are presented: the first one, “Len.”, refers to the number of different activities that compose a loop, whereas the second one shows the number of loop instances

Event log	Loops	Edges	Transitions	Places	Max. rep. loop	Avg. rep. loop
BPI-2012	18404807.6	381.2	190.6	68	3	2
BPI-2012-A	5	73.6	36.8	19	0	0
BPI-2012-C	18404807.6	384.4	192.2	69.6	56	4.73
BPI-2012-O	9	48	24	13	0	0
BPI-2012-W	15.6	95.6	47.8	28.0	3	2
BPI-2012-W-C	15.6	96	48	28.2	57	4.66
BPI-2013-C-P	50	96.4	48.2	28.2	5	2.16
Helpdesk	571.6	92.8	186.4	45.6	5	2.1
BPI-2013-I	128.4	65.2	32.6	16	19	2.07
Env-permit	83187.2	255.2	127.6	52.2	3	3

Table B.1: Average statistics of the process models.

with that length in the log. For example, two loops such as “AAAAA” and “AAAA” would be accounted as a tuple “(A, 2)” with $Len. = 1$ and $Num. = 2$. In these logs, most loops have few activities but with high incidence, like *BPI-2012-W*, whose average case and loop length are 17.61 and 2, respectively.

As shown by Table B.2, the best results are achieved in the highlighted logs, which have a high number of loops and a short average case length, with the exception of *Env-permit*. This shows that our approach is able to extract better the loop information when the traces have a high number of loops. Furthermore, when no loops are available in the event log, such as in *BPI-2012-A*, the performance of the LSTM and TACO is virtually the same.

In conclusion, the performance of TACO depends on two main factors: the complexity of the underlying process model and the information provided by the timestamps and event attributes of the event log. On the one hand, as shown in Table B.2, TACO provides an edge when the underlying process model is fairly complex. In these cases, an LSTM is unable to capture every complex relationship between the events of the log. On the other hand, the usage of the information from the timestamps and event attributes proves to be useful in most cases. However, in some logs, such as *BPI-2012-A*, the underlying process model is too simple—our predictive model captures the same information as any model not based on structural information—and the usage of the information from the timestamps and event attributes is

	Total cases	Avg. cases length	Cases with loops	% cases with loops	Number of loop appearances	LSTM	TACO	Difference
BPI-2012	13087	20.04	5643	43.12	9342	86.71	87.08	0.37
BPI-2012-C	13087	12.57	5712	43.64	9932	80.32	80.85	0.53
BPI-2012-A	13087	4.65	0	0	0	79.77	79.78	0.01
BPI-2012-O	5015	6.23	350	6.98	356	82.65	82.95	0.3
BPI-2012-W	9658	17.61	6583	68.16	10810	88.06	88.34	0.28
BPI-2012-W-C	9568	7.5	7011	72.59	13075	73.76	74.2	0.44
BPI-2013-C-P	1487	4.48	449	30.2	566	65.29	<u>67.53</u>	2.24
BPI-2013-I	7554	8.68	7114	94.18	10970	75.84	<u>77.72</u>	1.88
Env-permit	1434	5.98	30	2.09	32	85.9	<u>87.66</u>	1.76
Helpdesk	4580	4.66	950	20.74	1021	79.99	<u>85.2</u>	5.21

Table B.2: Loop statistics of the tested process logs.

not useful; so, in that cases, the performance will not be improved by TACO with respect to an LSTM that only uses the information from the activities of the event log.

B.3 Process discovery algorithm effect

In this section, we conducted a series of experiments to evaluate the sensitivity of TACO to the process discovery algorithm used for obtaining the underlying process model that describes the observed behavior in the event log. To this end, we compared the performance of our approach using both Split Miner and Inductive Miner [57]. The Inductive Miner is a widely adopted algorithm that aims to discover sound and fitting models in a finite time. Based on the divide-and-conquer principle, this algorithm recursively partitions the event log to create a process tree, which is subsequently converted into a Petri net. On the other hand, the Split Miner aims to discover a BPMN model, which is, in turn, converted into a Petri net. To do so, it begins by obtaining a directly follows graph (DFG) of the event log, which is subsequently pruned and refined by discovering XOR and AND gateways. We opted to use the Inductive Miner for the comparison due to its popularity and effectiveness in process discovery tasks and because it employs a different approach than the Split Miner, so the discovered process models by this algorithm have different structural properties.

BPI-2012		BPI-2012-C		BPI-2012-W		BPI-2012-W-C	
Len	Num	Len	Num	Len	Num	Len	Num
1	153	1	9719	1	160	1	12962
2	9018	2	8	2	10378	2	11
3	1	3	26	3	121	3	31
4	1	4	18	4	1	4	22
6	4	5	90	5	45	5	24
7	74	6	32	6	4	6	9
8	14	7	18	7	30	7	5
9	25	8	8	8	17	8	4
10	7	9	4	9	7	9	1
11	14	10	3	10	11	10	1
12	11	11	1	11	4	11	1
13	5	13	2	12	13	13	1
14	6	14	2	13	2	14	1
15	3	19	1	14	6	16	1
16	3			16	5	19	1
17	2			18	3		
18	2			19	1		
23	1			21	1		
25	1			23	1		

Table B.3: Fine grained loop statistics.

BPI-2012-O		BPI-2013-C-P		BPI-2013-I		Helpdesk		Env-permit	
Len	Num	Len	Num	Len	Num	Len	Num	Len	Num
4	320	1	388	1	7897	1	855	1	3
5	30	2	163	2	2737	2	149	2	28
8	6	3	13	3	173	3	12	3	1
		4	2	4	111	4	5		
				5	23				
				6	15				
				7	11				
				9	2				
				11	1				

Table B.4: Fine grained loop statistics (cont.).

In our experiments, both algorithms were configured to discover models with perfect fitness, ensuring a fair comparison. The results of this experimentation are highlighted in Table B.5, where it can be seen that there are very small differences in the results obtained by TACO with both algorithms. The biggest differences are in the event logs *BPI-2013-C-P* and *BPI-2012-W-C*, with a variation in performance of 0.9% and 0.34%, respectively. These results confirm that TACO is insensitive to the algorithm used to discover the process model.

	TACO S.M.	TACO I.M.	Difference
BPI-2012	87.00	86.83	0.17
BPI-2012-A	79.92	79.79	0.13
BPI-2012-C	80.89	80.90	-0.01
BPI-2012-O	83.07	82.96	0.11
BPI-2012-W	88.29	88.32	-0.03
BPI-2012-W-C	74.53	74.19	0.34
BPI-2013-C-P	67.86	66.96	0.9
BPI-2013-I	77.58	77.37	0.21
Env-Permit	87.65	87.56	0.09
Helpdesk	85.10	85.15	-0.05

Table B.5: Performance of TACO (accuracy) using the Split Miner (S.M.) and the Inductive Miner (I.M.).

Figures B.1 and B.2 show how the validation accuracy evolves using both model miners in the second cross-validation fold for *BPI-2013-C-P* and *BPI-2012-W-C*. For *BPI-2013-C-P*, TACO obtains an accuracy of 65.99% and 68.31% using Inductive Miner and Split Miner, respectively; while for *BPI-2012-W-C*, it gets 74.09% and 74.92%, respectively. These results correspond with the highest point in the validation accuracy plots of Figures B.1 and B.2 —epoch 6 for both event logs—. On the one hand, TACO with Split Miner takes slightly longer to trigger the early stopping criterion, an effect that is evidenced by the fact that it takes more epochs to stop training. On the other hand, TACO with Split Miner achieves higher validation accuracies during training, since in both figures the maximum accuracy is always reached with the Split Miner algorithm.

B.4 Hyperparameter effect

In this section, a search of hyperparameters was conducted to evaluate their influence on the performance of our approach. Considering that TACO has 12 different hyperparameters, each

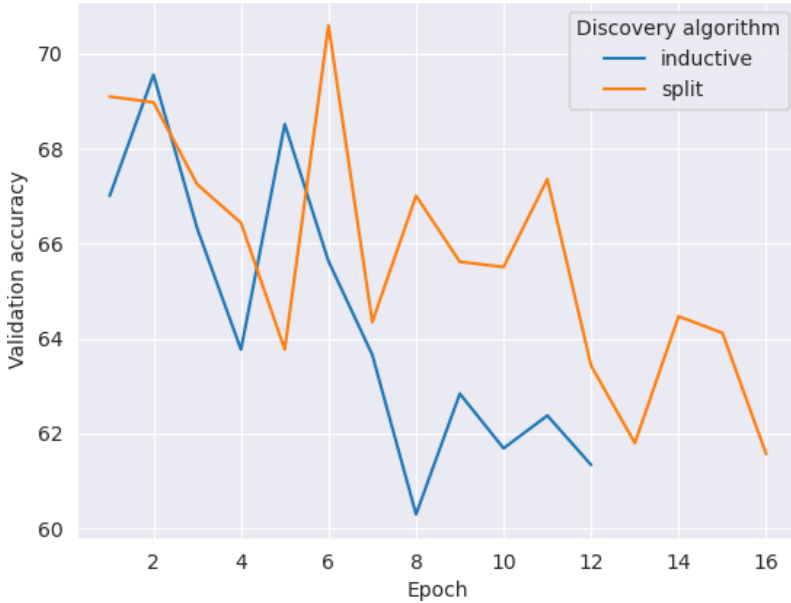


Figure B.1: Validation accuracy in *BPI-2013-C-P*

with three possible values and that 10 event logs are used in a 5-fold cross-validation to evaluate the influence of the hyperparameters in the performance of TACO, we would have had to train $3^{12} \cdot 5 \cdot 10 = 26,572,050$ different models. Thus, our experimentation has been restricted to only four hyperparameters that directly affect the graph-based part of our approach: the hidden size of the first and second GRNN layers —“Hidden 1” and “Hidden 2”, respectively— and the dropout of the first and second GRNN layers —“Dropout 1” and “Dropout 2”, respectively—.

Furthermore, in order to reduce the number of models tested, we only tried three different values for each hyperparameter, giving 81 possible combinations. In particular, the values of 64, 128, 256, and 0, 0.1, 0.2 were used for each possible hidden size and value of dropout [90], respectively. Finally, we performed a grid search on the first fold of two representative event logs: *Helpdesk* and *BPI-2012-W-C*.

Figure B.3, B.4, B.5, and B.6 exhibit the parallel coordinate plots for each hyperparameter evaluated in the two event logs. To simplify the interpretation of the figures, we separated the coordinate plots for each dataset into two distinct figures, encompassing the 10 best-performing sets of hyperparameters and the 10 worst-performing sets of hyperparameters,



Figure B.2: Validation accuracy in *BPI-2012-W-C*

respectively, for each evaluated event log. Thus, Figure B.3 and B.4 depict the parallel coordinate plots for each hyperparameter combination on the Helpdesk event log. The first plot, Figure B.3, showcases the 10 best-performing evaluations while Figure B.4, portrays the 10 worst-performing ones. As evidenced by these figures, most of the combinations successfully reach the optimal maximum with roughly 84% accuracy, barring a few notable exceptions with validation accuracies ranging from 18% to 71%. In this case, the failure to attain a satisfactory validation accuracy is directly linked to using a dropout value exceeding 0 in the second layer of the GRNN, whereas the first layer can maintain a dropout value above 0 without necessarily failing to converge. Figure B.5 and B.6 present the same coordinate plots, but for the BPI-2012-W-C event log. In this situation, most combinations reach a local optimum of approximately 72%, aside from those that fall short due to a dropout value higher than 0 in the second layer of the GRNN.

Finally, Figure B.7 and B.8 show the parallel plots for the results depicted in the aforementioned tables.

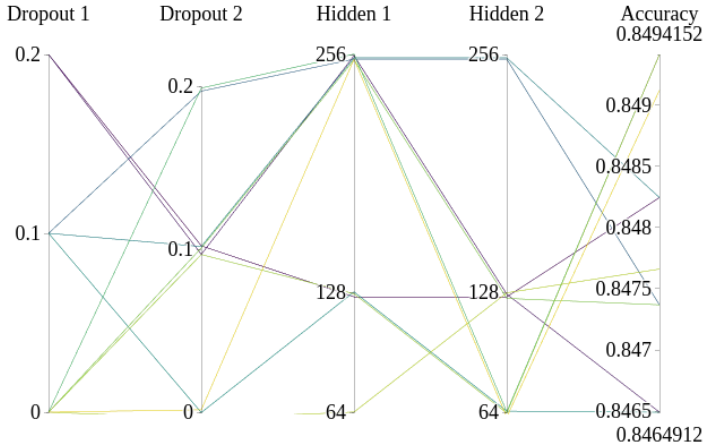


Figure B.3: Parallel coordinate plot of the 10 best-performing hyperparameters in the Helpdesk event log.

B.5 Computation time and memory

In this section, we compare TACO using embeddings or one-hot encoding in terms of computation memory and memory. The results of the memory and number of parameters for the whole training phase are depicted in Table B.6, where TACO, with embeddings, outperforms the one-hot encoding version in every dataset except in the *BPI 2012 A* and *BPI 2012 O*, where the results are almost the same as TACO with one-hot. The most significant improvement is in the *BPI 2013 Incidents* log, with a reduction of 70.43% and 56.31%, respectively. However, the results of computation time in seconds for one epoch, show that the one-hot encoding approach performs slightly better than the embedding one in this regard because the latter approach also has to update the parameters for each of the embeddings.

On the other hand, Table B.7 shows the results for the memory and number of parameters used if the transitions are added to the place graph (*full adjacency*). In this case, both variations of the approach use embeddings to encode the categorical features. As shown in the aforementioned tables, the usage of the place graph allows for a dramatic decrease in memory usage and a mild reduction of the number of parameters on every event log of the

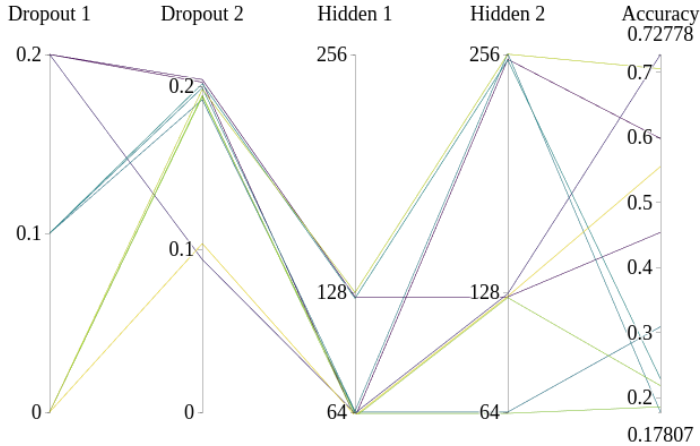


Figure B.4: Parallel coordinate plot of the 10 worst-performing hyperparameters in the Helpdesk event log.

	VRAM (MB)			Parameters (millions)			Computation time	
	TACO	one-hot	%	TACO	one-hot	%	TACO	one-hot
BPI-2012	14024	31594	55.61%	2.43	3.77	35.64%	7996	7923
BPI-2012-A	2121	2123	0.09%	2.02	3.77	1.25%	202	197
BPI-2012-C	8940	23884	62.57%	2.40	3.80	36.77%	2837	2830
BPI-2012-O	2331	2338	0.29%	1.98	1.95	-1.57%	132	129
BPI-2012-W	5742	8516	32.57%	2.12	2.19	3.16%	4015	3963
BPI-2012-W-C	3749	5034	25.53%	2.08	2.17	3.80%	772	770
BPI-2013-C-P	2842	3218	11.68%	2.14	2.87	25.34%	27	26
Helpdesk	2526	3579	29.42%	2.16	2.53	14.64%	55	54
BPI-2013-I	8750	29594	70.43%	2.61	5.97	56.31%	1119	1102
Env-permit	3430	7633	55.06%	2.38	3.19	25.44%	28	27

Table B.6: Comparison of VRAM, number of parameters, and computation time of TACO using embeddings or one-hot encodings. % denotes the percentage of reduction between each pair of strategies compared.

experimentation, allowing the *BPI 2012* event log to be computationally feasible. Moreover, in terms of the computation time, the reduction is not as noticeable as in the case of the memory consumption, but still, for every event log tested, there is a degree of reduction in the time of computation.

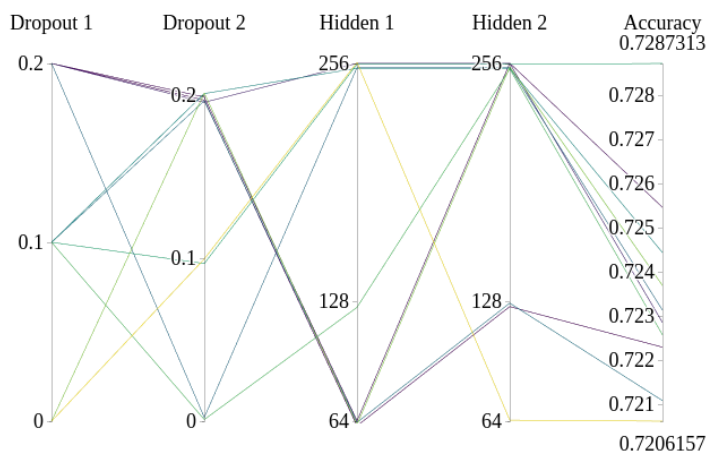


Figure B.5: Parallel coordinate plot of the 10 best-performing hyperparameters in the BPI-2012-W-C event log.

	VRAM (MB)			Parameters (millions)			Computation time	
	TACO	full adj.	%	TACO	full adj.	%	TACO	full adj.
BPI 2012	14024	OOM	OOM	2.43	OOM	OOM	7996	OOM
BPI 2012 A	2121	2377	10.77%	2.02	2.19	7.76%	202	212
BPI 2012 C.	8940	28638	68.78%	2.40	3.29	27.05%	2837	3534
BPI 2012 O	2331	2937	20.63%	1.98	2.10	5.7%	132	137
BPI 2012 W	5742	12272	53.21%	2.12	2.34	9.4%	4015	4052
BPI 2012 W C.	3749	7346	48.97%	2.08	2.30	9.57%	772	787
BPI 2013 C. P.	2842	4383	35.15%	2.14	2.36	9.32%	27	27
Helpdesk	2526	3894	35.13%	2.16	2.63	17.87%	55	59
BPI 2013 I.	8750	24064	63.64%	2.61	3.23	19.20%	1119	1181
Env. Permit	3430	6600	48.03%	2.38	2.97	19.87%	28	31

Table B.7: Comparison of VRAM, number of parameters, and computation time of TACO using the place graph or the full adjacency matrix. OOM represents an out-of-memory error. % denotes the percentage of reduction between each pair of strategies compared.

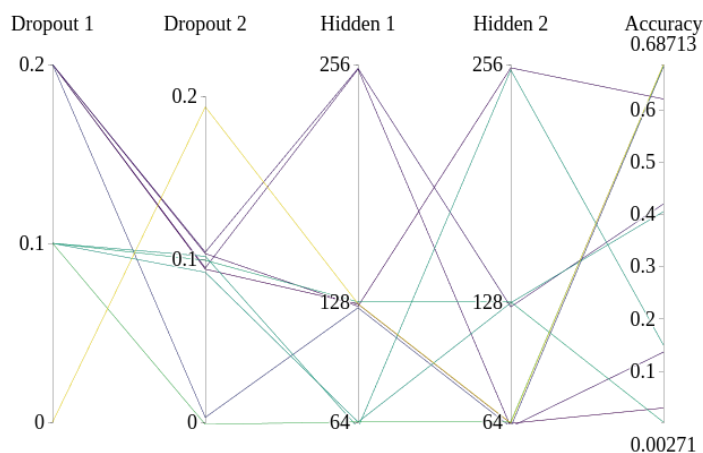


Figure B.6: Parallel coordinate plot of the 10 worst-performing hyperparameters in the BPI-2012-W-C event log.

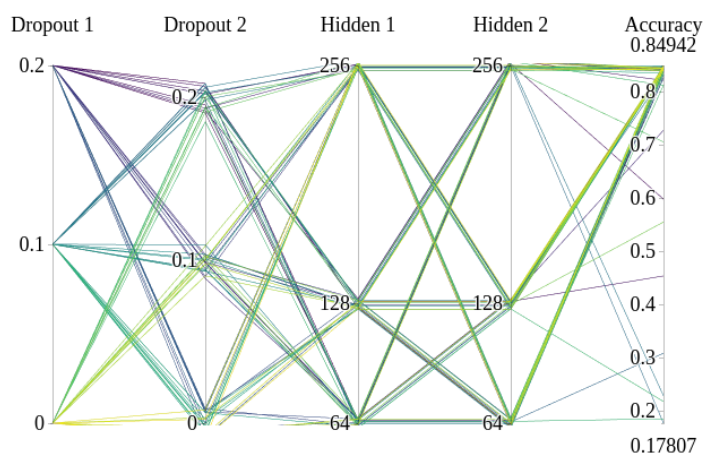


Figure B.7: Parallel coordinate plot of the performance of different hyperparameters in the Helpdesk event log.

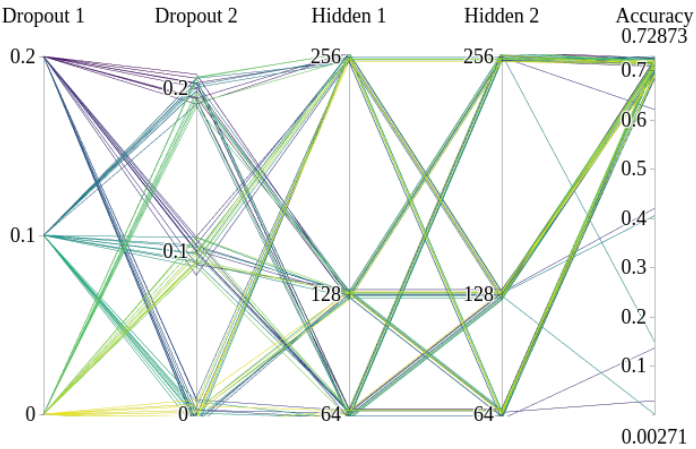


Figure B.8: Parallel coordinate plot of the performance of different hyperparameters in the BPI-2012-W-C event log.

APPENDIX C

LIST OF PUBLICATIONS, AUTHOR'S CONTRIBUTION AND COPYRIGHT INFORMATION

This Ph.D dissertation reproduces the contents of the following publications:

- Chapter 3 reproduces the contents of Efrén Rama-Maneiro, Juan Vidal, Manuel Lama, *Deep Learning for Predictive Business Process Monitoring: Review and Benchmark*, IEEE Transactions on Services Computing, 16:739-756, 2023 (Available online: December 2021). ISSN: 1939-1374. <https://doi.org/10.1109/TSC.2021.3139807>. Journal Impact Factor: 8.1 (JCR 2022), 2.11 (SJR 2022). Journal ranked in JCR 2022, in Computer science, information systems, (Q1 13/158); in Computer science, Software Engineering (Q1, 4/108).

- **Author's contributions:**

Efrén Rama-Maneiro: Methodology, Software, Validation, Formal Analysis, Investigation, Data Curation, Writing — Original Draft.

Juan Vidal: Conceptualization, Resources, Writing — Review & Editing, Visualization, Supervision, Project administration, Funding acquisition, Investigation.

Manuel Lama: Conceptualization, Resources, Writing — Review & Editing, Visualization, Supervision, Project administration, Funding acquisition, Investigation.

Reproduction authorization: IEEE allows to reuse accepted papers in a Ph.D dissertation without express permission (see <https://journals.ieeeauthorcenter.ieee.org/choose-a-publishing-agreement/avoid-infringement-upon-ieee-copyright/>).

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

- Chapter 4 reproduces the contents of Efrén Rama-Maneiro, Juan Vidal, Manuel Lama, *Embedding Graph Convolutional Networks in Recurrent Neural Networks for Predictive Monitoring*, IEEE Knowledge and Data Engineering, 1-16, 2023 (Available online: June 2023). ISSN: 1041-4347. <https://doi.org/10.1109/TKDE.2023.3286017>. Journal Impact Factor: 8.9 (JCR 2022), 2.36 (SJR 2022). Journal ranked in JCR 2022, in Computer Science, Artificial Intelligence (Q1, 18/145); in Computer Science, Information Systems (Q1, 9/158); in Engineering, Electrical & Electronic (Q1, 20/275).
- **Author's contributions:**

Efrén Rama-Maneiro: Methodology, Software, Validation, Formal Analysis, Investigation, Data Curation, Writing — Original Draft.

Juan Vidal: Conceptualization, Resources, Writing — Review & Editing, Visualization, Supervision, Project administration, Funding acquisition, Investigation.

Manuel Lama: Conceptualization, Resources, Writing — Review & Editing, Visualization, Supervision, Project administration, Funding acquisition, Investigation.

Reproduction authorization: IEEE allows to reuse accepted papers in a Ph.D dissertation without express permission (see <https://journals.ieeeauthorcenter.ieee.org/choose-a-publishing-agreement/avoid-infringement-upon-ieee-copyright/>).

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

- Chapter 5 reproduces the contents of Efrén Rama-Maneiro, Pablo Monteagudo-Lago, Juan Vidal, Manuel Lama, *Exploiting Recurrent Graph Neural Networks for Suffix Prediction in Predictive Monitoring*, <https://arxiv.org/abs/2211.16106>
- **Author's contributions:**

Efrén Rama-Maneiro: Methodology, Software, Validation, Formal Analysis, Investigation, Data Curation, Writing — Original Draft.

Pablo Monteagudo-Lago: Software, Writing — Review & Editing.

Juan Vidal: Conceptualization, Resources, Writing — Review & Editing, Visualization, Supervision, Project administration, Funding acquisition, Investigation.

Manuel Lama: Conceptualization, Resources, Writing — Review & Editing, Visualization, Supervision, Project administration, Funding acquisition, Investigation.

Bibliography

- [1] Arya Adriansyah and Joos C. A. M. Buijs. Mining process performance from event logs. In Marcello La Rosa and Pnina Soffer, editors, *Business Process Management Workshops - BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers*, volume 132 of *Lecture Notes in Business Information Processing*, pages 217–218. Springer, 2012.
- [2] 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, 2018.
- [3] Anonymous. The 37 implementation details of proximal policy optimization. In *Blog Track at ICLR 2022*, 2022.
- [4] Abel Armas-Cervantes, N. R. T. P. van Beest, Marcello La Rosa, Marlon Dumas, and Luciano García-Bañuelos. Interactive and incremental business process model repair. In Hervé Panetto, Christophe Debruyne, Walid Gaaloul, Mike P. Papazoglou, Adrian Paschke, Claudio Agostino Ardagna, and Robert Meersman, editors, *On the Move to Meaningful Internet Systems. OTM 2017 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2017, Rhodes, Greece, October 23-27, 2017, Proceedings, Part I*, volume 10573 of *Lecture Notes in Computer Science*, pages 53–74. Springer, 2017.
- [5] Adriano Augusto, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, and Artem Polyvyanny. Split miner: automated discovery of accurate and simple business process models from event logs. *Knowl. Inf. Syst.*, 59(2):251–284, 2019.

- [6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations (ICLR 2015)*, 2015.
- [7] Monya Baker. 1, 500 scientists lift the lid on reproducibility. *Nature*, 533(7604):452–454, May 2016.
- [8] Jörg Becker, Dominic Breuker, Patrick Delfmann, and Martin Matzner. Designing and implementing a framework for event-based predictive modelling of business processes. In Fernand Feltz, Bela Mutschler, and Benoît Otjacques, editors, *Enterprise modelling and information systems architectures - EMISA 2014, Luxembourg, September 25-26, 2014*, volume P-234 of *LNI*, pages 71–84. GI, 2014.
- [9] Alessio Benavoli, Giorgio Corani, Janez Demsar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *J. Mach. Learn. Res.*, 18:77:1–77:36, 2017.
- [10] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *28th Annual Conference on Neural Information Processing Systems (NIPS 2015)*, pages 1171–1179. ACM, 2015.
- [11] Alessandro Berti, Sebastiaan J. van Zelst, and Wil M. P. van der Aalst. Process mining for python (pm4py): Bridging the gap between process- and data science. *CoRR*, abs/1905.06169, 2019.
- [12] Alessandro Bettacchi, Alberto Polzonetti, and Barbara Re. Understanding production chain business process using process mining: A case study in the manufacturing scenario. In John Krogstie, Haralambos Mouratidis, and Jianwen Su, editors, *Advanced Information Systems Engineering Workshops - CAiSE 2016 International Workshops, Ljubljana, Slovenia, June 13-17, 2016, Proceedings*, volume 249 of *Lecture Notes in Business Information Processing*, pages 193–203. Springer, 2016.
- [13] Leonid Boytsov. Indexing methods for approximate dictionary searching. *ACM Journal of Experimental Algorithmics*, 16, May 2011.
- [14] Dominic Breuker, Martin Matzner, Patrick Delfmann, and Jörg Becker. Comprehensive predictive models for business processes. *MIS Q.*, 40(4):1009–1034, 2016.

- [15] GLENN W. BRIER. VERIFICATION OF FORECASTS EXPRESSED IN TERMS OF PROBABILITY. *Monthly Weather Review*, 78(1):1–3, January 1950.
- [16] Glenn W. Brier. Verification of forecasts in terms of probaiblity. *Monthly Weather Review*, 78(1):1–3, 1950.
- [17] J.C.A.M Buijs. Environmental permit application process (‘wabo’), coselog project – municipality 5, 2014.
- [18] Zaharah Allah Bukhsh, Aaqib Saeed, and Remco M. Dijkman. Processtransformer: Predictive business process monitoring with transformer network. *CoRR*, abs/2104.00721, 2021.
- [19] Cristina Cabanillas, Claudio Di Ciccio, Jan Mendling, and Anne Baumgrass. Predictive task monitoring for business processes. In Shazia Wasim Sadiq, Pnina Soffer, and Hagen Völzer, editors, *Business Process Management - 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings*, volume 8659 of *Lecture Notes in Computer Science*, pages 424–432. Springer, 2014.
- [20] Borja Calvo, Josu Ceberio, and Jose A. Lozano. Bayesian inference for algorithm ranking analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, July 2018.
- [21] Borja Calvo and Guzmán Santafé. scmamp: Statistical comparison of multiple algorithms in multiple problems. *R J.*, 8(1):248, 2016.
- [22] Manuel Camargo, Marlon Dumas, and Oscar González Rojas. Learning accurate LSTM models of business processes. In Thomas T. Hildebrandt, Boudewijn F. van Dongen, Maximilian Röglinger, and Jan Mendling, editors, *Business Process Management - 17th International Conference, BPM 2019, Vienna, Austria, September 1-6, 2019, Proceedings*, volume 11675 of *Lecture Notes in Computer Science*, pages 286–302. Springer, 2019.
- [23] Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on*

- Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL, 2014.
- [24] Jacek Dabrowski, Fitsum Meshesha Kifetew, Denisse Muñante, Emmanuel Letier, Alberto Siena, and Angelo Susi. Discovering requirements through goal-driven process mining. In *IEEE 25th International Requirements Engineering Conference Workshops, RE 2017 Workshops, Lisbon, Portugal, September 4-8, 2017*, pages 199–203. IEEE Computer Society, 2017.
 - [25] Benjamin Dalmas, Fabrice Baranski, and Daniel Cortinovis. Predicting process activities and timestamps with entity-embeddings neural networks. In Samira Si-Said Cherfi, Anna Perini, and Selmin Nurcan, editors, *Research Challenges in Information Science - 15th International Conference, RCIS 2021, Limassol, Cyprus, May 11-14, 2021, Proceedings*, volume 415 of *Lecture Notes in Business Information Processing*, pages 393–408. Springer, 2021.
 - [26] Michal Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. Frustratingly short attention spans in neural language modeling. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
 - [27] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 933–941. PMLR, 2017.
 - [28] Fábio de Lima Bezerra, Jacques Wainer, and Wil M. P. van der Aalst. Anomaly detection using process mining. In Terry A. Halpin, John Krogstie, Selmin Nurcan, Erik Proper, Rainer Schmidt, Pnina Soffer, and Roland Ukor, editors, *Enterprise, Business-Process and Information Systems Modeling, 10th International Workshop, BPMDS 2009, and 14th International Conference, EMMSAD 2009, held at CAiSE 2009, Amsterdam, The Netherlands, June 8-9, 2009. Proceedings*, volume 29 of *Lecture Notes in Business Information Processing*, pages 149–161. Springer, 2009.

- [29] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.
- [30] Davide Duma and Roberto Aringhieri. An ad hoc process mining approach to discover patient paths of an emergency department. *Flexible Services and Manufacturing Journal*, 32(1):6–34, December 2018.
- [31] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management, Second Edition*. Springer, 2018.
- [32] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep rl: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2020.
- [33] Joerg Evermann, Jana-Rebecca Rehse, and Peter Fettke. Predicting process behaviour using deep learning. *CoRR*, abs/1612.04600, 2016.
- [34] Angela Fan, Mike Lewis, and Yann N. Dauphin. Hierarchical neural story generation. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 889–898. Association for Computational Linguistics, 2018.
- [35] Chiara Di Francescomarino, Marlon Dumas, Fabrizio Maria Maggi, and Irene Teinmaa. Clustering-based predictive process monitoring. *CoRR*, abs/1506.01428, 2015.
- [36] 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. In Josep Carmona, Gregor Engels, and Akhil Kumar, editors, *Business Process Management - 15th International Conference, BPM 2017, Barcelona, Spain, September 10-15, 2017, Proceedings*, volume 10445 of *Lecture Notes in Computer Science*, pages 252–268. Springer, 2017.

- [37] Pedro Gamallo-Fernandez, Juan C. Vidal, and Manuel Lama. Learning context-based representations of events in complex processes. In *2023 IEEE International Conference on Web Services (ICWS)*, pages 214–223, 2023.
- [38] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. <http://www.deeplearningbook.org>.
- [39] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014.
- [40] J. Gorodkin. Comparing two k-category assignments by a k-category correlation coefficient. *Computational Biology and Chemistry*, 28(5-6):367–374, 2004.
- [41] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwinska, Sergio Gomez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John P. Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nat.*, 538(7626):471–476, 2016.
- [42] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970.
- [43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.
- [44] Kai Heinrich, Patrick Zschech, Christian Janiesch, and Markus Bonin. Process data properties matter: Introducing gated convolutional neural networks (GCNN) and key-value-predict attention networks (KVP) for next event prediction with deep learning. *Decis. Support Syst.*, 143:113494, 2021.

- [45] Markku Hinkka, Teemu Lehto, and Keijo Heljanko. Exploiting event log event attributes in RNN based prediction. In Paolo Ceravolo, Maurice van Keulen, and María Teresa Gómez López, editors, *Data-Driven Process Discovery and Analysis - 8th IFIP WG 2.6 International Symposium, SIMPDA 2018, Seville, Spain, December 13-14, 2018, and 9th International Symposium, SIMPDA 2019, Bled, Slovenia, September 8, 2019, Revised Selected Papers*, volume 379 of *Lecture Notes in Business Information Processing*, pages 67–85. Springer, 2019.
- [46] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.
- [47] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.
- [48] Gema Ibanez-Sanchez, Carlos Fernandez-Llata, Antonio Martinez-Millana, Angeles Celda, Jesus Mandingorra, Lucia Aparici-Tortajada, Zoe Valero-Ramon, Jorge Munoz-Gama, Marcos Sepúlveda, Eric Rojas, Víctor Gálvez, Daniel Capurro, and Vicente Traver. Toward value-based healthcare through interactive process mining in emergency rooms: The stroke case. *International Journal of Environmental Research and Public Health*, 16(10):1783, May 2019.
- [49] Abdulrahman Jalayer, Mohsen Kahani, Amin Beheshti, Asef Pourmasoumi, and Hamid Reza Motahari-Nezhad. Attention mechanism in predictive business process monitoring. In *24th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2020, Eindhoven, The Netherlands, October 5-8, 2020*, pages 181–186. IEEE, 2020.
- [50] Bokyoung Kang, Dongsoo Kim, and Suk-Ho Kang. Periodic performance prediction for real-time business process monitoring. *Ind. Manag. Data Syst.*, 112(1):4–23, 2011.
- [51] István Ketykó, Felix Mannhardt, Marwan Hassani, and Boudewijn F. van Dongen. What averages do not tell: predicting real life processes with sequential deep learning. In Jiman Hong, Miroslav Bures, Juw Won Park, and Tomás Cerný, editors, *SAC '22: The 37th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, April 25 - 29, 2022*, pages 1128–1131. ACM, 2022.

- [52] Muhammad Asjad Khan, Hung Le, Kien Do, Truyen Tran, Aditya Ghose, Khanh Hoa Dam, and Renuka Sindhgatta. Memory-augmented neural networks for predictive process analytics. *CoRR*, abs/1802.00938, 2018.
- [53] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [54] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. Open-Review.net, 2017.
- [55] Wai Lam Jonathan Lee, Denis Parra, Jorge Munoz-Gama, and Marcos Sepúlveda. Predicting process behavior meets factorization machines. *Expert Syst. Appl.*, 112:87–98, 2018.
- [56] Maikel Leemans. Nasa crew exploration vehicle (cev) software event log, 2017.
- [57] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Discovering block-structured process models from event logs - A constructive approach. In José Manuel Colom and Jörg Desel, editors, *Application and Theory of Petri Nets and Concurrency - 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings*, volume 7927 of *Lecture Notes in Computer Science*, pages 311–329. Springer, 2013.
- [58] Artini M. Lemos, Caio C. Sabino, Ricardo M. F. Lima, and Cesar A. L. Oliveira. Using process mining in software development process management: A case study. In *2011 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, October 2011.
- [59] Li Lin, Lijie Wen, and Jianmin Wang. Mm-pred: A deep predictive model for multi-attribute event sequence. In Tanya Y. Berger-Wolf and Nitesh V. Chawla, editors, *Proceedings of the 2019 SIAM International Conference on Data Mining, SDM 2019, Calgary, Alberta, Canada, May 2-4, 2019*, pages 118–126. SIAM, 2019.

- [60] Rafael Lorenz, Julian Senoner, Wilfried Sihm, and Torbjørn H. Netland. Using process mining to improve productivity in make-to-stock manufacturing. *Int. J. Prod. Res.*, 59(16):4869–4880, 2021.
- [61] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
- [62] R. Duncan Luce. *Individual Choice Behavior: A Theoretical Analysis*. Wiley, 1959.
- [63] Felix Mannhardt. Sepsis cases - event log, 2016.
- [64] Felix Mannhardt, Massimiliano de Leoni, and Hajo A. Reijers. Extending process logs with events from supplementary sources. In Fabiana Fournier and Jan Mendling, editors, *Business Process Management Workshops - BPM 2014 International Workshops, Eindhoven, The Netherlands, September 7-8, 2014, Revised Papers*, volume 202 of *Lecture Notes in Business Information Processing*, pages 235–247. Springer, 2014.
- [65] Alfonso Eduardo Márquez-Chamorro, Manuel Resinas, and Antonio Ruiz-Cortés. Predictive monitoring of business processes: A survey. *IEEE Trans. Serv. Comput.*, 11(6):962–977, 2018.
- [66] B.W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442–451, October 1975.
- [67] B.W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442–451, 1975.
- [68] Nicola Di Mauro, Annalisa Appice, and Teresa M. A. Basile. Activity prediction of business process instances with inception CNN models. In Mario Alviano, Gianluigi Greco, and Francesco Scarcello, editors, *AI*IA 2019 - Advances in Artificial Intelligence - XVIIIth International Conference of the Italian Association for Artificial Intelligence, Rende, Italy, November 19-22, 2019, Proceedings*, volume 11946 of *Lecture Notes in Computer Science*, pages 348–361. Springer, 2019.

- [69] Nijat Mehdiyev, Joerg Evermann, and Peter Fettke. A multi-stage deep learning approach for business process event prediction. In Peri Loucopoulos, Yannis Manolopoulos, Oscar Pastor, Babis Theodoulidis, and Jelena Zdravkovic, editors, *19th IEEE Conference on Business Informatics, CBI 2017, Thessaloniki, Greece, July 24-27, 2017, Volume 1: Conference Papers*, pages 119–128. IEEE Computer Society, 2017.
- [70] Nijat Mehdiyev, Joerg Evermann, and Peter Fettke. A novel business process prediction model using a deep learning method. *Bus. Inf. Syst. Eng.*, 62(2):143–157, 2020.
- [71] Clara Meister, Gian Wiher, and Ryan Cotterell. On decoding strategies for neural text generators. *Trans. Assoc. Comput. Linguistics*, 10:997–1012, 2022.
- [72] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.
- [73] Vincenzo Pasquadibisceglie, Annalisa Appice, Giovanna Castellano, and Donato Malerba. Using convolutional neural networks for predictive process analytics. In *International Conference on Process Mining, ICPM 2019, Aachen, Germany, June 24-26, 2019*, pages 129–136. IEEE, 2019.
- [74] Patrick Philipp, Ruben Jacob, Sebastian Robert, and Jürgen Beyerer. Predictive analysis of business processes using neural networks with attention mechanism. In *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*, pages 225–230, 2020.
- [75] Anastasiia Pika, Moe Thandar Wynn, Stephanus Budiono, Arthur H. M. ter Hofstede, Wil M. P. van der Aalst, and Hajo A. Reijers. Towards privacy-preserving process mining in healthcare. In Chiara Di Francescomarino, Remco M. Dijkman, and Uwe Zdun, editors, *Business Process Management Workshops - BPM 2019 International Workshops, Vienna, Austria, September 1-6, 2019, Revised Selected Papers*, volume 362 of *Lecture Notes in Business Information Processing*, pages 483–495. Springer, 2019.
- [76] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. OpenAI, 2018.

- [77] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.
- [78] Ricardo Ribeiro, César Analide, and Orlando Belo. Improving productive processes using a process mining approach. In Álvaro Rocha, Hojjat Adeli, Luís Paulo Reis, and Sandra Costanzo, editors, *Trends and Advances in Information Systems and Technologies - Volume 2 [WorldCIST'18, Naples, Italy, March 27-29, 2018]*, volume 746 of *Advances in Intelligent Systems and Computing*, pages 736–745. Springer, 2018.
- [79] Andreas Rogge-Solti, Ronny Mans, Wil M. P. van der Aalst, and Mathias Weske. Repairing event logs using timed process models. In Yan Tang Demey and Hervé Panetto, editors, *On the Move to Meaningful Internet Systems: OTM 2013 Workshops - Confederated International Workshops: OTM Academy, OTM Industry Case Studies Program, ACM, EI2N, ISDE, META4eS, ORM, SeDeS, SINCOM, SMS, and SOMOCO 2013, Graz, Austria, September 9 - 13, 2013, Proceedings*, volume 8186 of *Lecture Notes in Computer Science*, pages 705–708. Springer, 2013.
- [80] Anne Rozinat, R. S. Mans, Minseok Song, and Wil M. P. van der Aalst. Discovering simulation models. *Inf. Syst.*, 34(3):305–327, 2009.
- [81] Luana Ruiz, Fernando Gama, and Alejandro Ribeiro. Gated graph recurrent neural networks. *IEEE Transactions on Signal Processing*, 68:6303–6318, 2020.
- [82] Gavin A Rummery and Mahesan Niranjana. On-line q-learning using connectionist systems. Technical report, University of Cambridge, Department of Engineering, Cambridge, UK, 1994.
- [83] Riyanarto Sarno, Fernandes Sinaga, and Kelly Rossa Sungkono. Anomaly detection in business processes using process mining and fuzzy association rule learning. *J. Big Data*, 7(1):5, 2020.
- [84] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pages 593–607. Springer International Publishing, 2018.

- [85] Stefan Schöning, Richard Jasinski, Lars Ackermann, and Stefan Jablonski. Deep learning process prediction with discrete and continuous data features. In Ernesto Damiani, George Spanoudakis, and Leszek A. Maciaszek, editors, *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE 2018, Funchal, Madeira, Portugal, March 23-24, 2018*, pages 314–319. SciTePress, 2018.
- [86] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.
- [87] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [88] Dalwinder Singh and Birmohan Singh. Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, 97(Part B):105524, 2020.
- [89] Martin Spott, Detlef D. Nauck, and Paul N. Taylor. Modern analytics in field and service operations. In Gilbert Owusu, Paul O’Brien, John A. W. McCall, and Neil F. Doherty, editors, *Transforming Field and Service Operations, Methodologies for Successful Technology-Driven Business Transformation*, pages 85–99. Springer, 2013.
- [90] Nitish Srivastava and et al. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.
- [91] Ward Steeman. Bpi challenge 2013, closed problems, 2013.
- [92] Ward Steeman. Bpi challenge 2013, incidents, 2013.
- [93] Christian Sturm, Stefan Schöning, and Claudio Di Ciccio. Distributed multi-perspective declare discovery. In Robert Clarisó, Henrik Leopold, Jan Mendling, Wil M. P. van der Aalst, Akhil Kumar, Brian T. Pentland, and Mathias Weske, editors, *Proceedings of the BPM Demo Track and BPM Dissertation Award co-located with 15th International Conference on Business Process Modeling (BPM 2017), Barcelona, Spain, September 13, 2017*, volume 1920 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.

- [94] Xiaoxiao Sun, Yuke Ying, Siqing Yang, and Hujun Shen. Remaining activity sequence prediction for ongoing process instances. *Int. J. Softw. Eng. Knowl. Eng.*, 31(11&12):1741–1760, 2021.
- [95] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014.
- [96] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press Cambridge, 2018.
- [97] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9. IEEE Computer Society, 2015.
- [98] Niek Tax, Irene Teinemaa, and Sebastiaan J. van Zelst. An interdisciplinary comparison of sequence modeling methods for next-element prediction. *Softw. Syst. Model.*, 19(6):1345–1365, 2020.
- [99] Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. Predictive business process monitoring with LSTM neural networks. In Eric Dubois and Klaus Pohl, editors, *Advanced Information Systems Engineering - 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings*, volume 10253 of *Lecture Notes in Computer Science*, pages 477–492. Springer, 2017.
- [100] Farbod Taymouri, Marcello La Rosa, and Sarah M. Erfani. A deep adversarial model for suffix and remaining time prediction of event sequences. In Carlotta Demeniconi and Ian Davidson, editors, *Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021, Virtual Event, April 29 - May 1, 2021*, pages 522–530. SIAM, 2021.
- [101] Farbod Taymouri, Marcello La Rosa, Sarah M. Erfani, Zahra Dasht Bozorgi, and Ilya Verenich. Predictive business process monitoring via generative adversarial nets: The

- case of next event prediction. In Dirk Fahland, Chiara Ghidini, Jörg Becker, and Marlon Dumas, editors, *Business Process Management - 18th International Conference, BPM 2020, Seville, Spain, September 13-18, 2020, Proceedings*, volume 12168 of *Lecture Notes in Computer Science*, pages 237–256. Springer, 2020.
- [102] Julian Theis and Houshang Darabi. Decay replay mining to predict next process events. *IEEE Access*, 7:119787–119803, 2019.
- [103] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*, page 125. ISCA, 2016.
- [104] Wil M. P. van der Aalst. *Process Mining - Data Science in Action, Second Edition*. Springer, 2016.
- [105] Wil M. P. van der Aalst, Arya Adriansyah, and Boudewijn F. van Dongen. Causal nets: A modeling language tailored towards process discovery. In *CONCUR 2011 - Concurrency Theory - 22nd International Conference, CONCUR 2011, Aachen, Germany, September 6-9, 2011. Proceedings*, volume 6901 of *Lecture Notes in Computer Science*, pages 28–42. Springer, 2011.
- [106] Wil M. P. van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–1142, 2004.
- [107] Boudewijn van Dongen. Bpi challenge 2012, 2012.
- [108] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.
- [109] Ishwar Venugopal, Jessica Töllich, Michael Fairbank, and Ansgar Scherp. A comparison of deep-learning methods for analysing and predicting business processes. In

- International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021*, pages 1–8. IEEE, 2021.
- [110] Ilya Verenich. Helpdesk, 2016.
 - [111] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, May 1992.
 - [112] A. J. M. M. Weijters and Wil M. P. van der Aalst. Rediscovering workflow models from event-based data using little thumb. *Integr. Comput. Aided Eng.*, 10(2):151–162, 2003.
 - [113] Kilian Q. Weinberger, Anirban Dasgupta, John Langford, Alexander J. Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman, editors, *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 1113–1120. ACM, 2009.
 - [114] Sven Weinzierl. Exploring gated graph sequence neural networks for predicting next process activities. In Andrea Marrella and Barbara Weber, editors, *Business Process Management Workshops - BPM 2021 International Workshops, Rome, Italy, September 6-10, 2021, Revised Selected Papers*, volume 436 of *Lecture Notes in Business Information Processing*, pages 30–42. Springer, 2021.
 - [115] Yonghui Wu, Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, and et al. Google’s neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.
 - [116] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

List of Figures

Fig. 1.1	Business Process Management lifecycle —adapted from [31]—.	2
Fig. 1.2	Process mining framework (adapted from [104]).	3
Fig. 1.3	Petri net representing the behavior from the event log of Table 1.1.	6
Fig. 1.4	Example of prefixes and target variables for the trace <i>Case2032</i> in the event log depicted in 1.1. <i>[EOC]</i> represents the end-of-case token.	12
Fig. 2.1	Different event encoding schemes applied in predictive monitoring.	25
Fig. 3.1	Proposed evaluation framework for evaluating predictive monitoring approaches.	34
Fig. 3.2	Example of the Bayesian Plackett-Luce model for ranking.	47
Fig. 4.1	Process model for the event log in Table 4.1 and execution semantics—in red—for trace <i>214624</i>	55
Fig. 4.2	Node feature matrix after encoding the prefix shown on Table 4.1 using the process model from Figure 4.1.	59
Fig. 4.3	Attribute feature matrix after encoding the prefix from Table 4.1.	61

Fig. 4.4	Pipeline of the data flow in TACO. The sizes of the matrices are highlighted at each step. Here, $\mathcal{P}$ is the set of places from the Petri net process model, L is the length of the longest trace of the event log, H denotes hidden size, R is the set of attribute features, $\mathcal{A}$ is the set of activities from the event log, and Q is the set of node features.	62
Fig. 4.5	Loop overwriting problem exemplified. Encoding for places P6 to P10, temporal and attribute features omitted for the sake of clarity.	66
Fig. 4.6	Credible intervals —5%, 95% quantiles— and the expected probability of winning for the tested approaches —TACO is denoted as “Rama” in this diagram—.	69
Fig. 5.1	ASTON architecture during the training phase.	77
Fig. 5.2	Encoding of the prefix corresponding to the three first events of the trace <i>Case1934</i> from the log shown in Table 5.1. For the sake of clarity, categorical features are shown before the embedding phase.	81
Fig. 5.3	Encoding of the prefix corresponding to the trace <i>Case2032</i> from the log shown in Table 1.1 under its process model.	82
Fig. 5.4	Attention mechanism of ASTON when processing an input prefix. The circles with a + inside represent the addition operation, and the circles with a $\times$ inside represent the product between two vectors. <i>tanh</i> function omitted for clarity.	84
Fig. 5.5	ASTON architecture for the inference phase, where the encoder is omitted for clarity.	85
Fig. 5.6	Output probabilities for the next two activities of a given prefix, where the numbers in the top of the figure represent the probability of a given activity at a certain timestep. Depending on which activity is selected on timestep $t - 2$ —A, B, or C—, three different probability distributions for timestep $t - 1$ can be found out.	86
Fig. 5.7	Probability tree for a suffix of two activities and three possible predictions at each step.	87
Fig. 5.8	Credible intervals and the expected probability of winning for the tested approaches against ASTON.	91
Fig. 6.1	Pipeline of DOGE. Dotted lines depict dataflow from each timestep, while filled lines represent dataflow executed once per episode.	103

Fig. 6.2	Example of the observation encoding procedure. $\oplus$ denotes the concatenation operator. Details of the calculus of the time features can be found in Section 5.2.	104
Fig. 6.3	Credible intervals —quantiles 5% and 95%— and probability of winning for each of the evaluated approaches.	110
Fig. B.1	Validation accuracy in <i>BPI-2013-C-P</i>	134
Fig. B.2	Validation accuracy in <i>BPI-2012-W-C</i>	135
Fig. B.3	Parallel coordinate plot of the 10 best-performing hyperparameters in the Helpdesk event log.	136
Fig. B.4	Parallel coordinate plot of the 10 worst-performing hyperparameters in the Helpdesk event log.	137
Fig. B.5	Parallel coordinate plot of the 10 best-performing hyperparameters in the BPI-2012-W-C event log.	138
Fig. B.6	Parallel coordinate plot of the 10 worst-performing hyperparameters in the BPI-2012-W-C event log.	139
Fig. B.7	Parallel coordinate plot of the performance of different hyperparameters in the Helpdesk event log.	139
Fig. B.8	Parallel coordinate plot of the performance of different hyperparameters in the BPI-2012-W-C event log.	140

List of Tables

Tab. 1.1	Example of a event log with 3 different traces and 15 events. It records the exeuction of 10 different activities by 8 resources	5
Tab. 2.1	Morphological box of every possibility for each characteristic.	23
Tab. 2.2	Collected studies based on deep learning for next activity prediction.	27
Tab. 2.3	Collected studies based on deep learning for activity suffix prediction.	31
Tab. 3.1	Statistics of the event logs. Time-related measures are shown in days.	37
Tab. 3.2	Event attributes used for “data-aware” approaches.	38
Tab. 3.3	Example Plackett-Luce rankings and posterior probabilities of four example approaches.	47
Tab. 3.4	Example hierarchical bayesian tests comparing two approaches, <i>A</i> and <i>B</i>	50
Tab. 4.1	Example extracted from the BPI-2012-W-C event log.	54
Tab. 4.2	Mean accuracy of the 5-fold cross-validation. Best, second-best, and third-best approaches are highlighted in green, orange and yellow, respectively.	68
Tab. 4.3	Plackett-Luce rankings and posterior probabilities of the approaches.	68
Tab. 4.4	Accuracy cross-validation differences of TACO against other approaches. Highlighted in bold differences greater than 1%.	70

Tab. 4.5	Hierarchical bayesian tests comparing TACO (A) against Hinkka et al. (H) and Tax et al. (T). Underlined values signal statistical significance ($> 95\%$).	71
Tab. 4.6	Ablation study comparing the individual performance (accuracy) of the different components of TACO. <u>Underlined</u> values show results of the GCN model where it heavily underperforms with respect to the GRNN, while bold values show results where TACO heavily outperforms the LSTM model.	72
Tab. 4.7	Ablation study showing the performance (accuracy) of our approach using different inputs, represented by a different initial, being “A” activities, “N” activated places, “T” activated transitions, “I” time features, and “R” log attributes.	73
Tab. 5.1	Potential example of an event log from a hospital.	76
Tab. 5.2	ASTON Results of the 5-fold cross-validation for the activity suffix prediction task in terms of the Damerau-Levenshtein similarity. The best, second-best, and third-best results are highlighted in green, orange, and yellow, respectively.	90
Tab. 5.3	Plackett-Luce rankings and posterior probabilities of the tested approaches against ASTON.	91
Tab. 5.4	Hierarchical Bayesian tests per dataset: ASTON (A) vs Camargo/random (C). Underlined values show statistical significance.	92
Tab. 5.5	Hierarchical Bayesian tests for ASTON (A) vs Evermann (E). Underlined values show statistical significance.	93
Tab. 5.6	Normalized Damerau-Levenshtein performance of ASTON under several inference strategies.	94
Tab. 5.7	Average predicted suffix length for each inference strategy compared to the ground truth average suffix length.	95
Tab. 6.1	Results of the 5-fold cross-validation for the activity suffix prediction task in terms of the Damerau-Levenshtein similarity. The best, second-best, and third-best results are highlighted in green, orange, and yellow, respectively.	109
Tab. 6.2	Ranking and probability of each approach to be the best in terms of ranking.	110
Tab. 6.3	Hierarchical bayesian tests comparing DOGE (D) against ASTON. (A) and Camargo (random) (C) [22]. Statistical significant results (greater than 95%) are underlined.	112

Tab. A.1	Standard deviation of accuracy of the 5-fold cross-validation	124
Tab. A.2	Mean brier score of the 5-fold cross-validation	124
Tab. A.3	Mean weighted f1-score of the 5-fold cross-validation	125
Tab. A.4	Mean MCC of the 5-fold cross-validation	125
Tab. A.5	Mean precision of the 5-fold cross-validation	125
Tab. A.6	Mean recall of the 5-fold cross-validation	126
Tab. A.7	Accuracy measures in % for the prediction in a 64/16/20 split.	126
Tab. A.8	Brier score measures for the next activity prediction in a 64/16/20 split. . . .	126
Tab. A.9	Activity suffix prediction Damerau Levenshtein distance in a 64/16/20 split.	127
Tab. B.1	Average statistics of the process models.	130
Tab. B.2	Loop statistics of the tested process logs.	131
Tab. B.3	Fine grained loop statistics.	132
Tab. B.4	Fine grained loop statistics (cont.).	132
Tab. B.5	Performance of TACO (accuracy) using the Split Miner (S.M.) and the In- ductive Miner (I.M.).	133
Tab. B.6	Comparison of VRAM, number of parameters, and computation time of TACO using embeddings or one-hot encodings. % denotes the percentage of reduction between each pair of strategies compared.	137
Tab. B.7	Comparison of VRAM, number of parameters, and computation time of TACO using the place graph or the full adjacency matrix. OOM represents an out-of-memory error. % denotes the percentage of reduction between each pair of strategies compared.	138



In this thesis, we enhance predictive monitoring in process mining through the use of advanced deep-learning techniques. By integrating Graph Neural Networks with Recurrent Neural Networks, we learn directly from the process model while also considering event sequences. We introduce two neural models: the first aims to predict the next activity in a business process, while the second forecasts the remaining sequence of activities until the case finishes. For the latter problem, a new Reinforcement Learning model is also proposed to dynamically learn optimal activity selection strategies during training. All models are rigorously validated using real-world event logs under a novel evaluation methodology to facilitate robust and fair comparisons between different predictive monitoring approaches.