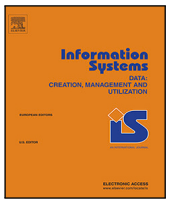
Information Systems 140 (2026) 102713

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is



Process mining between the lines: Extracting object-centric event logs from textual data

Alina Buss [a], Christoph Kecht [b,c,d,*], Wolfgang Kratsch [b,c,e], Maximilian Röglinger [b,c,d], Sareh Sadeghianasl [f], Moe T. Wynn [f]

[a] TUM School of Management, Technical University of Munich, Arcisstraße 21, 80333 München, Germany
[b] FIM Research Center for Information Management, Alter Postweg 101, 86159 Augsburg, Germany
[c] Branch Business & Information Systems Engineering of the Fraunhofer FIT, Alter Postweg 101, 86159 Augsburg, Germany
[d] University of Bayreuth, Wittelsbacherring 10, 95444 Bayreuth, Germany
[e] Technical University of Applied Sciences Augsburg, Alter Postweg 101, 86159 Augsburg, Germany
[f] Queensland University of Technology, 2 George St, Brisbane City QLD 4000, Australia

ARTICLE INFO

Recommended by Dennis Shasha

Keywords:
Process mining
Object-centric event logs
Natural language processing
Large language models
Generative artificial intelligence

ABSTRACT

Organizations generate vast amounts of unstructured textual data — a valuable source of information that frequently remains underutilized for process mining. However, textual descriptions often record exceptions and manual activities absent from structured data, and therefore, enable a better understanding of deviations from the expected business process behavior. Importantly, unstructured sources typically retain the object-centric characteristics of real-world processes — information that gets flattened or lost in case-centric event logs. Yet, existing approaches primarily target structured data sources or produce case-centric event logs. To address this gap, we present an automated approach to derive object-centric event logs directly from unstructured textual descriptions. The approach comprises two subcomponents: a *collector* that identifies events and objects (including their attributes and relationships), and a *refiner* that consolidates and cleans the extracted information. We instantiate each subcomponent in heuristic and generative implementations and create four pairwise combinations of collector and refiner instances to assess the effectiveness of heuristic natural language processing and generative artificial intelligence techniques. We compare these variants quantitatively and qualitatively in a controlled, artificial setting based on synthesized texts and demonstrate the practical utility on two naturally occurring corpora (fire status updates and a legal judgment). Our results show that the configurations with a generative collector achieve the highest extraction quality. In particular, the fully generative variant produces coherent and standardized event and object labels. Overall, this study fills a notable research gap by enabling the incorporation of textual information into process mining applications.

1. Introduction

Process mining analyzes and improves business processes by deriving insights from real-life event data. The starting point of all process mining activities is event logs, which are detailed records of events that capture the sequence and context of activities taken within a business process. Given their foundational role, the accurate extraction of event logs is paramount for the success of subsequent process mining procedures [1,2].

Most existing approaches focus on extracting event logs from structured data in organizations' core information systems, such as enterprise resource planning systems and process-aware information systems [3,4]. However, a substantial and rapidly growing amount of data is processed outside these systems in unstructured formats such as emails, contracts, and social media feeds [5,6]. Especially edge cases and manual activities are often handled in unstructured formats, leading to deviations from the expected business process behavior [7,8]. Consequently, extracting event logs from unstructured data, such as textual descriptions, has significant potential to advance process mining [3] by providing a more comprehensive representation of real-world processes [9], omitting blind spots [4], and allowing context-related insights [10]. Extant research showed how such textual descriptions can be analyzed using Natural Language Processing (NLP) techniques [3,11,12].

* Corresponding author at: FIM Research Center for Information Management, Alter Postweg 101, 86159 Augsburg, Germany.
E-mail addresses: alina.buss@tum.de (A. Buss), christoph.kecht@fim-rc.de (C. Kecht), wolfgang.kratsch@fim-rc.de (W. Kratsch), maximilian.roeglinger@fim-rc.de (M. Röglinger), s.sadeghianasl@qut.edu.au (S. Sadeghianasl), m.wynn@qut.edu.au (M.T. Wynn).

https://doi.org/10.1016/j.is.2026.102713
Received 20 October 2025; Received in revised form 26 February 2026; Accepted 26 February 2026
Available online 6 March 2026

Furthermore, real-world processes often exhibit object-centric characteristics typically reflected in textual descriptions. For instance, in a recruitment process, an event can involve multiple objects such as different applicants, applications, and vacancies [13]. Traditional, case-centric formats like the eXtensible Event Stream (XES) standard struggle to capture these object-centric characteristics due to simplifying assumptions that flatten multidimensional data to a single case representation. This flattening distorts the representation of event data and causes the loss of crucial information, which can ultimately lead to issues of deficiency, convergence, and divergence [14,15]. In response to these issues, advanced Object-Centric Event Log (OCEL) formats such as the Object-Centric Event Data (OCED) meta-model [16] and the OCEL 2.0 format [17] have been proposed recently [13,18]. However, to the best of our knowledge, no existing approaches aim to capture OCELs from unstructured textual descriptions.

To tackle this research gap, we explore the potential of heuristic NLP and generative Artificial Intelligence (AI) techniques to automatically extract OCELs from unstructured textual descriptions. Therefore, we develop an approach and instantiate it in different variants following the Design Science Research (DSR) methodology by Peffers et al. [19]. Our approach incorporates three Design Objectives (DOs) and comprises two key subcomponents: a *collector* that identifies events and objects (including their attributes and relationships), and a *refiner* that consolidates and cleans the extracted information. Both subcomponents are instantiated in heuristic and generative forms, resulting in four distinct combinations, referred to as extractor variants.

Our results show that the configurations with a generative collector achieve the highest extraction quality (F1-score of 0.67 combined with a generative refiner, F1-score of 0.66 combined with a heuristic refiner) and generalization capabilities on unseen data. In contrast, the variants with a heuristic collector underperform (F1-score of 0.53 combined with a generative refiner, F1-score of 0.54 combined with a heuristic refiner). However, the former configurations significantly differ in the formatting and interpretability of the outputs. Notably, the generative collector combined with a generative refiner provides coherent and standardized event and object labels, leading to cohesive Object-Centric Directly-Follows Graphs (OCDFGs) for further analysis.

This article builds upon our prior short conference paper [20] by significantly expanding both the technological and methodological scope of the work. On the technological side, we entirely redesigned the implementation of the generative refiner, which is now implemented as a sequence of Large Language Model (LLM)-guided normalization steps with deterministic post-processing, thereby replacing the single-prompt approach used in the conference version. We also upgraded the underlying LLM of both generative components from `gpt-4o-mini-2024-07-18` to `gpt-5-mini-2025-08-07`, thus harnessing the potential of its underlying reasoning capabilities. Informed by the results of the conference paper, these changes substantially improve the extraction quality and generalization capabilities.

On the methodological side, we embed the artifact development in the DSR methodology by Peffers et al. [19] and evaluate our approach more comprehensively according to the framework by Sonnenberg and vom Brocke [21] through four evaluation activities, referred to as EVAL1 to EVAL4. Having identified a lack of existing approaches to extract OCELs from unstructured textual descriptions in EVAL1, we now include a competing artifact analysis with 18 related approaches resulting from a structured literature review in EVAL2 to demonstrate the novelty of our approach. Beyond assessing the extraction quality of the four extractor variants on synthetic textual descriptions derived from six publicly available OCEL datasets (as in the conference paper), EVAL3 complements the quantitative results with a qualitative discussion along seven criteria. In addition, we extend EVAL3 with a measurable proxy for process inference from text by comparing Directly-Follows Graphs (DFGs) derived for each object type between the original and extracted logs. Furthermore, and in contrast to the purely synthetic evaluation in the conference paper, we validate the real-world applicability of the two best-performing extractor variants in EVAL4 by exposing them to two types of naturally occurring textual descriptions: fire status updates and a legal judgment. Altogether, these extensions provide a more holistic and naturalistic evaluation of the approach and demonstrate its generalizability beyond the conditions examined in the conference paper.

The contribution of our study is threefold. First, it presents a novel approach for automatically extracting OCELs directly from unstructured textual descriptions, filling a notable gap in process mining research by enabling the incorporation of textual data that captures complex, object-centric process behavior. Second, implementing and evaluating four distinct extractor variants offers a systematic comparative analysis of heuristic NLPs and generative AI techniques in this context. Third, to foster reproducibility and future research, we publish our implementation and datasets publicly available in a GitHub repository.[1]

The remainder of this paper is structured as follows: Section 2 introduces the theoretical background essential for understanding the approach, followed by a comprehensive description of the research design in Section 3. Section 4 elaborates on the DOs (Section 4.1), the general OCEL extraction approach (Section 4.2), and its instantiation in four extractor variants (Section 4.3). The approach and the instantiated extractor variants are subsequently evaluated in Section 5. Finally, Section 6 concludes the paper with a summary and opportunities for future work.

2. Theoretical background

2.1. Business process management, process mining, and object-centric event logs

Business Process Management (BPM) is a structured approach to improve the efficiency, effectiveness, and adaptability of organizational workflows. By combining methodologies and technologies to model, automate, monitor, and optimize business processes, BPM aligns operational activities with strategic objectives [22,23]. This alignment fosters continuous improvement and operational excellence while supporting cost reduction, quality enhancement, and increased customer satisfaction [23,24]. One particularly powerful technique in BPM is process mining, which has gained prominence for its data-driven approach [25].

While "traditional BPM techniques [...] use hand-made models [26], process mining is based on facts" [1, p. 2] using knowledge extracted from real-life event data. This event data typically resides in organizations' core information systems and can be converted into event logs, the essential prerequisite for process mining. Process mining can be divided into four main tasks: Process discovery, conformance checking, process enhancement [27], and operational support [2]. However, the starting point for all process mining tasks is event logs, making their proper extraction essential.

Traditional case-centric event logs consist of multiple traces, each containing numerous events. Every event corresponds to a specific activity and is related to a particular case. Events within a case follow a sequential order, forming a single execution path of the process. Additionally, event logs can capture further details about events, such as the resource (e. g., a person or device) responsible for executing or initiating the activity, the event's timestamp, and relevant attributes [1]. Following this definition, the XES format became the industry standard for representing event logs in the case-centric process mining field [28].

However, in recent years, there has been a paradigm shift from case-centric process mining to Object-Centric Process Mining (OCPM) [13,18]. This shift is due to real-life processes being more complex than the simplifying assumptions of the XES standard suggest. Flattening

[1] https://github.com/Alinabuss/OCEL_extractor_v2.

multidimensional event data to a single case representation, as required by the XES standard, leads to a distorted representation of event data and to a loss of important information that can ultimately result in deficiency, convergence, and divergence problems [14,15].

To resolve these problems, the OCPM field deviates from the simplifying assumptions of the XES standard by allowing an event to refer to multiple objects rather than a single case [13]. This paradigm shift led to the development of numerous OCEL formats [29] such as the eXtensible Object-Centric (XOC) format by Li and de Carvalho [30], the OCEL 1.0 format by Ghahfarokhi et al. [31], the OCED meta model by the OCED Working Group of the IEEE Task Force on Process Mining [16], and the OCEL 2.0 format by Berti et al. [17]. As the OCEL 2.0 format provides a wide range of tool support as well as exemplary datasets and is compatible with most existing process mining-specific programming libraries [32], we opt for this event log format for the instantiation of our artifact.

The OCEL 2.0 format is a comprehensive standard for OCPM that consists of two main components: objects and events. Objects represent entities involved in processes, such as physical items (e. g., products or machines), abstract entities (e. g., orders or contracts), or individuals (e. g., employees or suppliers). Each object is categorized by an object type, such as product, order, or supplier. These object types define specific attributes with multiple dynamic values that can change over time. Object-to-Object (O2O) relationships represent structural or functional associations between objects, independent of specific events. For example, an employee might belong to an organizational unit, or an item could be part of a particular shipment. O2O relationships are not limited to binary connections; they can include qualifiers that define the type or nature of the relationship, such as part-of, reports-to, or belongs-to.

Events in OCEL 2.0 represent discrete actions or occurrences within a process, such as approving an order, shipping an item, or making a payment. Each event is uniquely identified by an event ID and characterized by an event type that categorizes its nature. For instance, in a procurement process, event types might include "Order Created", "Invoice Sent", or "Payment Approved". Events are atomic, meaning they occur at a specific time and do not span a duration. Each event type can also include additional attributes that provide further details about the event, with the flexibility to accommodate multiple values for each attribute.

Furthermore, the OCEL 2.0 format captures relationships between events and objects through Event-to-Object (E2O) relationships. These relationships describe how objects participate in events and how events affect objects [17]. For example, an interview event might involve several recruiter objects, and the format allows roles to be defined using qualifiers. Therefore, the qualifiers can distinguish between roles, such as identifying the HR recruiter versus the technical recruiter.

2.2. Natural language processing

NLP is a multidisciplinary field that combines linguistics, computer science, and AI to enable computers to understand, interpret, and generate human language. It encompasses a wide range of tasks and applications, including speech recognition, text classification, sentiment analysis, machine translation, and question-answering systems [33,34] that have become increasingly prevalent in our daily lives [35,36].

NLP encompasses two primary subfields: Natural Language Understanding (NLU) and Natural Language Generation (NLG). These subfields form the foundation of NLP, with NLU focusing on processing and interpreting human language input, and NLG dealing with producing human-like language output based on structured data or machine representations [33]. Even though these concepts are distinct, they often complement each other [37]. NLU as well as NLG can be realized through a heuristic pipeline approach or an advanced end-to-end neural approach [38].

Within NLU, a heuristic pipeline approach relies on predefined linguistic rules and sequential processing stages to support the decomposition of language into manageable components. Popular processing stages include tokenization, lemmatization, Part-of-Speech (PoS) tagging, dependency parsing, and Named Entity Recognition (NER). Tokenization breaks down text into smaller units called tokens, which are usually words or subwords [39,40]. Lemmatization reduces words to their base or dictionary form, considering the context and meaning of words. Unlike stemming, which removes affixes, lemmatization considers words' grammatical role, ensuring accurate normalization [41]. PoS tagging assigns grammatical categories, such as nouns, verbs, or adjectives, to words in a sentence [42]. Dependency parsing examines the grammatical relationships between words in a sentence, forming a directed graph where nodes represent words and edges represent syntactic dependencies [43,44]. NER identifies and classifies named entities, such as person names, organizations, and locations within text [45,46]. Furthermore, Regular Expressions (REGEX) provide a flexible way to search for and manipulate text based on specific patterns [47]. A widespread tool for conducting NLU according to this approach is the Python library "SpaCy" [48], which offers high-performance solutions for building and customizing language processing pipelines that include these stages [49].

Within NLG, a heuristic pipeline approach also represents a structured, modular method of generating text through carefully defined sequential stages. It begins with text planning, where content is selected and discourse structure is arranged. Subsequent sentence planning involves aggregating sentence content, making lexical choices, and constructing referring expressions. The final linguistic realization stage performs orthographic, morphological, and syntactic processing to transform structured information into coherent text [50].

In both cases, a pipeline approach offers several advantages but also comes with notable challenges. One key benefit is the interpretability, as each stage of the pipeline is transparent and can be analyzed independently. Furthermore, the modularity of the pipeline allows components to be easily replaced or updated without disrupting the entire system. Additionally, domain-specific customization is facilitated, enabling the integration of tailored rules and knowledge to address specific requirements [51,52]. However, error propagation is a significant issue, as mistakes in earlier stages can compound through later ones. The pipeline approach also faces difficulties in handling complex language phenomena, such as context-dependent meanings and idiomatic expressions, limiting its ability to capture the nuances of human language. Moreover, the approach is often resource-intensive, demanding substantial annotated data and computational power to perform optimally across diverse languages and domains. Finally, the pipeline typically relies on manual feature engineering, requiring extensive linguistic expertise to design effective features, which can be time-consuming and labor-intensive [51].

An end-to-end neural approach, by contrast, uses LLMs to process language directly from input to output without intermediate representations for both NLU and NLG. These systems utilize encoder–decoder architectures like transformers [53] to learn mappings between human language and its corresponding structured representations from large datasets [50,54]. Self-attention mechanisms enable transformers to capture relationships between words in a sentence and process entire sequences in parallel, improving training efficiency and performance on long-range dependencies [53]. Prominent transformer-based LLMs include OpenAI's Generative Pre-trained Transformer (GPT) models [55]. Developers can interact with these models using user and system prompts and utilize vector memory to store and retrieve information for context-aware responses [56]. While current LLMs generate remarkably fluid, natural-sounding text, thereby reducing manual engineering requirements and increasing adaptability across different domains, LLMs also present challenges. They are computationally intensive, require substantial resources, and function as "black boxes", making the control of the generation process and the interpretation

of the output difficult. Additionally, they may suffer from biases in training data, Application Programming Interface (API) limitations, and inaccuracies due to hallucinations [57–59].

Recognizing the unique strengths and limitations of both approaches, contemporary research increasingly explores hybrid methodologies for NLU and NLG. These innovative techniques aim to combine the structural precision of traditional pipeline approaches with the adaptive learning capabilities of neural networks while mitigating the weaknesses of the underlying implementation. Furthermore, by combining techniques from both NLU and NLG, modern NLP systems can achieve sophisticated language understanding and generation capabilities, enabling applications such as chatbots [60], machine translation, and automated content generation [37,61].

2.3. Related work

The application of NLP in BPM and process mining has been widely explored. Within BPM, for instance, the integration of NLP enables the extraction of process-related information from unstructured data sources such as emails, documents, and communication logs [6,62]. For instance, Di Ciccio et al. [63] demonstrate how email messages can be automatically analyzed to construct workflow models of knowledge-intensive processes. Similarly, Delicado et al. [64] propose the web application NLP4BPM, which enables the conversion and comparison of textual descriptions into business process models and vice versa. By uncovering informal and ad-hoc processes often absent in structured systems, NLP provides a more comprehensive and realistic representation of organizational workflows [62].

Furthermore, NLP techniques have been used for all process mining tasks. For instance, Moon et al. [65] use a transformer model for automatic process discovery, while García-Bañuelos et al. [66] communicate the results of the conformance checking task in natural language. Furthermore, Ramos-Gutiérrez et al. [67] use different NLP techniques to enhance the event log quality, and Cabrera et al. [68] utilize contextualized word embeddings for predictive process monitoring. First attempts, such as Rebmann et al. [69] and Berti and Qafari [70], even apply NLP techniques in the object-centric field, proving the compatibility of these research streams, and thereby paving the way for future research at the intersection of OCPM and NLP.

Recent process mining research also leveraged NLP techniques to extract and enhance event logs, demonstrating the effectiveness of these techniques. For instance, Banziger et al. [11] apply Latent Dirichlet Allocation (LDA) algorithms to analyze Customer Relationship Management (CRM) interactions in emails, appointments, calls, and tasks by identifying predominant topics and constructing XES logs. Similarly, Kecht et al. [3] use supervised Natural Language Inference (NLI) algorithms to extract activities from Twitter customer service conversations. In contrast, Geeganage et al. [12] employ REGEX, NER, and dependency parsing on unstructured text fields of existing XES logs to enhance the logs. While these prior approaches effectively construct or enhance XES event logs, deriving OCELs entails a higher degree of technical complexity. Whereas XES event logs can be created or refined on the case-level, OCELs must be holistically amended. For example, object instances and their lifecycle can relate to an arbitrary number of events, and the proper assignment to parental object types is crucial for meaningful subsequent analysis. Therefore, deriving OCELs from unstructured data requires novel techniques beyond existing XES event log extraction and enrichment approaches.

3. Research design

To address the previously identified gap in the literature, we follow the DSR methodology by Peffers et al. [19] and develop an approach for extracting OCELs from unstructured textual descriptions. We instantiate the approach in four distinct extractor variants, thus leveraging heuristic NLP and generative AI techniques to different extents. Fig. 1 illustrates how our paper aligns with the six steps of the DSR methodology by Peffers et al. [19].

Problem identification and motivation: We entered the DSR process by the problem-centered initiation in Sections 1 and 2. In these sections, we identified a lack of approaches for automatically extracting OCELs from unstructured textual descriptions, despite the vast potential of unstructured data to enhance process mining by providing insights into edge cases and manual activities.

Objectives of a solution: Based on the problem definition, we derive three DOs in Section 4.1 that will guide the development of our approach.

Design and development: Initially, we develop an approach to extract OCELs from textual descriptions by utilizing heuristic NLP and generative AI techniques. Guided by our development framework (see Section 4.3.1) that enables continuous validation of the extraction capabilities, we instantiate the approach in four distinct extractor variants. Within the framework, we develop a generator instance (to synthetically generate textual descriptions based on publicly available OCELs), instances of the four extractor variants, and a comparison instance (to compare the extracted OCELs with the initial OCELs). The instantiation phase incorporates several validation stages that enable rapid identification of weaknesses and inaccuracies, facilitating swift refinements in subsequent iterations. The proposed approach will be presented in Section 4.2, followed by a detailed documentation of its instantiation in Section 4.3.

Demonstration: To showcase the extraction capabilities of our four extractor variants, we apply each extractor variant on synthetic textual descriptions derived from six publicly available OCEL datasets, each dataset containing 1000 events. The results of this demonstration step will be discussed in further detail in the subsequent evaluation activity in Section 5.3. In addition, we demonstrate the approach on naturally occurring textual descriptions in Section 5.4 by applying the two best-performing variants to fire status updates and to a legal judgment, and by visualizing the extracted OCELs as OCDFGs using the Python library PM4Py.

Evaluation: A comprehensive evaluation of the general approach and its final extractor variants is conducted in Section 5. We structure the evaluation along the framework by Sonnenberg and vom Brocke [21], comprising four evaluation activities, referred to as EVAL1 to EVAL4. In Sections 1 and 2, we identified a lack of approaches to automatically extract OCELs from unstructured textual descriptions (EVAL1). We then conduct a competing artifact analysis in EVAL2 with 18 related approaches resulting from a structured literature review to validate the design specification of our approach. In EVAL3, we quantitatively assess the extraction capabilities of the four extractor variants at the log level and process level using synthetic textual descriptions derived from six publicly available OCEL datasets and used during the demonstration phase. Furthermore, we discuss the real-world applicability of the different extractor variants qualitatively based on seven criteria, incorporating insights from the demonstration phase. Lastly, in EVAL4, we validate the real-world applicability of the two best-performing extractor variants against two types of naturally occurring textual descriptions, namely fire status updates and a legal judgment.

Communication: We share the resulting artifact and our findings throughout this paper. The implemented extractor variants are publicly available on GitHub and can be reused by the community.

4. Design specification

4.1. Design objectives

(DO1) Event log extraction in an object-centric format. Event logs are the foundational element for all process mining activities, applicable to both traditional and object-centric process mining domains. Consequently, the accurate extraction of these logs is crucial for the

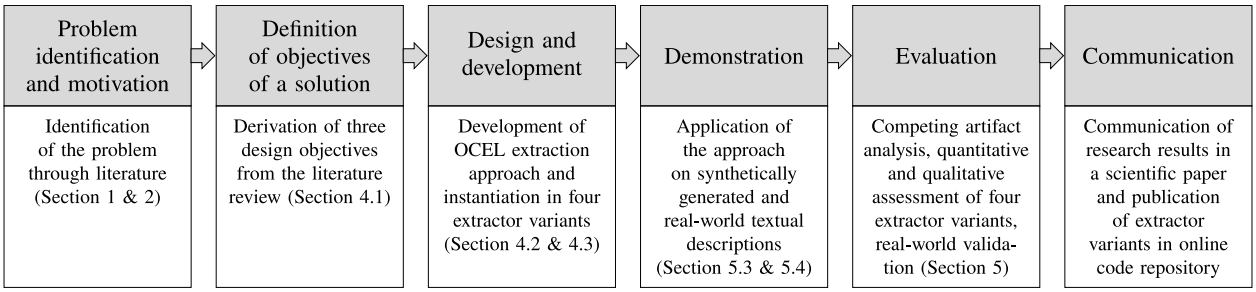


Fig. 1. DSR process based on Peffers et al. [19].

success of subsequent process mining procedures [27]. However, traditional event logs, such as those in XES format, flatten multi-dimensional event data to a single case representation, potentially leading to issues such as deficiency, convergence, and divergence [13,14]. In contrast, OCEL formats address these challenges by preserving multi-dimensional data structures [15]. Potential OCEL formats include the OCED metamodel by Fahland et al. [16] and the OCEL 2.0 format by Berti et al. [17].

(DO2) Event log extraction from unstructured textual descriptions. Most existing approaches to event log extraction concentrate primarily on constructing event logs from structured data within organizations' core information systems. However, a significant amount of valuable data is generated and processed outside these systems [3]. To date, unstructured data accounts for approximately 80 to 90 percent of all available data and is growing at a rate three times faster than structured data [5,71,72]. Furthermore, edge cases and manual activities are often handled in unstructured formats, which, if overlooked, can lead to deviations from the expected business process behavior [7,8]. This unstructured data, which includes formats such as emails, free text fields, chat logs, and contracts, remains largely untapped despite its potential for process mining [3,6]. Consequently, the extraction of event logs from unstructured data, such as textual descriptions, has significant potential to advance process mining [3] by providing a more comprehensive representation of real-world processes [9], omitting blind spots [4], and enabling context-related insights [10].

(DO3) Event log extraction at scale, fully automated, and without human intervention. Given organizations' substantial volumes of unstructured data, developing an automatic and scalable extraction approach is imperative to effectively manage and process this information [73]. Approaches depending on human intervention are impractical with such extensive datasets [74]. Moreover, minimizing the reliance on expert knowledge within the system enhances flexibility, allowing the extraction process to adapt to different data sources and contexts without requiring extensive reconfiguration. It also improves usability by making the system more accessible to non-expert users, thus enabling a wider range of individuals to interact with and benefit from the extraction process without specialized training. This broader accessibility can facilitate faster adoption, reduce operational costs, and increase overall efficiency in handling large-scale unstructured data [75,76].

4.2. Object-centric event log extraction approach

Stemming from these DOs, we develop an approach to extract OCELs from unstructured textual descriptions. The approach is designed to be universally applicable across domains, minimizing the need for human intervention by leveraging heuristic NLP and generative AI techniques, while enabling an automated extraction process that efficiently handles large datasets. The extractor comprises two subcomponents: a collector and a refiner. Fig. 2 illustrates their functionality.

Initially, textual descriptions of arbitrary length are iteratively imported by the collector subcomponent. The collector aims to extract relevant information from each description and structure it into a preliminary OCEL format. Processing each description in isolation ensures that relevant details are captured without interference from other descriptions, making the approach more feasible for real-world applications. Working with smaller subsets reduces execution time and computational cost, creating a more efficient process. This incremental approach is also more realistic, as textual descriptions can be added progressively instead of requiring all data to be gathered upfront.

Once these preliminary OCEL snippets are gathered, they are passed to the refiner subcomponent, whose goal is to ensure the coherence and quality of the final output. Therefore, the refiner concatenates the individual snippets into a unified version, which it then cleans, refines, and harmonizes. This step is necessary because, without refinement, inconsistencies or redundant information across descriptions could lead to misinterpretations or an incomplete representation of the process. The refiner aligns data structures and terminology across entries, enhances data quality by removing noise or errors, and ensures compatibility of all elements. This process ultimately improves the precision of the final OCEL, making it a more reliable and valuable resource for further analysis and decision-making.

4.3. Artifact instantiation

We instantiate our approach in four distinct extractor variants, leveraging heuristic NLP and generative AI techniques, to evaluate their respective strengths and weaknesses. The first extractor variant is a heuristic extractor (HEU-HEU) that features a heuristic collector and a heuristic refiner. The second variant is a generative extractor (GEN-GEN) consisting of a generative collector and a generative refiner. Lastly, we develop two hybrid extractors, namely a GEN-HEU extractor, employing a generative collector and a heuristic refiner, and a HEU-GEN extractor, consisting of a heuristic collector and a generative refiner. Fig. 3 illustrates the four extractor variants.

4.3.1. Development framework

The four extractor variants are developed sequentially within a development framework that enables continuous validation of their extraction capabilities. Fig. 4 illustrates the three components of the framework: a generator instance, an extractor instance, and a comparison instance.

Generator Instance: Initially, a test subset consisting of an original OCEL is provided to the generator instance of the development framework. The generator instance is then tasked with converting the events of the OCEL into corresponding textual descriptions. The generator instance can generate textual descriptions at three levels of complexity. At Complexity Level 1, it creates one textual description per event, while at Complexity Level 2, it generates non-overlapping daily reports by grouping events per day. Lastly, at Complexity Level 3, it generates overlapping reports in which events are grouped by their related objects. To ensure the comparability of the results with our prior conference paper [20], we use the same textual descriptions (generated with OpenAI's `gpt-4o-mini-2024-07-18` LLM) in this article. A sample textual description at Complexity Level 3 based on a logistics log appears as follows:

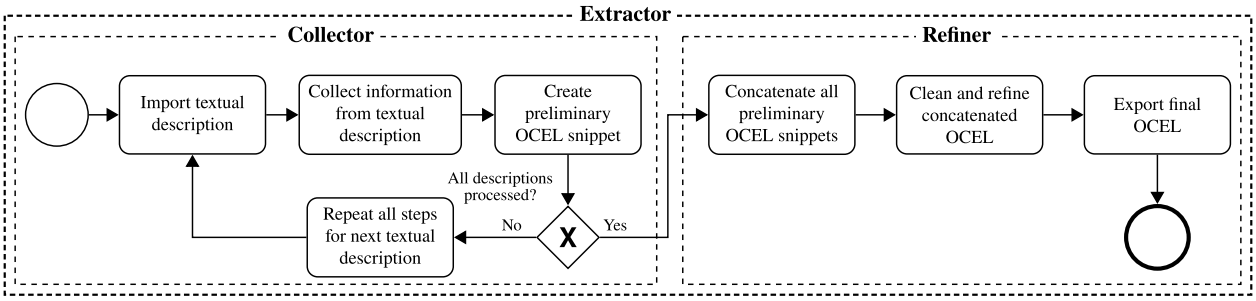


Fig. 2. A novel approach to extract OCELs from textual descriptions.

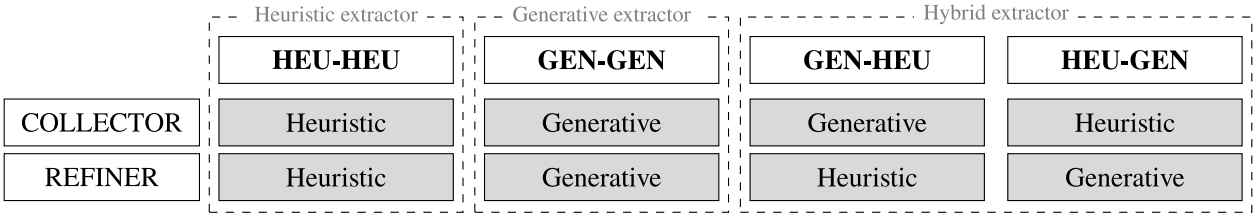


Fig. 3. Four extractor variants.

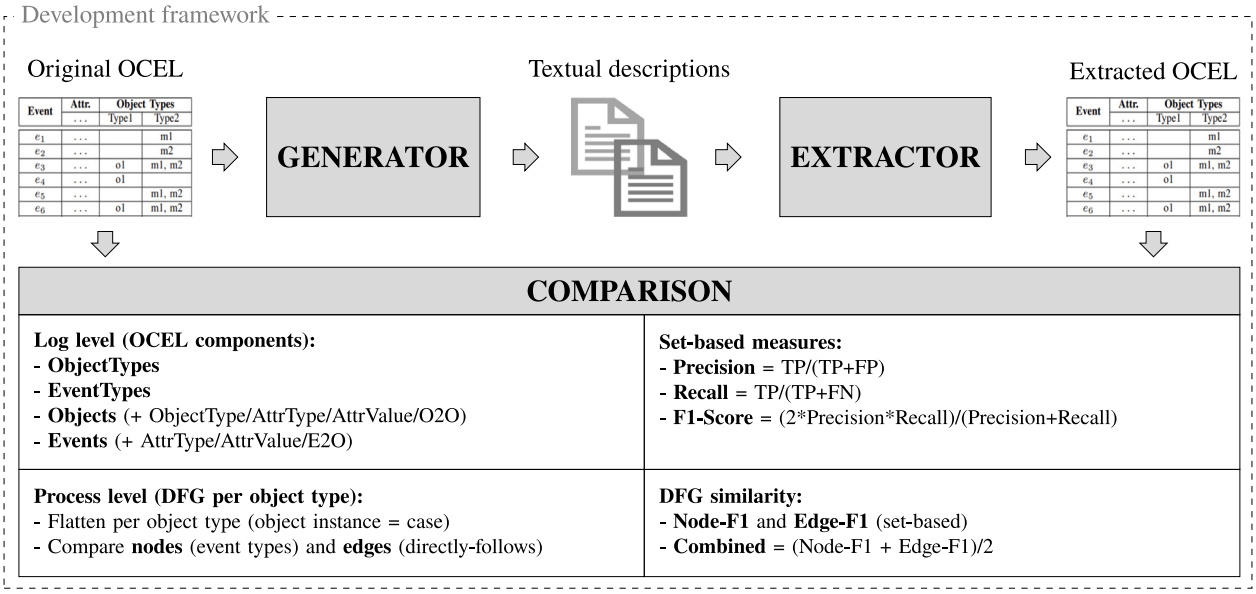


Fig. 4. Development framework.

Report on Container ID cr3

On May 25, 2023, a container with ID cr3 was initially picked up at 00:23:40 UTC. At that point, it was in an empty state, carrying 6.0 handling units and weighing 1210.0 kg. Subsequently, at 07:37:58 UTC, a truck with ID tr4 loaded the container cr3, which was still in an empty state and had the same specifications (6.0 handling units and 1210.0 kg). At the time of loading, container cr3 contained a handling unit with ID hu9. Later that day, at 07:41:41 UTC, truck tr4 loaded handling unit hu8 along with container cr3, which was now marked as full.

Extractor Instance: The generated synthetic textual descriptions are then handed over to the extractor instance of the development framework. The extractor instance, corresponding to our four extractor variants, is then tasked with analyzing the provided textual descriptions to reconstruct the original OCEL. Each extractor variant, therefore, leverages its respective heuristic and generative subcomponents. A detailed description of how the different extractor variants extract the OCELs from the textual descriptions will be provided in Sections Section 4.3.2, 4.3.3, and 4.3.4. As a result, one extracted OCEL is created for each original OCEL.

Comparison Instance: Lastly, the extracted OCELs are compared with their original counterparts using the comparison instance, which evaluates the alignment of the logs across various categories and levels of detail. The levels of detail, comprising parent levels and absolute and relative child levels, follow the structure of the OCEL 2.0 format. At the parent level, categories such as object types, event types, object instances, and event instances are analyzed to ensure the existence of corresponding values in the extracted logs. Meanwhile, the child-level assessments determine whether specific child values are accurately mapped to their parent categories. For example, the comparison verifies whether object types, attribute types, and attribute values are correctly

linked to object labels, and whether the appropriate O2O and E2O relationships are identified. This analysis is performed both on an absolute level (across the entire event log) and a relative level (assuming the parent category is correctly identified).

Various metrics are applied for in-depth analysis, including confusion-matrix-based measures such as precision, recall, and F1-score. Originally, confusion matrices were used to evaluate a classification model by comparing predicted values with actual ones [77]. In our use case, we follow the interpretation by Derczynski [78] for entity extraction systems in the NLP domain. Here, True Positives (TPs) refer to correctly extracted entities, False Negatives (FNs) to entities that were missed, and False Positives (FPs) to superfluous entities that were incorrectly included, with the original log serving as the ground truth. For such systems, True Negatives (TNs) are excluded since the notion of correctly not extracting entities is irrelevant and would be misleading if included. Based on these values, we calculate precision, recall, and F1-score. Precision measures the proportion of correct identifications among retrieved entities, while recall quantifies the number of relevant entities from the original dataset that were successfully retrieved. Since precision and recall often trade off against each other, the F1-score balances these metrics by calculating their harmonic mean [78]. To report the overall performance on a particular dataset, we calculate the overall precision and recall by averaging across all parent and absolute child levels, and the overall F1-score as the harmonic mean of the resulting precision and recall.

By doing so, the development framework enables the continuous validation of the extraction capabilities of the different extractor variants during their instantiation phase. However, unlike typical binary classification problems, where a random classifier can achieve an F1-score of 0.5 on a balanced dataset, extracting OCELs from textual descriptions is an open-ended task with an unrestricted solution space. Due to significant noise and irrelevant terms in the textual descriptions, the dataset is highly skewed, with a very low proportion of relevant words. Moreover, beyond merely identifying relevant words, these words must be correctly assigned to specific OCEL components, making this a challenging multi-class classification problem. Additionally, the extracted values must be mapped accurately to one another to construct the final OCEL structure. This combined complexity makes the extraction task difficult; a random extractor would yield values close to 0 rather than 0.5. Given these challenges, we set a validation threshold of 0.5, requiring a non-trivial balance of precision and recall that yields OCELs of sufficient quality for further analysis.

In addition to the component-based alignment, the comparison instance also provides a process-level perspective on whether the extracted logs preserve the original process behavior. Specifically, for each object type, we flatten the original and extracted OCELs into case-centric event logs using the object type as the case notion, then derive a DFG from each flattened log, and finally compare the two DFGs on both their node set (event types) and edge set (directly-follows relations) using precision, recall, and F1-score. We define the combined DFG similarity for an object type as the arithmetic mean of node-F1 and edge-F1. To ensure comparability despite differing labels, we reuse the object-type and event-type mappings already established during the aforementioned OCEL comparison. Event types that occur only in the extracted log are treated as additional granularity and excluded from this process-level score, whereas missing event types are reflected as false negatives, thereby lowering recall. We aggregate the DFG similarity across object types, where weights correspond to the number of object instances per type in the original OCEL.

4.3.2. Heuristic extractor

Leveraging the described development framework, we start with instantiating the heuristic HEU-HEU extractor. The heuristic extractor, consisting of a heuristic collector and a heuristic refiner, follows a heuristic NLP pipeline approach that depends on predefined rules to extract OCELs from textual descriptions. As all these rules must be manually defined, their development spans nine Design Iterations (DIs), each progressively more complex. This progression is structured into three complexity levels, each applied to three distinct training event logs and one level Closing Validation (CV) log, as illustrated in Table 1.

Table 1
Development logic of the HEU-HEU extractor.

	Event logs			
	(1) Recruitment	(2) Logistics	(3) P2P	(4) Order management
Level 1	DI 1.1	DI 1.2	DI 1.3	Level 1 CV
Level 2	DI 2.1	DI 2.2	DI 2.3	Level 2 CV
Level 3	DI 3.1	DI 3.2	DI 3.3	Level 3 CV

The three complexity levels reflect the increasing difficulty of the textual descriptions provided to the heuristic extractor. The generator instance of the development framework artificially generates these descriptions. Specifically, textual descriptions at Complexity Level 1 correspond to event-specific descriptions, while Complexity Level 2 involves non-overlapping daily reports, with events grouped by day. At Complexity Level 3, the descriptions consist of overlapping object reports, with events grouped by their related objects. This progression realistically introduces complexity by gradually increasing difficulty without overcomplicating the initial setup.

Additionally, an increase in complexity is also reflected in the selection of event logs used for training. The first event log, a recruitment log [79], includes timestamps, activities, object labels, object types, and E2O relationships but lacks object attributes, event attributes, and O2O relationships. The second event log, a logistics log [80], contains the same components as the recruitment log, with the addition of object attributes and O2O relationships, though it still lacks event attributes. The third log, a Procure-To-Payment (P2P) log [81], introduces event attributes, completing the full range of features.

The logs are divided into non-overlapping training, validation, and test subsets at the start of development. Specifically, the first 100 events from each log are allocated for training, the following 100 events for validation, and an additional 1000 events are assigned to test sets for EVAL3. In each design iteration, the training subsets of the relevant logs are used to define and refine the rules of the heuristic extractor. At the end of each iteration, validation is performed on the validation subset of the respective event log, leveraging the development framework's comparison instance. If the extractor's performance on the validation subset meets the threshold (F1-scores over 0.5 across all comparison categories), the next design iteration is initiated; otherwise, predefined rules are further refined based on the training subset. At the conclusion of each complexity level, a level CV is conducted using a fourth event log, an order management log [82], to assess the generalization capabilities of the heuristic extractor.

Leveraging this development logic, the heuristic collector and refiner are iteratively implemented and refined until they reach their final status. Fig. 5 illustrates the functionality of the final version of the heuristic extractor.

Heuristic Collector

As shown in Fig. 5, textual descriptions are iteratively provided to the heuristic collector, which is tasked with constructing preliminary OCEL snippets for the corresponding descriptions. Therefore, the heuristic collector first processes the provided textual description into an NLP matrix using a SpaCy NLP parsing pipeline [49]. This pipeline tokenizes the text and extracts key token features, including dependency labels, PoS tags, NER labels, and syntactic dependency relations such as children and ancestor tokens. For instance, the sentence "On January 15, 2023, the employee John Doe attended a training session" yields the NLP matrix shown in Table 2.

Following a set of predefined rules, the heuristic collector evaluates the tokens, their dependencies, PoS tags, and NER tags in the NLP

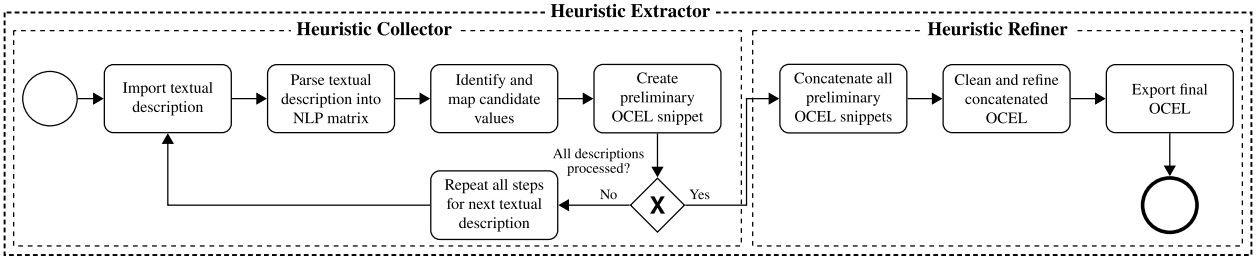


Fig. 5. Functionality of the heuristic HEU-HEU extractor.

Table 2
Exemplary NLP matrix for extracting OCEL component candidates.

Token	Dep.	PoS	NER	Children	Ancestor
On	prep	ADP		[January]	[attended]
January	pobj	PROPN	DATE	[15, ,, 2023]	[On, attended]
15	nummod	NUM	DATE	[]	[January, On, attended]
,	punct	PUNCT	DATE	[]	[January, On, attended]
2023	nummod	NUM	DATE	[]	[January, On, attended]
,	punct	PUNCT		[]	[attended]
the	det	DET		[]	[employee, attended]
employee	nsubj	NOUN		[the, Doe]	[attended]
John	compound	PROPN	PERSON	[]	[Doe, employee, attended]
Doe	appos	PROPN	PERSON	[John]	[employee, attended]
attended	ROOT	VERB		[On, employee, session]	[]
a	det	DET		[]	[session, attended]
training	compound	NOUN		[]	[session, attended]
session	dobj	NOUN		[a, training]	[attended]
.	punct	PUNCT		[]	[attended]

matrix to identify candidate values for the essential OCEL components: timestamps, activities, object labels, object types, attribute values, and attribute types. For instance, using NER tags "DATE" and "PERSON", the collector extracts "January 15, 2023" and "John Doe" as potential values for the timestamp and object label, respectively. Moreover, the combination of a "nsubj" dependency and "NOUN" PoS tag in the token "employee" indicates an object type, while the "ROOT" dependency and "VERB" PoS tag in the token "attended" identify an activity. After refining the extracted values through lemmatization, analysis of their surroundings for reference values, and filtering redundant words extracted for multiple categories, the collector outputs the following candidate values:

- `Timestamps = ["2023-01-15T00:00:00Z"]`
- `Activities = ["Attend training session"]`
- `Object Labels = ["John Doe"]`
- `Object Types = ["Employee"]`
- `Attribute Values = []`
- `Attribute Types = []`

Once candidate values are extracted, they are mapped to each other according to defined relationships. Although straightforward in this example, this mapping becomes complex when multiple values are extracted across categories. The mappings established include object labels to object types, attribute values to attribute types, object labels to other object labels to reveal O2O relationships, activities to timestamps, attributes to timestamps, object labels to activity-timestamp combinations to extract E2O relationships, and attribute values to object labels and activity-timestamp combinations. These mappings are performed using heuristic rules that evaluate the associated children and ancestor tokens of each candidate value in the NLP matrix, as well as its positional context within the text. In the current example, the token "attended" is an ancestor of the timestamp "January 15, 2023" and of the object label "John Doe". Furthermore, the object type "employee" is also an ancestor token of "John Doe", resulting in the following mappings:

- `"Attend training session"` → `"2023-01-15T00:00:00Z"`
- `"John Doe"` → `"Attend training session/2023-01-15T00:00:00Z"`
- `"John Doe"` → `"Employee"`

Based on these mappings, the heuristic collector generates a preliminary OCEL snippet per textual description. For this instantiation, we decided to export the OCEL snippets into the OCEL 2.0 format.

Heuristic Refiner

Afterwards, the preliminary OCEL snippets are passed to the heuristic refiner that concatenates them into a unified log. The unified log then undergoes a series of cleaning and refinement steps, leveraging heuristic rules and majority-based approaches, that are repeated until the log attains a final state with a maximum of five iterations. Within these iterations, the refiner alleviates data quality issues by, for example, resolving name inconsistencies, merging synonyms, and enforcing alignment between the object types, event types, objects, and events sections of the OCEL. For the initial example, a follow-up message could be "January 18, 2023: John completed the training". From this text entry, several data quality issues emerge at the collector level. First, "John" will be identified only partially for the object label, and second, the object type "Employee" will be missing as both values were not restated explicitly. However, as "John" matches the previously extracted label "John Doe", which was also already mapped to the

object type "Employee", the heuristic refiner can resolve both issues, resulting in the final OCEL:


```
{
 "objectTypes": [
  {"name": "Employee", "attributes": []}
 ],
 "eventTypes": [
  {"name": "attend training session", "attributes": []},
  {"name": "complete training", "attributes": []}
 ],
 "objects": [
  {"id": "John Doe", "type": "Employee"}
 ],
 "events": [
  {
   "id": "1",
   "type": "attend training session",
   "time": "2023-01-15T00:00:00Z",
   "relationships": [
    {"objectId": "John Doe", "qualifier": null}
   ]
  },
  {
   "id": "2",
   "type": "complete training",
   "time": "2023-01-18T00:00:00Z",
   "relationships": [
    {"objectId": "John Doe", "qualifier": null}
   ]
  }
 ]
}
```


4.3.3. Generative extractor

After the development of the heuristic extractor is finalized, we continue to develop the second extractor variant, known as the generative GEN-GEN extractor, which consists of a generative collector and a generative refiner. Unlike the heuristic extractor, which employs heuristic rules and majority-based approaches within a pipeline architecture, the generative extractor relies primarily on an LLM without specific fine-tuning for this particular task. Fig. 6 illustrates the functionality of the generative extractor and includes examples of the utilized prompts.

Generative Collector

The generative collector is implemented using OpenAI's `gpt-5-mini-2025-08-07` LLM with high reasoning effort provided through the Azure OpenAI API. Whereas the implementation in our earlier conference paper [20] utilized `gpt-4o-mini-2024-07-18` and the assistants API with file search capabilities, the current implementation solely relies on the chat completion API. For future instantiations, the specific LLM can be substituted as needed. After importing a textual description, the LLM is invoked with a system prompt containing instructions on the desired output and a user prompt containing the textual description. To ensure the output adheres to the OCEL 2.0 standard [17], we set the `response_format` parameter available through the Azure OpenAI API to `json_schema` and provide the OCEL 2.0 JavaScript Object Notation (JSON) schema[2] as the desired schema.

We formulated all prompts iteratively, using advanced prompting techniques [83]. We considered three prompt variants of increasing specificity, referred to as *lightweight*, *middleweight*, and *heavyweight*. The three prompt variants are provided in Appendix A.1. All prompt variants were tested with the same LLM and reasoning effort, and with strict JSON schema enforcement to ensure comparability. To quantify the impact of the prompt specificity, we evaluated the three prompt variants on the validation subset of the Level 1 CV log used during the extractor development, which consists of 100 textual descriptions from the order management event log. For each textual description, the collector produced a separate OCEL snippet. We then concatenated the 100 snippets and compared the resulting OCEL with the original OCEL using the comparison instance (see Section 4.3.1). We observed a monotonic improvement in F1-scores from lightweight (0.39) to middleweight (0.51) to heavyweight (0.58) prompting (see Table A.1), indicating that definitions, examples, and constraints from the OCEL 2.0 specification measurably improve extraction quality. Given this trend, we adopt the heavyweight prompt as the default configuration, as indicated in Fig. 6.

The heavyweight and final system prompt begins by assigning a role to the LLM, instructing it to act as a "process mining expert" that extracts OCELs from textual descriptions, followed by an empty OCEL. The task is further specified by explaining the OCEL components to extract, along with detailed descriptions taken from the OCEL 2.0 specification [17]. Furthermore, we explicitly instruct the LLM to solely use information from the provided textual descriptions and how to represent E2O and O2O relationships. Finally, we specify the OCEL 2.0 standard as the desired output format.

Following this approach, we iteratively provide textual descriptions to obtain a preliminary OCEL snippet for each textual description. The resulting collection of these snippets is then handed over to the generative refiner.

Generative Refiner

Subsequently, the preliminary OCEL snippets are combined into a concatenated OCEL based on predefined rules. These rules ensure that the concatenated event log adheres to the OCEL 2.0 standard.

The refinement is implemented as a sequence of LLM-guided normalization steps utilizing OpenAI's `gpt-5-mini-2025-08-07` with strict JSON output control and additional deterministic post-processing. In contrast to the implementation in our earlier conference paper [20], which just submitted the whole concatenated OCEL along with a respective refinement prompt to `gpt-4o-mini-2024-07-18`, the steps outlined in the following significantly increase the scalability of our approach to larger volumes of unstructured data for two reasons. First, the refiner modularizes LLM prompts by OCEL component, which makes the LLM's input length limit far harder to reach than with the entire OCEL. Second, responses are constrained by a JSON schema and applied through deterministic rewrite and validation rules, which improves internal consistency across OCEL components and conformance with the OCEL 2.0 JSON format. Several steps are executed in parallel to further reduce runtime on larger OCELs.

Analogous to the generative collector, we developed lightweight, middleweight, and heavyweight prompt variants for the generative refiner's LLM-guided normalization steps (see Appendix A.2 for the full prompts). To assess the quality of the prompt variants, we compared the refined OCELs with the original OCELs from the validation subset of the Level 1 CV log. While the lightweight prompt was not detailed enough to yield syntactically valid OCELs, the middleweight and heavyweight prompt variants produced valid OCELs with improved quality (see Table A.2 for details). Overall, in terms of F1-scores, the heavyweight prompt (0.76) performed slightly better than the middleweight prompt (0.72). We therefore use the heavyweight prompt variant for the generative refiner, as indicated in Fig. 6. While additional prompt tweaks may yield incremental improvements, the attainable gains are bounded by ambiguity and missing information in the textual descriptions, as well as by non-deterministic LLM outputs. Moreover, the heavyweight prompts already combine explicit task constraints with strict JSON schema enforcement.

[2] https://www.ocel-standard.org/2.0/ocel20-schema-json.json.

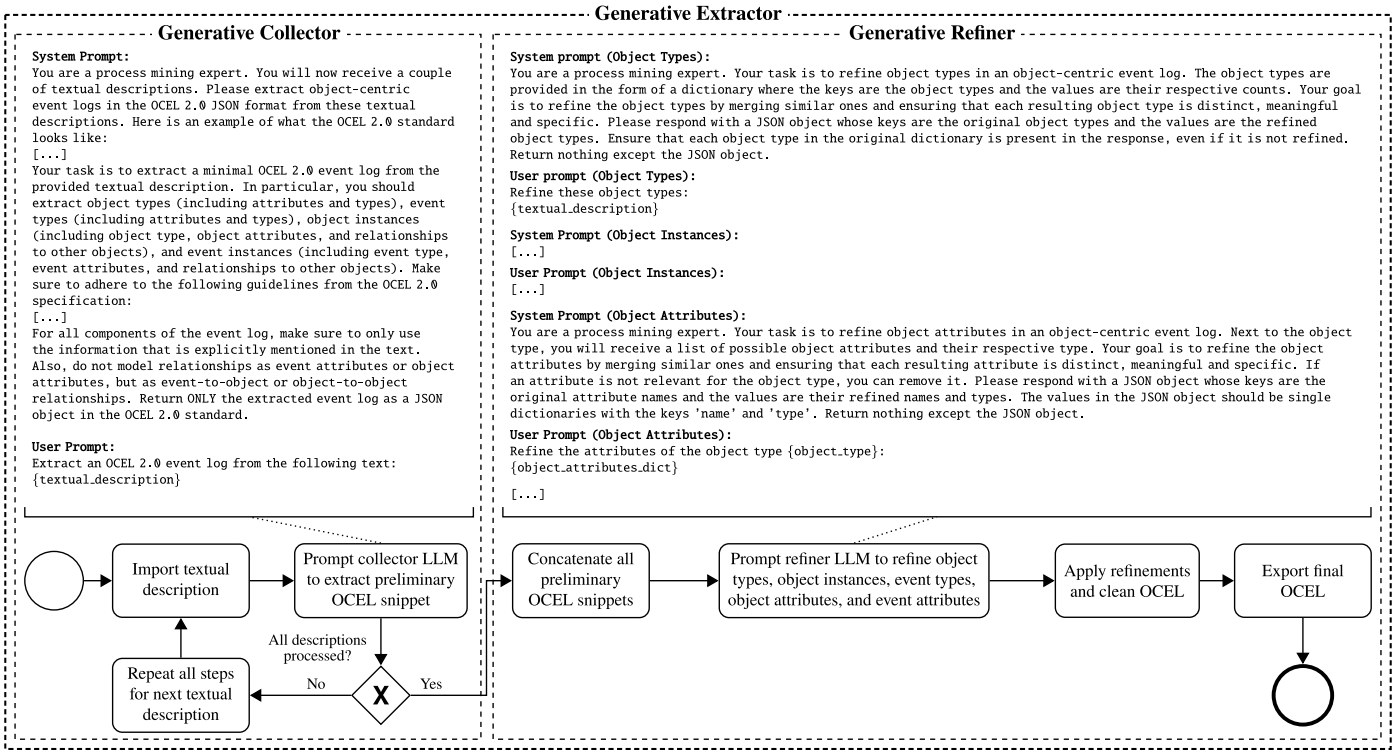


Fig. 6. Functionality of the generative GEN-GEN extractor.

To consolidate *object types*, the generative refiner first aggregates all object types and their frequencies in the concatenated OCEL, and prompts the LLM to merge similar object types such that each resulting type is distinct, meaningful, and specific. The returned mapping is applied to the list of available object types and the individual object instances. Object instances whose types cannot be mapped are removed.

For *object instances*, the generative refiner submits the list of object instance identifiers per refined object type and requests a one-to-one normalization that merges near-duplicates while preserving a distinct, meaningful, and specific identity for every remaining object instance. The resulting mappings are propagated to all existing object instances, E2O relationships, and O2O relationships.

Event types are harmonized analogously. The generative refiner aggregates event types with counts and asks the LLM to consolidate them into refined event types that usually start with a verb in the present tense and may include a related object type. The mapping is applied to the list of available event types and to all individual event instances.

After object types, object instances, and event types are normalized, the generative refiner harmonizes *attributes* for both object and event types. For each refined object and event type, the generative refiner supplies the set of observed attribute candidates (name and primitive type) and requests a canonical schema in which attributes are merged if they are synonymous and pruned if they are irrelevant, aiming to produce distinct, meaningful, and specific entries. The resulting mappings per type are then applied to all object and event instances: attribute names and types are rewritten, duplicates (with the same name and type) are removed, and attributes not present in the mapping are discarded.

When applying the refinements to events, timestamps are validated against ISO 8601. Events with invalid dates are skipped. The refiner further prevents duplicates by checking for identical event identifiers or collisions on the combination of timestamp and refined event type. Attributes are only appended to an existing object or event if an attribute with the same name and type is not already present, and only if the refined type matches.

Finally, O2O and E2O relationships are made consistent with the refined object and event instance space. For object instances, the refiner copies all O2O relationships, excluding self-relationships and duplicates, and only if both object instances still exist after refinement. For event instances, E2O relationships are retained under the same conditions. The refiner then prunes empty event and object types (no remaining instances) and removes attributes with no non-null values across any instance of their type. The outcome is a compact, semantically coherent OCEL with refined object types and event types (each with canonicalized attribute schemas), deduplicated object and event instances with normalized identifiers and attributes, and consistent O2O and E2O relationships.

4.3.4. Hybrid extractor

Lastly, we instantiate the hybrid extractor variants. The first variant, the GEN-HEU extractor, combines a generative collector with a heuristic refiner. In contrast, the second variant, the HEU-GEN extractor, integrates a heuristic collector with a generative refiner. These hybrid extractors are created by merging the collector and refiner subcomponents from the previously developed heuristic and generative extractor variants, eliminating the need for additional development logic.

5. Demonstration and evaluation

5.1. Evaluation strategy

As outlined in Section 3, we structure the evaluation of our OCEL extraction approach along the framework by [21]. EVAL1 strives to ensure "that a meaningful DSR problem is selected and formulated" Sonnenberg and vom Brocke [21, p. 394]. In Sections 1 and 2, we identified the lack of existing approaches to extract OCELs from textual descriptions, despite the potential of unstructured data for process mining and the paradigm shift to OCPM in the recent past. In EVAL2, we validate the design specification by demonstrating the novelty of our approach through a comparison with 18 related approaches resulting from a structured literature review. EVAL3 serves to demonstrate the

applicability of an instance of an artifact. Therefore, we quantitatively assess the extraction capabilities of four extractor variants on synthetic textual descriptions derived from six publicly available OCEL datasets, followed by a qualitative discussion based on seven criteria relevant in practical applications. To show the usefulness of our artifact in practice, we apply the two best-performing extractor variants on two types of naturally occurring textual descriptions, namely fire status updates and a legal judgment, and derive OCDFGs from the extracted OCELs in EVAL4.

5.2. EVAL2: Competing artifact analysis

To validate the design specification of our approach in EVAL2, we conduct a structured literature review and assess the consideration of our design objectives by related approaches. The literature review was conducted in June 2024 in preparation for our earlier conference paper [20] and covers the search engines Web of Science and Scopus, focusing on filtering titles, abstracts, and keywords using the following combinations of search terms:

- (“Process Mining” OR “BPM” OR “Business Process Management”) AND (“NLP" OR “Natural Language Processing” OR “Natural Language”)
- (“Process Mining” OR “BPM” OR “Business Process Management”) AND (“Extract*” OR “Generat*” OR “Construct*”) AND (“Unstructured*” OR “Text*”)
- “Object-Centric Process Mining” OR “Object-Centric Event Log*”

We also double-checked recent important conferences to ensure no relevant papers were overlooked. These conferences include CAiSE, BPM, ICPM, ICIS, ECIS, and ER. After extracting the papers, we screened their titles and abstracts to identify the relevant ones. If deemed important, we skimmed the full papers and analyzed their references, including additional relevant works. The search resulted in 73 papers focusing on event log extraction in a broader sense. Subsequently, we applied the following exclusion rules to refine the sample:

- Exclusion of papers published in a language other than English. (−1)
- Exclusion of papers with existing follow-up versions. (−5)
- Exclusion of artifacts targeting infrequently utilized data sources, such as UI logs, blockchain data, software execution traces, or visual information. (−11)
- Exclusion of artifacts that only translate event logs from one format to another. (−3)
- Exclusion of artifacts extracting only partial information, such as case IDs or activities, instead of complete event logs. (−29)
- Exclusion of artifacts extracting case-centric event logs from structured data sources, as they fail to fulfill DO1 and DO2. (−6)

Applying these rules narrowed the initial sample to 18 relevant competing artifacts. These artifacts were then analyzed to assess the extent to which they fulfill our DOs. The results of this analysis are summarized in Table 3.

From the competing artifact analysis, it becomes evident that most extraction endeavors target case-centric event log formats. Of the 18 artifacts, only three address the extraction of OCELs, which aligns with the requirements of DO1. However, these three artifacts fall short in meeting DO2, as they rely on structured datasets. For instance, Li et al. [84] propose the XOC log generator, which utilizes predefined causal relationships to map database table columns and values to events and objects in the XOC format. Similarly, Berti et al. [85] introduce a generic approach to extract OCED into the OCEL 1.0 format from SAP ERP databases, leveraging predefined relationships and blueprints. Furthermore, Swevels et al. [86] employ a reference ontology to extract OCED into event knowledge graphs compliant with the OCED metamodel. Despite fulfilling DO1, these approaches depend heavily on predefined relationships and ontologies, thereby limiting their degree of automation as requested by DO3.

In contrast, the remaining 15 artifacts explore extraction methods for unstructured textual data. These efforts aim to extract general intents from emails and online conversations to construct case-centric event logs. For example, van der Aalst and Nikolov [87] propose the EmailAnalyzer, which analyzes the subject and recipients of an email using a combination of manual processes and predefined rules to partially automate the extraction of event logs into the MXML format. Similarly, Banziger et al. [11] analyze CRM data from emails, appointments, and tasks to extract XES logs. To identify activities within a document, they employ an LDA algorithm, thus reducing the need for human intervention. Furthermore, Laga et al. [88], Chambers et al. [89], Bicknell and Krebs [90], and Elleuch et al. [91] address the same problem of extracting XES logs from emails utilizing supervised classification models, unsupervised bag-of-words models, a combination of various NLP techniques, and grouping and pattern recognition methods, respectively.

Supervised NLP techniques, such as utterance classification and NLI algorithms, were also used by Epure et al. [92], Holstrup et al. [93], and Kecht et al. [3] to identify activities from online conversations. Similarly, Holz et al. [94] applies supervised LDA algorithms to derive XES logs from personal service documents, while Geeganage et al. [12] and Nai et al. [95] employ unsupervised NLP techniques like REGEX, PoS tagging, and NER on textual descriptions and tender documents to enrich existing XES logs. In contrast, ontology-based approaches were used by Pospiech et al. [96], Zayakin et al. [97], and Dwyer et al. [98] to extract event logs from diverse sources such as heterogeneous documents, internet data, and textual code lists.

In conclusion, the competing artifact analysis reveals that none of the reviewed approaches simultaneously fulfill both DO1 and DO2. While a few artifacts address OCEL extraction, they rely on structured datasets. On the other hand, case-centric extraction efforts focus on unstructured textual data but do not meet the requirements of DO1. Regarding DO3, heuristic rules and unsupervised NLP approaches have proven effective in minimizing human intervention and handling diverse data sources. Therefore, we aim to specifically leverage these algorithms in our approach, striving to ultimately fulfill all DOs comprehensively.

5.3. EVAL3: Artificial assessment of extraction capabilities

In EVAL3, we comprehensively assess the extraction capabilities of the four instantiated extractor variants, combining quantitative and qualitative evaluation in an artificial context. This step aligns with the DSR methodology by simultaneously incorporating both the demonstration and evaluation phases, where we demonstrate the approach and assess its extraction performance. By leveraging our development framework, we create synthetic textual descriptions for six publicly available OCELs, provide these descriptions to the four extractor variants, and compare the reconstructed logs with their original counterparts. This way, we can assess the effectiveness of the different variants performing the extraction task. Afterwards, we qualitatively discuss the applicability of the four extractor variants across seven criteria: extraction quality, process inference, generalization capabilities, semantic utility, processing speed, controllability, and affordability. This artificial evaluation approach has been chosen due to the absence of real, publicly available textual descriptions suitable for benchmarking the effectiveness of extraction endeavors.

Data preparation: Our test data set consists of six publicly available event logs in the OCEL 2.0 format. Three of these logs – the recruitment log [79], logistics log [80], and the P2P log [81] – were previously employed in the development and validation of the heuristic extractor. The remaining three logs – an order management log [82], a production

Table 3
Competing artifact analysis.

Competing artifact	DO1 (Event log extraction in an object-centric format)	DO2 (Event log extraction from unstructured textual descriptions)	DO3 (Event log extraction at scale, fully automated, and without human intervention)
van der Aalst and Nikolov [87]	○ Extraction into case-centric MXML format	◑ Extraction from semi-structured emails using timestamps, senders, recipients, and subject headers	◔ Human must filter down emails manually, must partially tag messages, and must define rules for a translation algorithm
Pospiech et al. [96]	○ Extraction into XML-file that follows similar structures as a case-centric event log	● Extraction from heterogeneous data such as structured databases, semi-structured emails, and fully unstructured Word documents	◑ Ontology-based approach that requires domain knowledge upfront to define the ontologies
Banziger et al. [11]	○ Extraction into case-centric XES format	◑ Extraction from semi-structured Emails, Appointments, Phone Calls, and Tasks	● Fully automated extraction using unsupervised LDA algorithm
Epure et al. [92]	○ Extraction into informal representation of a case-centric event log	◑ Extraction from semi-structured Reddit conversations	◕ Usage of a supervised classification algorithm to classify activities that requires manual pre-labeling of a training data set
Li et al. [84]	● Extraction into object-centric XOC format	○ Extraction from structured databases	◑ Domain knowledge necessary upfront to define causal relationships between database columns and XOC categories
Laga et al. [88]	○ Extraction into case-centric XES format	◑ Extraction from semi-structured emails using timestamps, senders, recipients, and subject headers	◕ Usage of a supervised classification algorithm to classify activities that requires manual pre-labeling of a training data set
Chambers et al. [89]	○ Extraction into case-centric XES format	◑ Extraction from semi-structured emails using timestamps, senders, recipients, and subject headers	◕ Usage of a supervised classification algorithm to classify activities that requires manual pre-labeling of a training data set
Holstrup et al. [93]	○ Extraction into case-centric XES format	◑ Extraction from semi-structured online conversations using timestamps, senders, and content	◑ Utterance classification based on manually predefined taxonomy
Bicknell and Krebs [90]	○ Extraction into case-centric XES format	◑ Extraction from semi-structured emails using timestamps, senders, recipients, and subject headers	● Fully automated extraction using a bag-of-words model
Holz et al. [94]	○ Extraction into case-centric XES format	● Extraction from unstructured textual documents	◕ Usage of a supervised classification algorithm to classify case IDs, activities, and timestamps that requires manual pre-labeling of a training data set
Kecht et al. [3]	○ Extraction into case-centric XES format	◑ Extraction from semi-structured Twitter conversations using timestamps and senders	◕ Usage of a supervised NLI algorithm that requires manual definition of domain-specific topics and pre-labeling of a tiny data sample
Geeganage et al. [12]	○ Extraction into case-centric XES format	◕ Extraction from unstructured textual columns within a structured event log	● Fully automated event log enhancement using different NLP techniques such as NER, dependency parsing, PoS tagging, and semantic similarity validation
Zayakin et al. [97]	○ Extraction into informal representation of a case-centric event series	● Extraction from unstructured heterogeneous internet sources	◑ Ontology-based approach that requires domain knowledge upfront to define the ontologies
Berti et al. [85]	● Extraction into object-centric OCEL 1.0 format	○ Extraction from structured SAP ERP databases	◑ Domain knowledge necessary upfront to define graph of relationships and blueprints that map relational structures to data objects
Elleuch et al. [91]	○ Extraction into case-centric event log that is similar to the XES format	◑ Extraction from semi-structured emails using timestamps and conversational history	● Fully automated extraction by mapping text to local features and extracting patterns between features
Nai et al. [95]	○ Extraction into case-centric XES format	◑ Extraction from semi-structured tender documents using document number and timestamps	● Fully automated event log enhancement using NER, REGEX, and text matching
Dwyer et al. [98]	○ Extraction into informal representation of a case-centric event log	◑ Extraction from semi-structured textual healthcare code list	◑ Ontology-based approach that requires domain knowledge upfront to define the ontologies
Swevels et al. [86]	● Extraction into event knowledge graphs compliant with object-centric OCED metamodel	○ Extraction from legacy data in structured databases	◑ Ontology-based approach that requires domain knowledge upfront to define the ontologies

log [99], and an Age of Empires log [100] – were not used during development, providing an opportunity to evaluate the generalization capabilities of each extractor variant. A standardized test subset of 1000 events is created for each event log, ensuring no overlap between the test subsets and the development and validation subsets of the heuristic extractor.

Generation of textual descriptions: The test subsets are then processed by the generator instance of the development framework, tasked with converting the events into textual descriptions across three levels of complexity. To this end, each subset is divided into three equal parts. One-third of the events are transformed into Complexity Level 1 descriptions — one textual description per event. Another third is converted into Complexity Level 2 descriptions — non-overlapping daily reports, with events grouped by day. The final third is transformed into Complexity Level 3 descriptions — overlapping reports, where events are grouped based on their related objects. To ensure the comparability of the results with our prior conference paper [20], we use the same textual descriptions (generated with OpenAI's `gpt-4o-mini-2024-07-18` LLM) in this article.

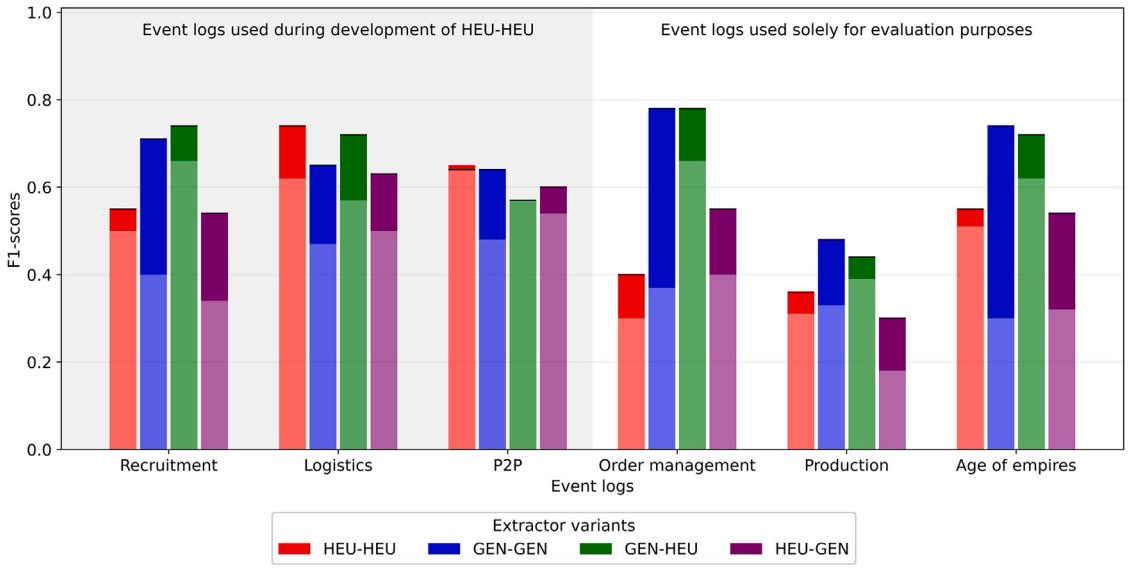


Fig. 7. F1-score by event log and extractor variant.
Saturated overlays indicate the change in performance compared to our previous conference paper [20]

OCEL extraction: The synthetic textual descriptions are then provided to the four extractor variants. These variants leverage their respective heuristic and generative collector and refiner subcomponents to reconstruct the original logs. While the generative subcomponents in the conference paper [20] used OpenAI's `gpt-4o-mini-2024-07-18` LLM, the upgraded versions now leverage the `gpt-5-mini-2025-08-07` LLM. Both setups result in one extracted log in OCEL 2.0 format for each original log.

Comparison: Lastly, the extracted OCELs are compared with their original counterparts using the comparison instance of the development framework. For this purpose, the comparison instance calculates the alignment between the extracted and original OCELs across various categories and levels of detail, using precision, recall, and F1-score. Additionally, the comparison instance assesses whether the extracted OCELs preserve the original process behavior by calculating the DFG similarity for each object type in the original OCEL. These measures indicate the effectiveness of the distinct extractor variants in performing the extraction task, which consists of collection and refinement.

5.3.1. Quantitative results

Fig. 7 shows the F1-score per event log and for each of the four extractor variants, including the change in performance compared to our previous conference paper [20]. In four event logs (recruitment, order management, production, and Age of Empires), the GEN-GEN and GEN-HEU extractor variants achieve comparable performance and significantly outperform the other two variants. In contrast, all four variants achieve comparable performance on the logistics and the P2P event log.

Compared to our previous conference paper [20], we observe a significant increase in performance of the GEN-GEN extractor, achieving an improvement of 0.28 in F1-score across the six event logs. We attribute this improvement primarily to the revised implementation of both generative components (as pointed out in Section 4.3.3) and to updating the underlying LLM from `gpt-4o-mini-2024-07-18` to `gpt-5-mini-2025-08-07`. As the performance increases of the GEN-HEU and HEU-GEN variants show (0.08 and 0.15, respectively), the changes in both generative components contributed to the overall improvement. Although the implementation of the heuristic components did not change, we observe an overall performance increase of 0.06 (despite a 0.01 decrease on the P2P event log) in the respective F1-score. We trace this observation back to changing the used SpaCy language model from small (`en_core_web_sm`) to medium (`en_core_web_md`), and to a minor fix in the comparison instance to improve the matching of original and retrieved components. Nevertheless, the different performance of the HEU-HEU extractor variant on the three event logs used during its development, compared to its performance on the three event logs used for evaluation purposes, indicates that the heuristic extractor was fine-tuned to the characteristics of the former event logs.

Table 4 reports the average precision and recall values of the six event logs for each extractor variant, along with the resulting F1-score (harmonic mean of precision and recall) for each level of detail reported through the comparison instance. The average overall precision and recall values in the last row are calculated based on parent levels (bold rows) and absolute child levels, since the absolute level evaluates the correctness on the entire event log and not only if the parent category is correctly identified, and therefore, provides the tougher benchmark.

Analyzing these results shows that the GEN-GEN, GEN-HEU, and HEU-GEN variants achieve higher recall (ranging from 0.65 to 0.76) than precision values (0.44 to 0.61), resulting in harmonized F1-scores between 0.53 and 0.67, whereas the HEU-HEU extractor achieves similar precision (0.54) and recall (0.55) values. Hence, the former three extractor variants capture most of the important information but also integrate irrelevant noise to some extent. Reassessing the roles of the collector and refiner subcomponents, the goal of the collector is to collect extensive information from the textual descriptions, ensuring a high recall, while the refiner should clean and align the previously collected information to improve precision. Given the results, the generative collector consistently outperforms the heuristic collector, whereas the generative and the heuristic refiner yield comparable results for a fixed collector instance.

Furthermore, the gap between recall and precision is especially evident in the object types and event types categories, where high recall values are counterbalanced by lower precision, resulting in moderate F1-scores. This imbalance arises because the number of object types and event types in the original OCEL is very low, and already a few newly introduced types significantly impair the precision. In identifying object instances and matching object instances to object types, all extractor variants perform comparably well, contributing positively to both recall and precision. This observation suggests that identifying and correctly associating object instances with their respective object types is a relatively straightforward task for the distinct extractor variants. In contrast, extracting and correctly mapping event attributes proves to be particularly challenging. Distinguishing between object attributes

Table 4
Precision (average), recall (average), and F1-score (harmonic mean of precision and recall) of the four extractor variants across the six OCELs. Average precision and recall values are calculated based on parent levels (bold rows) and absolute child levels.

	HEU-HEU			GEN-GEN			GEN-HEU			HEU-GEN		
	Pre.	**Rec.**	**F1**	**Pre.**	**Rec.**	**F1**	**Pre.**	**Rec.**	**F1**	**Pre.**	**Rec.**	**F1**
Object type level	0.66	0.85	0.74	0.53	0.97	0.68	0.58	0.96	0.72	0.27	0.93	0.41
Object instance level	0.97	0.91	0.94	0.92	0.96	0.94	0.89	0.99	0.94	0.84	0.94	0.89
Object-to-type (Abs)	0.79	0.73	0.76	0.90	0.94	0.92	0.86	0.95	0.90	0.52	0.56	0.54
Object-to-type (Rel)	0.82	0.81	0.82	0.98	0.98	0.98	0.96	0.96	0.96	0.61	0.61	0.61
Object-to-attribute-type (Abs)	0.37	0.44	0.40	0.48	0.58	0.52	0.64	0.62	0.63	0.33	0.55	0.41
Object-to-attribute-type (Rel)	0.40	0.59	0.48	0.54	0.64	0.58	0.75	0.62	0.68	0.40	0.71	0.52
Object-to-attribute-value (Abs)	0.37	0.47	0.41	0.49	0.58	0.53	0.58	0.60	0.59	0.29	0.48	0.36
Object-to-attribute-value (Rel)	0.39	0.63	0.48	0.55	0.63	0.59	0.68	0.60	0.64	0.35	0.64	0.45
Object-to-Object (Abs)	0.67	0.44	0.53	0.39	0.62	0.48	0.54	0.72	0.62	0.53	0.46	0.49
Object-to-Object (Rel)	0.74	0.52	0.61	0.46	0.66	0.55	0.62	0.73	0.67	0.71	0.52	0.60
Event type level	0.34	0.75	0.46	0.44	0.99	0.61	0.40	0.83	0.54	0.22	0.93	0.36
Event instance level	0.51	0.48	0.50	0.76	0.82	0.79	0.67	0.69	0.68	0.55	0.74	0.63
Event-to-attribute-type (Abs)	0.38	0.33	0.35	0.47	0.61	0.53	0.46	0.53	0.49	0.40	0.53	0.45
Event-to-attribute-type (Rel)	0.69	0.72	0.70	0.60	0.74	0.66	0.70	0.81	0.75	0.66	0.72	0.69
Event-to-attribute-value (Abs)	0.38	0.33	0.35	0.55	0.65	0.60	0.49	0.54	0.51	0.43	0.56	0.49
Event-to-attribute-value (Rel)	0.68	0.72	0.70	0.72	0.82	0.77	0.76	0.84	0.80	0.71	0.75	0.73
Event-to-object (Abs)	0.48	0.33	0.39	0.61	0.67	0.64	0.59	0.62	0.60	0.51	0.43	0.47
Event-to-object (Rel)	0.91	0.65	0.76	0.82	0.83	0.82	0.89	0.88	0.88	0.92	0.61	0.73
Overall	**0.54**	**0.55**	**0.54**	**0.60**	**0.76**	**0.67**	**0.61**	**0.73**	**0.66**	**0.44**	**0.65**	**0.53**

and event attributes, or even deciding whether an attribute is worth capturing at all, is often difficult. In practice, structuring elements – especially event attributes – within an OCEL is a design choice that varies significantly among individuals. As a result, the same information can be represented by different data models since no ground truth for modeling OCEL components exists (as implied in this evaluation step). Consequently, some decisions made by our extractor variants may still be legitimate and valuable, even though the original OCELs counterparts follow a different representation.

In conclusion, both the GEN-GEN and the GEN-HEU extractor variants outperform the other variants, particularly on event logs not used during the development of the heuristic components. Whereas our previous conference paper [20] recommended the GEN-HEU extractor as the most suitable variant, the revision of the generative components not only increased the performance of this variant, but also enabled the GEN-GEN extractor variant to achieve an equivalent performance.

Fig. 8 complements the log-level comparison by reporting the weighted combined DFG similarity between the original and extracted OCELs. Across the three event logs used for evaluation purposes, the GEN-GEN variant achieves the highest average weighted combined DFG similarity (F1-score: 0.50), followed by GEN-HEU (0.40), HEU-GEN (0.38), and HEU-HEU (0.17). However, we observe that the weighted combined DFG similarity in Fig. 8 varies substantially more across the datasets and extractor variants than the log-level F1-scores visualized in Fig. 7. The HEU-HEU extractor variant consistently underperforms on the three event logs not used during its development, thus confirming the aforementioned observation that its rules were fine-tuned to the characteristics of the event logs used during its development.

In contrast, extractor variants with generative components preserve not only OCEL components but also a considerable portion of the directly-follows relations that characterize process behavior per object type. On average, the GEN-GEN extractor variant achieves the highest weighted combined DFG similarity, but the absolute gains depend on the distribution of object types in each dataset. In the order management log, for example, the object type `items` accounts for more than 90% of all object instances and achieves a combined DFG similarity of 0.64. Because the weighted metric emphasizes frequent object types, the weighted similarity of 0.63 largely reflects the performance on `items`. Similarly, the production log includes large object types like `FormedPart` and `MalePart`, with combined DFG similarities of 0.33 and 0.25, respectively, thereby diluting the effect of better-performing event types such as `SteelSheet` (0.71). Consequently, the process-level performance depends on accurately recognizing the most frequent object types and their instances, the associated event types and their instances, and the respective E2O relationships.

5.3.2. Qualitative discussion

Building upon the insights of the artificial evaluation, we conduct a qualitative discussion of the applicability of the four extractor variants, examining seven criteria: extraction quality, process inference, generalization capabilities, semantic utility, processing speed, controllability, and affordability. The findings of this evaluation are summarized in Table 5.

Extraction quality: Given the results at the log level (Fig. 7 and Table 4), the GEN-GEN extractor exhibits the highest extraction quality with an average overall F1-score of 0.67, followed by the GEN-HEU extractor with an F1-score of 0.66. In contrast, the HEU-HEU and HEU-GEN extractors underperform with respective F1-scores of 0.54 and 0.53. Therefore, the extraction quality primarily depends on the implementation of the collector subcomponent. The heuristic collector analyzes the syntactic structure of a sentence based on predefined rules to extract extensive information from the provided textual descriptions. While effective, this approach introduces three potential sources of error. First, the heuristic collector, lacking semantic understanding, can only extract entities explicitly stated in the text. This characteristic places high demands on the precision and completeness of the textual descriptions, which may not always be met in real-world scenarios. Second, although the SpaCy NLP parsing pipeline generally provides accurate NLP outputs – such as correct tokens, dependencies, PoS tags, and NER labels – occasional errors still occur. These errors can lead to missing or incorrectly categorized information, resulting in incomplete or erroneous preliminary OCEL snippets, despite correct predefined rules. Third, the predefined rules themselves are a potential source of error. Being manually defined, these rules may overlook edge cases or overfit to a specific tone, leading to inaccuracies. In contrast, the generative collector integrates both syntactic and semantic understanding to extract information from textual descriptions, addressing many limitations of the heuristic counterpart, such as overfitting and the need for precise textual descriptions.

Process inference: The process-level comparison evaluates whether the extracted OCEL induces a similar directly-follows structure as the original OCEL and is particularly sensitive to consistent event types across textual descriptions and complete E2O relationships, because both factors determine the specific activity sequences after flattening per each object type. Accordingly, the largest gains in Fig. 8 stem

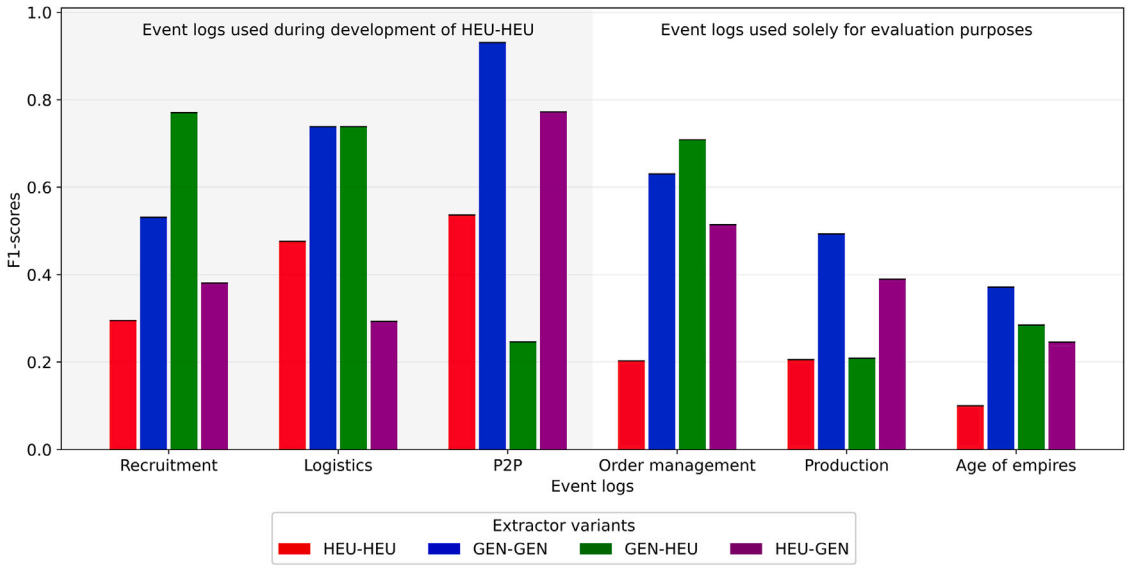


Fig. 8. Weighted combined DFG similarity by event log and extractor variant.
DFG similarity is computed per object type (after flattening), comparing node and edge sets via F1-score and aggregating using the number of object instances in the original OCEL.

Table 5
Comparison of performance and utility across the four extractor variants.

Criteria	HEU-HEU	GEN-GEN	GEN-HEU	HEU-GEN
Extraction quality	◔ F1-score = 0.54	◕ F1-score = 0.67	◕ F1-score = 0.66	◔ F1-score = 0.53
Process inference	○ Weighted combined DFG similarity = 0.17	◑ Weighted combined DFG similarity = 0.50	◔ Weighted combined DFG similarity = 0.40	◔ Weighted combined DFG similarity = 0.38
Generalization capabilities	○ Overfitting risk of heuristic rules	● Low overfitting risk of LLM prompts	◑ Overfitting risk of heuristic rules	◑ Overfitting risk of heuristic rules
Semantic utility	○ Inconsistent and implausible outputs	● Coherent and standardized outputs	◑ Formatting issues and unnormalized phrases	◑ Implausible outputs
Processing speed	● No external API restrictions	◔ Limited by LLM API endpoint	◑ Limited by LLM API endpoint	◑ Limited by LLM API endpoint
Controllability	● Full controllability due to pre-defined rules	◔ Controllability only via prompts	◑ Partial controllability via prompts and pre-defined rules	◑ Partial controllability via prompts and pre-defined rules
Affordability	◕ Only computational infrastructure	◔ LLM API cost and computational infrastructure	◑ LLM API cost and computational infrastructure	◑ LLM API cost and computational infrastructure

from object types with many instances and richer behavior, where fragmented labeling or brittle object linking quickly removes or distorts directly-follows edges. These patterns suggest that generative components improve process inference primarily by reducing semantic omissions in the collector and harmonizing near-duplicate labels and identifiers in the refiner, thereby stabilizing trace construction and preventing fragmentation into spurious variants.

Generalization capabilities: Fig. 7 indicates that the generative collector used in the GEN-GEN and GEN-HEU extractors exhibits superior generalization capabilities compared to the heuristic collector employed in the HEU-HEU and HEU-GEN extractors. This observation can be traced back to the generative collector's ability to operate on broad instructions, leveraging both syntactic and semantic understanding. Conversely, the heuristic collector, constrained by manually defined rules, relies solely on syntactic analysis, limiting its generalization capacity. The manual definition of these rules thereby significantly increases the risk of overfitting.

Semantic utility: Semantic utility refers to the usefulness and logical coherence of the extracted OCELs from a human perspective. Specifically, the heuristic collector, which relies exclusively on predefined rules to analyze the syntactic structure of the textual descriptions, often yields counterintuitive results. The static nature of these rules can lead to semantically incorrect outputs. For example, a recruiter's name, such as "Dionne Kershaw", might be misclassified as an object or attribute type, or nonsensical event types like "occur again" may be generated. Sometimes, the heuristic collector even produces syntactically incorrect outputs, such as an event type labeled "assign she to the application". These issues significantly impede the output quality of the heuristic refiner, as its rule-based implementation fails to repair implausible modeling, such as creating an attribute for each individual applicant for the object type application. While the generative refiner can resolve these issues to some extent, it nevertheless introduces implausible object types, such as "Relation", "Temporal", and "Activity". In contrast, the generative collector, leveraging an LLM trained on extensive datasets, consistently generates grammatically accurate and semantically reasonable outputs. Despite only minor differences in the F1-score, the GEN-GEN and GEN-HEU extractors significantly differ in the formatting and interpretability of the outputs. We further elaborate on these differences in Section 5.4.

Processing speed: With the current implementation of our evaluated heuristic and generative subcomponents, the main performance bottleneck with large datasets is the calling of an external LLM API endpoint with token and request limits, such as the services provided by Azure OpenAI. We conducted the artificial evaluation on an online Jupyter Notebook instance featuring 32 cores, 3 NVIDIA A100 GPUs, and 1.4 TB of RAM, and measured execution times ranging from 10 min to 1 h per dataset with the HEU-HEU extractor. However, despite numerous LLM API invocations, the HEU-GEN extractor achieved the

fastest processing speed, completing tests within 7 to 13 min per dataset. In turn, the GEN-GEN and GEN-HEU extractors exhibit the slowest processing speed, with execution times ranging from 1.5 to 3 h per test dataset. However, a locally deployed LLM can significantly increase the performance of the generative subcomponents.

Controllability: The HEU-HEU extractor, relying solely on manually predefined rules, offers the highest level of controllability. In contrast, the GEN-GEN extractor, which utilizes LLMs, often regarded as "black boxes", demonstrates the lowest level of controllability. Similarly, the generative components of the GEN-HEU and HEU-GEN extractors face comparable controllability limitations. Currently, the controllability of these generative components is limited to the proper definition of system and user prompts. However, anticipated advancements in LLM technology will likely address these constraints and improve controllability in the future.

Affordability: In our instantiation, the Azure OpenAI API was utilized as the foundation for the generative collector and refiner. Both generative components can be significant cost drivers, as the generative collector executes one API call per provided textual description, and the generative refiner follows a sequence of LLM-guided normalization steps, which include one API call per identified object and event type. Consequently, all extractor variants with generative subcomponents can produce substantial API costs, thus impeding affordability. While the heuristic components in the HEU-HEU and HEU-GEN extractors do not rely on external APIs, they still demand considerable computational resources, leading to execution costs. These costs, however, remain significantly lower compared to the potential expenses associated with LLM APIs. In theory, the API costs could be entirely avoided by employing publicly available LLMs, for example, a locally deployed open-weight LLM. Nevertheless, depending on the utilized LLM, this would likely come at the expense of extraction quality.

In summary, in line with the quantitative results, we continue to recommend the GEN-GEN extractor variant, as it exhibits the highest levels of extraction quality, process inference, generalization capabilities, and semantic utility. The greater the influence of processing speed, controllability, and affordability on the decision to choose an extractor variant, the more viable the GEN-HEU variant remains, despite its reduced semantic utility. While the HEU-HEU extractor variant excels in the latter three criteria, its low generalization capabilities and semantic utility will significantly lower the basis for further analysis and decision-making based on the extracted OCELs. These limitations also extend to the HEU-GEN extractor variant, as the generative refiner cannot fully resolve the data quality issues induced by the heuristic collector.

5.4. EVAL4: Assessment of the real-world applicability of the GEN-GEN and GEN-HEU extractor variants

Building on the results from EVAL3, where the GEN-GEN and the GEN-HEU extractor variants outperformed the other two extractor variants in controlled, synthetic environments, we now aim to assess the performance of the GEN-GEN and the GEN-HEU extractor variants on real-world textual descriptions in EVAL4. To the best of our knowledge, there are currently no publicly available process mining benchmark datasets that jointly provide real-world textual descriptions, a corresponding OCEL, and an external business process specification that would enable a fully measurable evaluation of process inference from text.

We select two distinct types of real-world textual descriptions for EVAL4. The first dataset consists of fire status updates regularly posted by the California Department of Forestry and Fire Protection,[3] which provide real-time information about ongoing wildfire events. The second dataset comprises a legal judgment from the European Court of Human Rights,[4] offering a complex and varied linguistic structure. This evaluation activity examines how well the GEN-GEN and GEN-HEU extractor variants generalize beyond synthetic datasets and perform on naturally occurring textual descriptions.

Fire Status Updates: Initially, we utilize web scraping to gather status updates for a total of 10 fire incidents from the official website of the California Department of Forestry and Fire Protection. Each status update is transformed into a separate textual description, capturing essential details such as the specific fire involved, the date and time the update was reported, and the situation summary provided in the original status update. As a result, we collected 280 status updates, distributed across the 10 fire incidents. An example fire status update appears as follows:

> ***Lilac Fire as of 01/21/2025 at 10:56 AM:***
>
> *The fire is located near Old Highway 395 and Lilac Road. The forward rate of spread has been stopped, and the fire is now 30% contained. Fire crew will continue to build containment lines and extinguish hotspots within the fire's perimeter.*

Such fire status updates are interesting for process mining as they provide a detailed, time-sequenced record of evolving wildfire events. The updates include crucial information such as containment progress, resource allocation, and incident escalation, which allows for identifying patterns in how fire management tasks are performed and how they evolve in response to changing conditions.

The textual descriptions are then processed in isolation by both the GEN-GEN and the GEN-HEU extractor variants, which are tasked with creating an OCEL based on the provided fire status updates. The GEN-GEN extractor variant identifies 306 object instances across 27 object types, with `OperationalUnit` (52), `GeographicArea` (38), and `Organization` (29) representing the most frequent object types. Furthermore, it derives 628 event instances belonging to 199 event types, including `Publish status update` (40), `Publish situation report` (40), and `Establish unified command` (19) as the most frequent event types.

In turn, the GEN-HEU extractor variant identifies 381 object instances across 52 object types and 547 event instances across 103 event types. The most frequent object types include `location` (59), `fire` (52), and `organization` (26), whereas `fire reported` (80), `situation report` (64), and `status update` (57) represent the most frequent event types. Both extractor variants identify a similar number of E2O relationships (GEN-GEN: 2185, GEN-HEU: 1843).

Using the `discover_ocdfg` function provided by the Python library PM4Py [101], we derive OCDFGs to visualize the efforts taken for the different fire incidents. Figs. 9 and 10, for instance, showcase the OCDFG with frequency annotations for a specific fire instance, namely the Lilac Fire that started on January 21, 2025, in San Diego. Both figures were created with an activity and edge threshold of 2, thus containing events and transitions for the same object type occurring at least twice in the corresponding OCEL.

Fig. 9 (based on the GEN-GEN extractor variant) presents a compact control flow with four event types, three of them related to providing an update on the Lilac Fire and the fourth indicating progress in the containment efforts. The high number of events per event type indicates that multiple updates are published for a single fire, with the update frequently containing the structure of the fire, the involved firefighter crew, and geographical information, such as the location of the fire. Consequently, the textual information of the exemplary fire status update at the beginning of this section is comprehensively represented in the resulting OCDFG.

In contrast, Fig. 10 (based on the GEN-HEU extractor variant) explicitly contains an event type modeling the planning of containment

[3] https://www.fire.ca.gov/incidents/.

[4] https://hudoc.echr.coe.int/?i=001-242203.

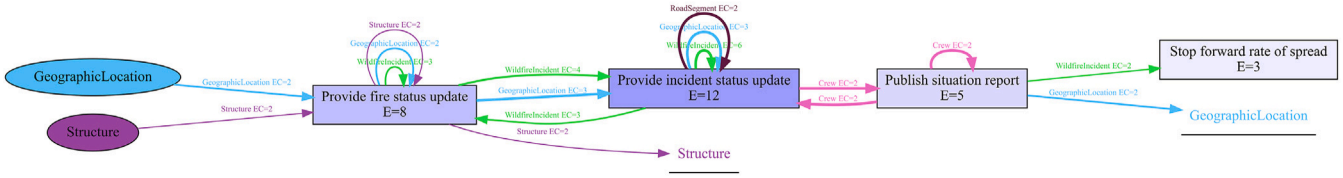


Fig. 9. OCDFG based on an OCEL derived with the GEN-GEN extractor variant for fire status updates regarding the Lilac Fire in January 2025 in San Diego.

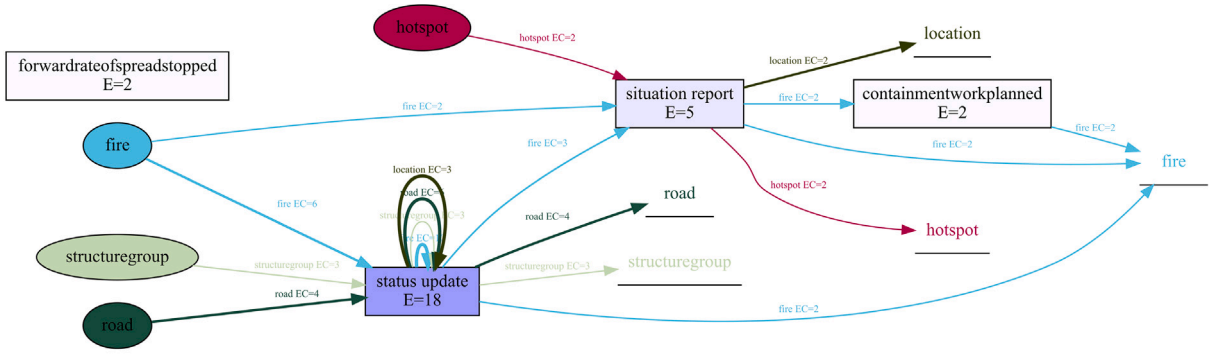


Fig. 10. OCDFG based on an OCEL derived with the GEN-HEU extractor variant for fire status updates regarding the Lilac Fire in January 2025 in San Diego.

work and exposes a slightly richer set of object types. However, the event type and object type vocabularies are less normalized (e. g., concatenated labels such as `forwardrateofspreadstopped`), thus impeding the readability and interpretability in practice.

Legal Judgment: Next, we retrieve a legal judgment from the HUDOC database of the European Court of Human Rights, specifically the case of F.S.M. v. Spain. This case concerns the criminal conviction of an elderly individual with cognitive impairments for tax-related offenses, raising important questions about the right to a fair trial and the adequacy of procedural safeguards. The judgment navigates complex legal arguments, procedural safeguards, and balances individual rights with justice requirements. Legal judgments like this one are well-suited for process mining, as they follow a structured yet intricate sequence of legal reasoning, procedural developments, and factual assessments. Extracting events from such documents enables a deeper understanding of legal workflows, decision-making patterns, and the interplay between domestic and international legal standards.

Given that the judgment spans over 30 pages, we segment it into smaller, more manageable subsets at logical breakpoints, such as the end of a paragraph. This process results in 205 textual descriptions, preserving the content of the original judgment, which are then processed in isolation by both the GEN-GEN and GEN-HEU extractor variants to generate a corresponding OCEL.

The GEN-GEN extractor variant identifies 383 object instances across 37 object types, with `LegalNorm` (56), `Case` (39), and `Person` (39) representing the most frequent object types. Furthermore, it derives 342 event instances belonging to 246 event types, including `Issue judgment` (13), `Issue decision` (11), and `Communicate` (8) as the most frequent event types.

In turn, the GEN-HEU extractor variant identifies 376 object instances across 101 object types and 294 event instances across 193 event types. The most frequent object types include `person` (32), `case` (31), and `forensicreport` (21), whereas `decisionissued` (7), `supremecourtdecision` (6), and `line` (6) represent the most frequent event types. Both extractor variants identify a comparable density of E2O relationships (GEN-GEN: 1067, GEN-HEU: 902).

Again, we use PM4Py's `discover_ocdfg` function to visualize the event and object types involved in the legal judgment of F.S.M. v. Spain. Figs. 11 and 12, both created with an exemplary activity threshold of 5 and edge threshold of 2, show the resulting OCDFGs for the GEN-GEN and the GEN-HEU extractor variant, respectively.

Fig. 11 (based on the GEN-GEN extractor variant) shows a compact backbone of adjudicative steps. The `Case` object instance acts as the main hub linking central decision-making activities based on legal norms and state laws. Apart from the isolated `Record line` event instances, the generated OCDFG provides a coherent view of the legal trajectory.

In contrast, Fig. 12 (based on the GEN-HEU extractor variant) surfaces a more granular but fragmented picture due to five separate connected components. Besides core outcomes, such as `decisionissued` and `supremecourtdecisision`, it exposes several procedural details, such as `examined` and `assessment`. The `case` object instance again anchors most event instances, but additional object types (e. g., `forensicreport` and `request`) participate in the process. Again, some labels, for example, `accommodationsandadjustments_made_by_court` and `psychiatric report` reflect formatting issues and unnormalized phrases, indicating a noisier vocabulary of the GEN-HEU extractor variant.

Taken together, the results on naturally occurring textual descriptions show that both the GEN-GEN and GEN-HEU extractor variants generalize beyond the synthetic datasets in EVAL3. Across the fire status updated and legal judgment corpora, the extracted OCELs are densely grounded in object and event instances, and yield OCDFGs that capture the central storyline of each domain. Nevertheless, this evaluation highlights existing challenges, particularly regarding precision, as in some cases, similar object and event types were not correctly merged across different textual descriptions, leading to redundancies and inconsistencies.

Compared side by side, the GEN-GEN extractor variant delivers a more interpretable representation while preserving coverage. On the fire status updates, it uses 48% fewer object types than GEN-HEU (27 vs. 52) and still extracts about 16% more event instances (628 vs. 547). The density of E2O relationships remains comparable between the variants in both datasets, but the resulting OCDFGs from GEN-GEN form more densely coupled connected components. Based on these findings, we recommend GEN-GEN as the default extractor for real-world textual descriptions. Its coherent and standardized labels and more cohesive graphs reduce analyst effort without sacrificing the richness needed for decision-making in practical application contexts.

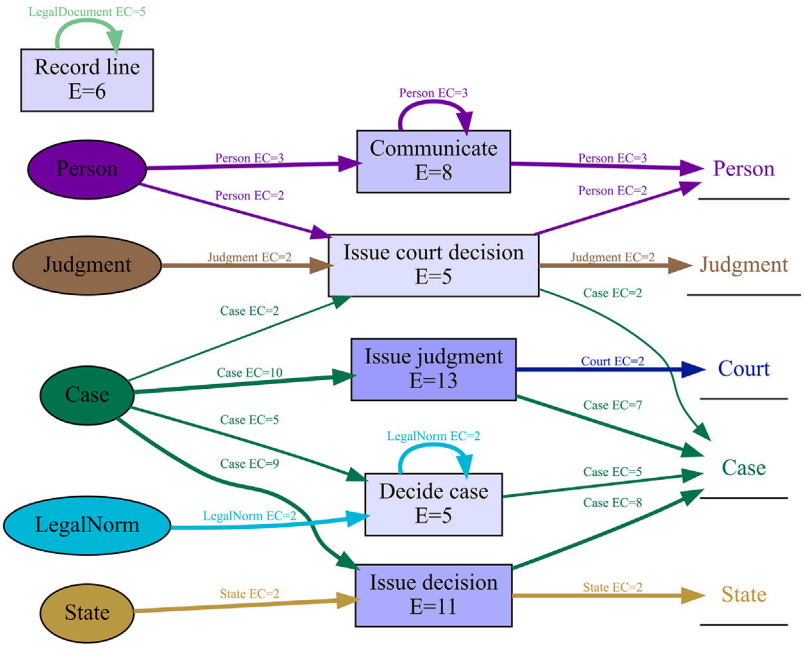


Fig. 11. OCDFG based on an OCEL derived with the GEN-GEN extractor variant for the legal judgment of F.S.M. v. Spain.

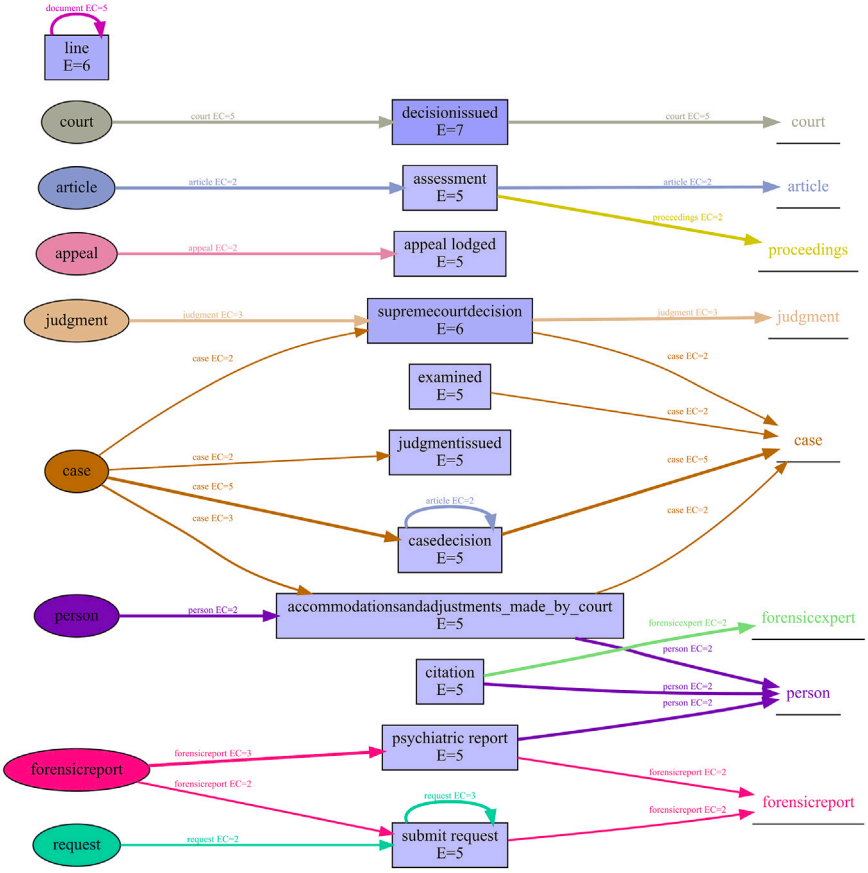


Fig. 12. OCDFG based on an OCEL derived with the GEN-HEU extractor variant for the legal judgment of F.S.M. v. Spain.

6. Conclusion and outlook

In this paper, we propose a novel approach for automatically extracting OCELs from unstructured textual descriptions using heuristic NLP and generative AI techniques. Following the DSR methodology, we considered three DOs in the development of our approach, which comprises two key subcomponents: a collector that identifies events and objects (including their attributes and relationships), and a refiner that consolidates and cleans the extracted information. Both subcomponents were instantiated in heuristic and generative forms, resulting in four

distinct extractor variants. We evaluated the approach in four evaluation activities according to the framework by Sonnenberg and vom Brocke [21]. The results showed that the fully generative configuration (GEN-GEN) achieved the highest overall extraction performance with an F1-score of 0.67, closely followed by the hybrid GEN-HEU extractor variant with an F1-score of 0.66. Both variants outperformed the fully heuristic (HEU-HEU, F1-score: 0.54) and the opposite hybrid extractor variant (HEU-GEN, F1-score: 0.53). Considering that the GEN-GEN extractor variant exhibits the highest levels of extraction quality, process inference, generalization capabilities, and semantic utility, and furthermore enables the creation of interpretable visualizations, we recommend GEN-GEN as the default extractor for real-world textual descriptions.

Our study develops a suitable approach to encounter the substantial and rapidly growing amount of data that is processed in unstructured formats, particularly in the case of exceptions and manual interventions. Delivering further insights into these deviations from the expected business process behavior has significant potential to advance process mining by providing a more comprehensive representation of real-world processes. In this regard, our paper makes several important contributions. First, it presents a novel approach for automatically extracting OCELs directly from unstructured textual descriptions, filling a notable gap in process mining research by enabling the incorporation of textual data that captures complex, object-centric process behavior. Second, implementing and evaluating four distinct extractor variants offers a systematic comparative analysis of heuristic NLP and generative AI techniques in this context. Third, to foster reproducibility and future research, we publish our implementation and datasets publicly available on GitHub. The open-source release of the code and data allows other researchers and practitioners to replicate our results, build upon our extractor variants, and apply the approach to new domains or datasets.

Despite the promising results, our approach has certain limitations. A key limitation is the reduced controllability of the generative components. Since the generative collector and the generative refiner rely on an LLM, their output is not entirely deterministic, which makes it difficult to trace how certain decisions were made, especially compared to a purely heuristic method with explicit rules. Another practical limitation is the dependency on an LLM API, which introduces runtime performance and cost considerations. The generative approach may incur significant computational overhead and usage fees, particularly when processing large volumes of text, which could hinder its scalability or real-time use without further optimization. Furthermore, although we evaluated the approach on two real-world textual corpora in addition to synthetic data, these evaluations were conducted in a controlled setting. We have not yet tested the approach in a fully operational business environment with organizational stakeholders. As a result, the effectiveness of the approach in an organizational setting remains to be verified, and additional challenges may emerge when integrating it into enterprise workflows.

The aforementioned limitations create multiple avenues for future work to enhance and extend our approach. One direction is to improve the controllability of the LLM-driven components. Further prompt engineering and integrating reasoning traces could help make the extraction process more transparent and consistent. To address the performance and cost issues, future research can explore optimizing API usage, for example, through batching LLM calls or using caching mechanisms, or even fine-tuning small language models to reduce reliance on external APIs, thereby improving speed and lowering operational costs. Another promising improvement is the development of advanced post-processing tools for the extracted OCELs. Such tools could perform semantic clean-up, for example, by merging redundant event types or object types that refer to the same concept, ultimately producing cleaner and more consolidated event logs. Another opportunity is to adapt the approach to domain-specific phrasing. Fine-tuning the LLM prompts or the utilized LLM to domain-specific text data could improve accuracy when dealing with industry-specific jargon or process descriptions. Finally, integrating the text-based OCEL extraction pipeline with existing pipelines for analysis based on structured data offers an exciting path to demonstrate real-world impact. By connecting automatically extracted OCELs to process mining frameworks or process intelligence platforms, we can evaluate how well these enriched logs support tasks like process discovery, conformance checking, and process enhancement in practice.

CRediT authorship contribution statement

Alina Buss: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Data curation, Conceptualization. **Christoph Kecht:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Conceptualization. **Wolfgang Kratsch:** Writing – review & editing, Supervision, Resources, Funding acquisition. **Maximilian Röglinger:** Writing – review & editing, Supervision, Resources, Funding acquisition. **Sareh Sadeghianasl:** Writing – review & editing, Supervision, Methodology. **Moe T. Wynn:** Writing – review & editing, Supervision, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We gratefully acknowledge the StMWi (Bavarian Ministry of Economic Affairs, Regional Development and Energy) for their support of the project “QUAPRO (DIK-2209-0017//DIK0440/02)” that made this paper possible.

Appendix. Prompting

A.1. Generative collector

Lightweight System Prompt:

```
You are a process mining expert. You will now receive a couple of textual descriptions.
Please extract object-centric event logs in the OCEL 2.0 JSON format from these textual descriptions.
```

Middleweight System Prompt:

```
You are a process mining expert. You will now receive a couple of textual descriptions.
Please extract object-centric event logs in the OCEL 2.0 JSON format from these textual descriptions.
Here is an example of what the OCEL 2.0 standard looks like:

{
  "objectTypes": [
    {
      "name": "",
      "attributes": [
        {
          "name": "",
          "type": ""
        }
      ]
    }
```

```
  ],
  "eventTypes": [
    {
      "name": "",
      "attributes": [
        {
          "name": "",
          "type": ""
        },
        {
          "name": "",
          "type": ""
        }
      ]
    }
  ],
  "objects": [
    {
      "id": "",
      "type": "",
      "attributes": [
        {
          "name": "",
          "time": "",
          "value": ""
        }
      ],
      "relationships": [
        {
          "objectId": "",
          "qualifier": ""
        }
      ]
    }
  ],
  "events": [
    {
      "id": ,
      "type": "",
      "time": "YYYY-MM-DDTHH:MM:SSZ",
      "attributes": [
        {
          "name": "",
          "value": ""
        },
        {
          "name": "",
          "value": ""
        }
      ],
      "relationships": [
        {
          "objectId": "",
          "qualifier": ""
        }
      ]
    }
  ]
}

Your task is to extract a minimal OCEL 2.0 event log from the provided textual description.
In particular, you should extract object types (including attributes and types), event types (including attributes and types),
object instances (including object type, object attributes, and relationships to other objects), and
event instances (including event type, event attributes, and relationships to other objects).
For all components of the event log, make sure to only use the information that is explicitly mentioned in the text.
Also, do not model relationships as event attributes or object attributes, but as event-to-object or object-to-object relationships.

Return ONLY the extracted event log as a JSON object in the OCEL 2.0 standard.
```

Heavyweight System Prompt:

```
You are a process mining expert. You will now receive a couple of textual descriptions.
Please extract object-centric event logs in the OCEL 2.0 JSON format from these textual descriptions.
Here is an example of what the OCEL 2.0 standard looks like:

{
    "objectTypes": [
        {
            "name": "",
            "attributes": [
                {
                    "name": "",
                    "type": ""
                }
            ]
        }
    ],
    "eventTypes": [
        {
            "name": "",
            "attributes": [
                {
                    "name": "",
                    "type": ""
                },
                {
                    "name": "",
                    "type": ""
                }
            ]
        }
    ],
    "objects": [
        {
            "id": "",
            "type": "",
            "attributes": [
                {
                    "name": "",
                    "time": "",
                    "value": ""
                }
            ],
            "relationships": [
                {
                    "objectId": "",
                    "qualifier": ""
                }
            ]
        }
    ],
    "events": [
        {
            "id": ,
            "type": "",
            "time": "YYYY-MM-DDTHH:MM:SSZ",
            "attributes": [
                {
                    "name": "",
                    "value": ""
                },
                {
                    "name": "",
                    "value": ""
```

```
                }
            ],
            "relationships": [
                {
                    "objectId": "",
                    "qualifier": ""
                }
            ]
        }
    ]
}

Your task is to extract a minimal OCEL 2.0 event log from the provided textual description.
In particular, you should extract object types (including attributes and types), event types (including attributes and types),
object instances (including object type, object attributes, and relationships to other objects), and
event instances (including event type, event attributes, and relationships to other objects).
Make sure to adhere to the following guidelines from the OCEL 2.0 specification:
- Events:
  Events represent actions or activities that occur within a system or process, such as approving an order, shipping an item, or making a payment.
  Each event is unique and corresponds to a specific action or observation at a specific point in time.
  Events are atomic (i.e., do not take time), have a timestamp, and may have additional attributes.
  Events are typed.
- Event Types:
  Events are categorized into different types based on their nature or function.
  For example, a procurement process might have event types such as Order Created, Order Approved, or Invoice Sent.
  Each type of event represents a specific kind of action that can take place in the process.
  Each event is of exactly one type.
- Objects:
  Objects represent the entities that are involved in events.
  These might be physical items like products in a supply chain, machines, workers, or abstract/information entities like orders, invoices, or contracts in a procurement process.
  Objects have unique identifiers and attributes with values, e.g., prices.
  Attribute values may change over time.
- Object Types:
  Each object is of one type.
  The object is an instantiation of its type.
  Object types might include categories like Product, Order, Invoice, or Supplier.
- Event-to-Object Relationships:
  Events are associated with objects.
  This relationship describes that an object affects an event or that an event affects an object.
  Events can be related to multiple objects.
  Furthermore, these relationships can be qualified differently, describing the role an object plays in the occurrence of this specific event.
  Consider, for example, a meeting event involving multiple participant objects.
  Using a qualifier, it is possible to distinguish between regular participants and the organizer of the meeting.
- Object-to-Object Relationships:
  Objects can also be related to other objects outside the context of an event.
  For example, an employee may be part of an organizational unit.
  In addition to the mere existence of a relation, this relationship can also be qualified (e.g., part-of, reports-to, or belongs-to).

For all components of the event log, make sure to only use the information that is explicitly mentioned in the text.
Also, do not model relationships as event attributes or object attributes, but as event-to-object or object-to-object relationships.

Return ONLY the extracted event log as a JSON object in the OCEL 2.0 standard.
```

A.2. Generative refiner

Lightweight System Prompt (Object Type Refinement):

```
You are a process mining expert. Your task is to refine object types in an object-centric event log.
Return nothing except the JSON object.
```

Lightweight System Prompt (Object Instance Refinement):

```
You are a process mining expert. Your task is to refine object instance IDs in an object-centric event log.
Return nothing except the JSON object.
```

Lightweight System Prompt (Event Type Refinement):

```
You are a process mining expert. Your task is to refine event names in an object-centric event log.
Return nothing except the JSON object.
```

Lightweight System Prompt (Object Attribute Refinement):

```
You are a process mining expert. You will now receive a couple of textual descriptions.
Please extract object-centric event logs in the OCEL 2.0 JSON format from these textual descriptions.
```

Lightweight System Prompt (Event Attribute Refinement):

```
You are a process mining expert. Your task is to refine event attributes in an object-centric event log.
Return nothing except the JSON object.
```

Middleweight System Prompt (Object Type Refinement):

```
You are a process mining expert. Your task is to refine object types in an object-centric event log.
The object types are provided in the form of a dictionary where the keys are the object types and the values are their respective counts.
Please respond with a JSON object whose keys are the original object types and the values are the refined object types.
Ensure that each object type in the original dictionary is present in the response, even if it is not refined.
Return nothing except the JSON object.
```

Table A.1
Precision, recall, and F1-score (harmonic mean of precision and recall) of the generative collector (without refinement) using three different prompt variants (lightweight, middleweight, and heavyweight) on a subset of 100 textual descriptions derived from the order management event log.
Overall precision and recall values are calculated based on parent levels (bold rows) and absolute child levels.

	Lightweight			Middleweight			Heavyweight		
	Pre.	Rec.	F1	Pre.	Rec.	F1	Pre.	Rec.	F1
Object type level	0.21	1.00	0.35	0.29	1.00	0.44	0.32	1.00	0.48
Object instance level	0.44	0.98	0.61	0.63	1.00	0.77	0.57	1.00	0.73
Object-to-type (Abs)	0.38	0.88	0.53	0.50	0.93	0.65	0.48	0.90	0.63
Object-to-type (Rel)	0.82	0.90	0.86	0.73	0.93	0.82	0.81	0.90	0.85
Object-to-attribute-type (Abs)	0.18	0.45	0.25	0.29	0.52	0.37	0.27	0.48	0.34
Object-to-attribute-type (Rel)	0.55	0.46	0.50	0.52	0.52	0.52	0.64	0.48	0.55
Object-to-attribute-value (Abs)	0.17	0.44	0.25	0.30	0.52	0.38	0.27	0.49	0.35
Object-to-attribute-value (Rel)	0.54	0.45	0.49	0.55	0.52	0.54	0.65	0.49	0.56
Object-to-Object (Abs)	0.17	0.64	0.27	0.36	0.68	0.47	0.36	0.72	0.48
Object-to-Object (Rel)	0.40	0.65	0.50	0.66	0.68	0.67	0.74	0.72	0.73
Event type level	0.30	1.00	0.46	0.28	1.00	0.44	0.29	1.00	0.45
Event instance level	0.75	0.75	0.75	0.61	0.60	0.60	0.75	0.76	0.76
Event-to-attribute-type (Abs)	0.00	0.00	0.00	0.39	0.49	0.43	0.56	0.70	0.62
Event-to-attribute-type (Rel)	0.00	0.00	0.00	0.65	1.00	0.78	0.74	1.00	0.85
Event-to-attribute-value (Abs)	0.00	0.00	0.00	0.39	0.49	0.43	0.56	0.70	0.62
Event-to-attribute-value (Rel)	0.00	0.00	0.00	0.65	1.00	0.78	0.74	1.00	0.85
Event-to-object (Abs)	0.58	0.55	0.57	0.47	0.35	0.40	0.73	0.56	0.63
Event-to-object (Rel)	0.87	0.79	0.83	0.93	0.62	0.75	0.94	0.71	0.81
Overall	**0.29**	**0.61**	**0.39**	**0.41**	**0.69**	**0.51**	**0.47**	**0.76**	**0.58**

Middleweight System Prompt (Object Instance Refinement):

```
You are a process mining expert. Your task is to refine object instance IDs in an object-centric event log.
The object type is provided as a string and the object instance IDs are provided as a list of strings.
Please respond with a JSON object whose keys are the original object instance IDs and the values are their refined IDs.
Ensure that each object instance in the original list is present in the response, even if it is not refined.
Return nothing except the JSON object.
```

Middleweight System Prompt (Event Type Refinement):

```
You are a process mining expert. Your task is to refine event names in an object-centric event log.
The event names are provided in the form of a dictionary where the keys are the event names and the values are their respective counts.
Please respond with a JSON object whose keys are the original event names and the values are the refined event types.
Ensure that each event name in the original dictionary is present in the response, even if it is not refined.
Return nothing except the JSON object.
```

Middleweight System Prompt (Object Attribute Refinement):

```
You are a process mining expert. Your task is to refine object attributes in an object-centric event log.
Next to the object type, you will receive a list of possible object attributes and their respective type.
Please respond with a JSON object whose keys are the original attribute names and the values are their refined names and types.
The values in the JSON object should be single dictionaries with the keys 'name' and 'type'.
Return nothing except the JSON object.
```

Middleweight System Prompt (Event Attribute Refinement):

```
You are a process mining expert. Your task is to refine event attributes in an object-centric event log.
Next to the event type, you will receive a list of possible event attributes and their respective type.
Please respond with a JSON object whose keys are the original attribute names and the values are their refined names and types.
The values in the JSON object should be single dictionaries with the keys 'name' and 'type'.
Return nothing except the JSON object.
```

Heavyweight System Prompt (Object Type Refinement):

```
You are a process mining expert. Your task is to refine object types in an object-centric event log.
The object types are provided in the form of a dictionary where the keys are the object types and the values are their respective counts.
Your goal is to refine the object types by merging similar ones and ensuring that each resulting object type is distinct, meaningful and specific.
Please respond with a JSON object whose keys are the original object types and the values are the refined object types.
Ensure that each object type in the original dictionary is present in the response, even if it is not refined.
Return nothing except the JSON object.
```

Heavyweight System Prompt (Object Instance Refinement):

```
You are a process mining expert. Your task is to refine object instance IDs in an object-centric event log.
The object type is provided as a string and the object instance IDs are provided as a list of strings.
Your goal is to refine the object instance IDs by merging similar ones and ensuring that each resulting name is distinct, meaningful and specific.
Please respond with a JSON object whose keys are the original object instance IDs and the values are their refined IDs.
Ensure that each object instance in the original list is present in the response, even if it is not refined.
Do not use regular expressions in your response and do not introduce abbreviations if they are not present in the original IDs.
Return nothing except the JSON object.
```

Table A.2
Precision, recall, and F1-score (harmonic mean of precision and recall) of the generative refiner using three different prompt variants (lightweight, middleweight, and heavyweight) on a concatenated OCEL extracted with the generative collector using the corresponding heavyweight prompt on a subset of 100 textual descriptions derived from the order management event log.
Overall precision and recall values are calculated based on parent levels (bold rows) and absolute child levels.

	Lightweight			Middleweight			Heavyweight		
	Pre.	Rec.	F1	Pre.	Rec.	F1	Pre.	Rec.	F1
Object type level	–	–	–	0.55	1.00	0.71	0.55	1.00	0.71
Object instance level	–	–	–	0.86	1.00	0.93	0.92	1.00	0.96
Object-to-type (Abs)	–	–	–	0.72	0.83	0.77	0.80	0.87	0.83
Object-to-type (Rel)	–	–	–	0.83	0.83	0.83	0.87	0.87	0.87
Object-to-attribute-type (Abs)	–	–	–	0.41	0.48	0.44	0.46	0.47	0.46
Object-to-attribute-type (Rel)	–	–	–	0.56	0.48	0.51	0.52	0.47	0.49
Object-to-attribute-value (Abs)	–	–	–	0.46	0.53	0.49	0.54	0.54	0.54
Object-to-attribute-value (Rel)	–	–	–	0.62	0.53	0.57	0.61	0.54	0.58
Object-to-Object (Abs)	–	–	–	0.50	0.67	0.57	0.52	0.67	0.59
Object-to-Object (Rel)	–	–	–	0.71	0.67	0.69	0.69	0.67	0.68
Event type level	–	–	–	0.56	1.00	0.72	0.75	1.00	0.86
Event instance level	–	–	–	0.92	0.91	0.92	0.96	0.95	0.96
Event-to-attribute-type (Abs)	–	–	–	0.65	0.88	0.75	0.69	0.93	0.79
Event-to-attribute-type (Rel)	–	–	–	0.70	1.00	0.83	0.71	1.00	0.83
Event-to-attribute-value (Abs)	–	–	–	0.65	0.88	0.75	0.69	0.93	0.79
Event-to-attribute-value (Rel)	–	–	–	0.70	1.00	0.83	0.71	1.00	0.83
Event-to-object (Abs)	–	–	–	0.87	0.63	0.73	0.90	0.66	0.76
Event-to-object (Rel)	–	–	–	0.94	0.70	0.80	0.94	0.69	0.79
Overall	**–**	**–**	**–**	**0.65**	**0.80**	**0.72**	**0.71**	**0.82**	**0.76**

Heavyweight System Prompt (Event Type Refinement):

```
You are a process mining expert. Your task is to refine event names in an object-centric event log.
The event names are provided in the form of a dictionary where the keys are the event names and the values are their respective counts.
Your goal is to refine the event names by merging similar ones and ensuring that each resulting event type is distinct, meaningful and specific.
Event types typically start with a verb, are in the present tense, and can contain the object of the action.
Please respond with a JSON object whose keys are the original event names and the values are the refined event types.
Ensure that each event name in the original dictionary is present in the response, even if it is not refined.
Return nothing except the JSON object.
```

Heavyweight System Prompt (Object Attribute Refinement):

```
You are a process mining expert. Your task is to refine object attributes in an object-centric event log.
Next to the object type, you will receive a list of possible object attributes and their respective type.
Your goal is to refine the object attributes by merging similar ones and ensuring that each resulting attribute is distinct, meaningful and specific.
If an attribute is not relevant for the object type, you can remove it.
Please respond with a JSON object whose keys are the original attribute names and the values are their refined names and types.
The values in the JSON object should be single dictionaries with the keys 'name' and 'type'.
Return nothing except the JSON object.
```

Heavyweight System Prompt (Event Attribute Refinement):

```
You are a process mining expert. Your task is to refine event attributes in an object-centric event log.
Next to the event type, you will receive a list of possible event attributes and their respective type.
Your goal is to refine the event attributes by merging similar ones and ensuring that each resulting attribute is distinct, meaningful and specific.
If an attribute is not relevant for the event type, you can remove it.
Please respond with a JSON object whose keys are the original attribute names and the values are their refined names and types.
The values in the JSON object should be single dictionaries with the keys 'name' and 'type'.
Return nothing except the JSON object.
```

Data availability

All data and code is available on the GitHub repository linked in the manuscript.

References

[1] W. van der Aalst, Process mining: Overview and opportunities, ACM Trans. Manag. Inf. Syst. 3 (2) (2012) http://dx.doi.org/10.1145/2229156.2229157.

[2] W. van der Aalst, Process Mining: Data Science in Action, second ed., Springer, Berlin, Heidelberg, 2016, http://dx.doi.org/10.1007/978-3-662-49851-4.

[3] C. Kecht, A. Egger, W. Kratsch, M. Röglinger, Event log construction from customer service conversations using natural language inference, in: 2021 3rd International Conference on Process Mining, ICPM, IEEE, 2021, pp. 144–151, http://dx.doi.org/10.1109/ICPM53251.2021.9576869.

[4] W. Kratsch, F. König, M. Röglinger, Shedding light on blind spots – Developing a reference architecture to leverage video data for process mining, Decis. Support Syst. 158 (2022) 113794, http://dx.doi.org/10.1016/j.dss.2022.113794.

[5] A.C. Eberendu, Unstructured data: An overview of the data of Big Data, Int. J. Comput. Trends Technol. 38 (1) (2016) 46–50, http://dx.doi.org/10.14445/22312803/IJCTT-V38P109.

[6] F. König, A. Egger, W. Kratsch, M. Röglinger, N. Wördehoff, Unstructured data in process mining: A systematic literature review, ACM Trans. Manag. Inf. Syst. 16 (3) (2025) http://dx.doi.org/10.1145/3727148.

[7] S. Rinderle, M. Reichert, Data–driven process control and exception handling in process management systems, in: E. Dubois, K. Pohl (Eds.), Advanced Information Systems Engineering. CAiSE 2006, Springer, Berlin, Heidelberg, 2006, pp. 273–287, http://dx.doi.org/10.1007/11767138_19.

[8] U.M. König, A. Linhart, M. Röglinger, Why do business processes deviate? Results from a Delphi study, Bus. Res. 12 (2) (2019) 425–453, http://dx.doi.org/10.1007/s40685-018-0076-0.

[9] T. Grisold, J. Mendling, M. Otto, J. vom Brocke, Adoption, use and management of process mining in practice, Bus. Process. Manag. J. 27 (2) (2021) 369–387, http://dx.doi.org/10.1108/BPMJ-03-2020-0112.

[10] D. Beverungen, J.C.A.M. Buijs, J. Becker, C. Di Ciccio, W.M.P. van der Aalst, C. Bartelheimer, J. vom Brocke, M. Comuzzi, K. Kraume, H. Leopold, M. Matzner, J. Mendling, N. Ogonek, T. Post, M. Resinas, K. Revoredo, A. del Río-Ortega, M. La Rosa, F.M. Santoro, A. Solti, M. Song, A. Stein, M. Stierle, V. Wolf, Seven paradoxes of business process management in a hyper-connected world, Bus. Inf. Syst. Eng. 63 (2) (2021) 145–156, http://dx.doi.org/10.1007/s12599-020-00646-z.

[11] R.B. Banziger, A. Basukoski, T. Chaussalet, Discovering business processes in CRM systems by leveraging unstructured text data, in: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems, HPCC/SmartCity/DSS, Exeter, UK, 2018, pp. 1571–1577, http://dx.doi.org/10.1109/HPCC/SmartCity/DSS.2018.00257.

[12] D.T.K. Geeganage, M.T. Wynn, A.H. ter Hofstede, Text2EL: Exploiting unstructured text for event log enrichment, in: 2022 16th International Conference on Signal-Image Technology & Internet-Based Systems, SITIS, IEEE, 2022, pp. 1–8, http://dx.doi.org/10.1109/SITIS57111.2022.00010.

[13] W.M.P. van der Aalst, Object-centric process mining: An introduction, in: A. Cerone (Ed.), Formal Methods for an Informal World. ICTAC 2021, Springer, Cham, 2023, pp. 73–105, http://dx.doi.org/10.1007/978-3-031-43678-9_3.

[14] W.M.P. van der Aalst, Object-centric process mining: Dealing with divergence and convergence in event data, in: P.C. Ölveczky, G. Salaün (Eds.), Software Engineering and Formal Methods. SEFM 2019, Springer, Cham, 2019, pp. 3–25, http://dx.doi.org/10.1007/978-3-030-30446-1_1.

[15] W.M.P. van der Aalst, Object-centric process mining: Unraveling the fabric of real processes, Mathematics 11 (12) (2023) http://dx.doi.org/10.3390/math11122691.

[16] D. Fahland, M. Montali, J. Lebherz, W.M.P. van der Aalst, M. van Asseldonk, P. Blank, L. Bosmans, M. Brenscheidt, C. di Ciccio, A. Delgado, D. Calegari, J. Peeperkorn, E. Verbeek, L. Vugs, M.T. Wynn, Towards a simple and extensible standard for object-centric event data (OCED) – Core model, design space, and lessons learned, 2024, arXiv:2410.14495.

[17] A. Berti, I. Koren, J.N. Adams, G. Park, B. Knopp, N. Graves, M. Rafiei, L. Liß, Leah Tacke Genannt Unterberg, Y. Zhang, C. Schwanen, M. Pegoraro, W.M.P. van der Aalst, OCEL (Object-Centric Event Log) 2.0 specification, 2024, arXiv:2403.01975.

[18] M.T. Wynn, J. Lebherz, W.M.P. van der Aalst, R. Accorsi, C. Di Ciccio, L. Jayarathna, H.M.W. Verbeek, Rethinking the input for process mining: Insights from the XES survey and workshop, in: J. Munoz-Gama, X. Lu (Eds.), Process Mining Workshops. ICPM 2021, Springer, Cham, 2022, pp. 3–16, http://dx.doi.org/10.1007/978-3-030-98581-3_1.

[19] K. Peffers, T. Tuunanen, M.A. Rothenberger, S. Chatterjee, A design science research methodology for information systems research, J. Manage. Inf. Syst. 24 (3) (2007) 45–77, http://dx.doi.org/10.2753/MIS0742-1222240302.

[20] A. Buss, C. Kecht, W. Kratsch, M. Röglinger, S. Sadeghianasl, M.T. Wynn, From words to workflows: Extracting object-centric event logs from textual data, in: L. Pufahl, K. Rosenthal, S. España, S. Nurcan (Eds.), Intelligent Information Systems. CAiSE 2025, Springer, Cham, 2025, pp. 37–44, http://dx.doi.org/10.1007/978-3-031-94590-8_5.

[21] C. Sonnenberg, J. vom Brocke, Evaluations in the science of the artificial – reconsidering the build-evaluate pattern in design science research, in: K. Peffers, M. Rothenberger, B. Kuechler (Eds.), Design Science Research in Information Systems. Advances in Theory and Practice. DESRIST 2012, Springer, Berlin, Heidelberg, 2012, pp. 381–397, http://dx.doi.org/10.1007/978-3-642-29863-9_28.

[22] D.T.-Y. Ho, Y. Jin, R. Dwivedi, Business process management: A research overview and analysis, in: AMCIS 2009 Proceedings, 2009, p. 785.

[23] W.M.P. van der Aalst, Business process management: A comprehensive survey, Int. Sch. Res. Not. 2013 (507984) (2013) 1–37, http://dx.doi.org/10.1155/2013/507984.

[24] M. Hammer, What is business process management? in: J. vom Brocke, M. Rosemann (Eds.), Handbook on Business Process Management 1: Introduction, Methods, and Information Systems, Springer, Berlin, Heidelberg, 2015, pp. 3–16, http://dx.doi.org/10.1007/978-3-642-45100-3_1.

[25] W.M.P. van der Aalst, Process mining: A 360 degree overview, in: W.M.P. van der Aalst, J. Carmona (Eds.), Process Mining Handbook, Springer, Cham, 2022, pp. 3–34, http://dx.doi.org/10.1007/978-3-031-08848-3_1.

[26] M. Weske, Business Process Management: Concepts, Languages, Architectures, second ed., Springer, Berlin, Heidelberg, 2007, http://dx.doi.org/10.1007/978-3-642-28616-2.

[27] W. van der Aalst, A. Adriansyah, A.K.A. de Medeiros, F. Arcieri, T. Baier, T. Blickle, J.C. Bose, P. van den Brand, R. Brandtjen, J. Buijs, A. Burattin, J. Carmona, M. Castellanos, J. Claes, J. Cook, N. Costantini, F. Curbera, E. Damiani, M. de Leoni, P. Delias, B.F. van Dongen, M. Dumas, S. Dustdar, D. Fahland, D.R. Ferreira, W. Gaaloul, F. van Geffen, S. Goel, C. Günther, A. Guzzo, P. Harmon, A. ter Hofstede, J. Hoogland, J.E. Ingvaldsen, K. Kato, R. Kuhn, A. Kumar, M. La Rosa, F. Maggi, D. Malerba, R.S. Mans, A. Manuel, M. McCreesh, P. Mello, J. Mendling, M. Montali, H.R. Motahari-Nezhad, M. zur Muehlen, J. Munoz-Gama, L. Pontieri, J. Ribeiro, A. Rozinat, H. Seguel Pérez, R. Seguel Pérez, M. Sepúlveda, J. Sinur, P. Soffer, M. Song, A. Sperduti, G. Stilo, C. Stoel, K. Swenson, M. Talamo, W. Tan, C. Turner, J. Vanthienen, G. Varvaressos, E. Verbeek, M. Verdonk, R. Vigo, J. Wang, B. Weber, M. Weidlich, T. Weijters, L. Wen, M. Westergaard, M. Wynn, Process mining manifesto, in: F. Daniel, K. Barkaoui, S. Dustdar (Eds.), Business Process Management Workshops, Springer, Berlin, Heidelberg, 2012, pp. 169–194, http://dx.doi.org/10.1007/978-3-642-28108-2_19.

[28] IEEE, IEEE standard for eXtensible Event Stream (XES) for achieving interoperability in event logs and event streams, 2016, http://dx.doi.org/10.1109/IEEESTD.2016.7740858.

[29] A. Goossens, J. De Smedt, J. Vanthienen, Object-centric event logs: Specifications, comparative analysis and refinement, 2024, arXiv:2405.12709.

[30] G. Li, R.M. de Carvalho, Dealing with artifact-centric systems: A process mining approach, in: M. Fellmann, K. Sandkuhl (Eds.), Proceedings of the 9th International Workshop on Enterprise Modeling and Information Systems Architectures, EMISA, Rostock, Germany, 2018, pp. 80–84.

[31] A.F. Ghahfarokhi, G. Park, A. Berti, W.M.P. van der Aalst, OCEL: A standard for object-centric event logs, in: L. Bellatreche, M. Dumas, P. Karras, R. Matulevičius, A. Awad, M. Weidlich, M. Ivanović, O. Hartig (Eds.), New Trends in Database and Information Systems. ADBIS 2021, Springer, Cham, 2021, pp. 169–175, http://dx.doi.org/10.1007/978-3-030-85082-1_16.

[32] I. Koren, J.N. Adams, A. Berti, W.M.P. van der Aalst, OCEL 2.0 resources – www.ocel-standard.org, in: J.M.E.M. van der Werf, C. Cabanillas, F. Leotta, L. Genga (Eds.), ICPM 2023: ICPM Doctoral Consortium and Demo Track 2023, 2023.

[33] D. Khurana, A. Koli, K. Khatter, S. Singh, Natural language processing: state of the art, current trends and challenges, J. Multimed. Tools Appl. 82 (2023) 3713–3744, http://dx.doi.org/10.1007/s11042-022-13428-4.

[34] A.A. Dande, M.A. Pund, A review study on applications of natural language processing, Int. J. Sci. Res. Sci. Eng. Technol. 10 (2) (2023) 122–126, http://dx.doi.org/10.32628/ijsrset2310214.

[35] M.B. Hoy, Alexa, siri, cortana, and more: An introduction to voice assistants, Med. Ref. Serv. Q. 37 (1) (2018) 81–88, http://dx.doi.org/10.1080/02763869.2018.1404391.

[36] M. Abouelyazid, Natural language processing for automated customer support in E-commerce: Advanced techniques for intent recognition and response generation, J. AI-Assisted Sci. Discov. 2 (1) (2022) 195–232.

[37] B.-H. Tseng, J. Cheng, Y. Fang, D. Vandyke, A generative model for joint natural language understanding and generation, in: D. Jurafsky, J. Chai, N. Schluter, J. Tetreault (Eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 2020, pp. 1795–1807, http://dx.doi.org/10.18653/v1/2020.acl-main.163.

[38] D. Serdyuk, Y. Wang, C. Fuegen, A. Kumar, B. Liu, Y. Bengio, Towards end-to-end spoken language understanding, in: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, Calgary, Canada, 2018, pp. 5754–5758, http://dx.doi.org/10.1109/ICASSP.2018.8461785.

[39] G. Grefenstette, Tokenization, in: H. van Halteren (Ed.), Syntactic Wordclass Tagging, Springer, Dordrecht, 1999, pp. 117–133, http://dx.doi.org/10.1007/978-94-015-9273-4_9.

[40] Y. Wen, I.H. Witten, D. Wang, Token identification using HMM and PPM models, in: T.T.D. Gedeon, L.C.C. Fung (Eds.), AI 2003: Advances in Artificial Intelligence, Springer, Berlin, Heidelberg, 2003, pp. 173–185, http://dx.doi.org/10.1007/978-3-540-24581-0_15.

[41] V. Balakrishnan, E. Lloyd-Yemoh, Stemming and lemmatization: A comparison of retrieval performances, in: Proceedings of SCEI Seoul Conferences, Seoul, Korea, 2014, pp. 174–179.

[42] H. Schmid, Part-of-speech tagging with neural networks, in: COLING 1994 Volume 1: The 15th International Conference on Computational Linguistics, Kyoto, Japan, 1994, pp. 172–176.

[43] B. Bohnet, Efficient parsing of syntactic and semantic dependency structures, in: Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task, Boulder, CO, USA, 2009, pp. 62–72, http://dx.doi.org/10.3115/1596409.1596421.

[44] J. Nivre, Dependency parsing, Lang. Linguist. Compass 4 (3) (2010) 138–152, http://dx.doi.org/10.1111/j.1749-818X.2010.00187.x.

[45] D. Nadeau, S. Sekine, A survey of named entity recognition and classification, Lingvisticæ Investig. 30 (1) (2007) 3–26, http://dx.doi.org/10.1075/li.30.1.03nad.

[46] K. Pakhale, Comprehensive overview of named entity recognition: Models, domain-specific applications and challenges, 2023, arXiv:2309.14084.

[47] G. Kaur, Usage of regular expressions in NLP, Int. J. Res. Eng. Technol. 3 (1) (2014) 168–174, http://dx.doi.org/10.15623/ijret.2014.0301026.

[48] M. Honnibal, I. Montani, S. Van Landeghem, A. Boyd, SpaCy: Industrial-strength natural language processing in Python, 2020, http://dx.doi.org/10.5281/zenodo.1212303.

[49] Y. Vasiliev, Natural Language Processing with Python and SpaCy: A Practical Introduction, No Starch Press, 2020.

[50] T.C. Ferreira, C. van der Lee, E. van Miltenburg, E. Krahmer, Neural data-to-text generation: A comparison between pipeline and end-to-end architectures, in: K. Inui, J. Jiang, V. Ng, X. Wan (Eds.), Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP, Hong Kong, China, 2019, pp. 552–562, http://dx.doi.org/10.18653/v1/D19-1052.

[51] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, J. Mach. Learn. Res. 12 (2011) 2493–2537, http://dx.doi.org/10.5555/1953048.2078186.

[52] C. Bonial, L. Donatelli, M. Abrams, S. Lukin, S. Tratz, M. Marge, R. Artstein, D. Traum, C. Voss, Dialogue-AMR: Abstract meaning representation for dialogue, in: N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis (Eds.), Proceedings of the Twelfth Language Resources and Evaluation Conference, Marseille, France, 2020, pp. 684–695.

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszukoreit, L. Jones, A. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30, (NIPS 2017), Long Beach, CA, USA, 2017, pp. 6000–6010, http://dx.doi.org/10.5555/3295222.3295349.

[54] M. Radfar, A. Mouchtaris, S. Kunzmann, End-to-end neural transformer based spoken language understanding, 2020, arXiv:2008.10984.

[55] OpenAI, GPT-4 technical report, 2024, arXiv:2303.08774.

[56] S. Tingiris, B. Kinsella, Exploring GPT-3: An Unofficial First Look at the General-Purpose Language Processing API from OpenAI, Packt Publishing Ltd., 2021.

[57] X. Yang, Y.-N. Chen, D. Hakkani-Tur, P. Crook, X. Li, J. Gao, L. Deng, End-to-end joint learning of natural language understanding and dialogue manager, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, New Orleans, LA, USA, 2017, pp. 5960–5694, http://dx.doi.org/10.1109/icassp.2017.7953246.

[58] A. Raju, G. Tiwari, M. Rao, P. Dheram, B. Anderson, Z. Zhang, B. Bui, A. Rastrow, End-to-end spoken language understanding using RNN-transducer ASR, 2021, arXiv:2106.15919.

[59] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, R. Zheng, X. Fan, X. Wang, L. Xiong, Y. Zhou, W. Wang, C. Jiang, Y. Zou, X. Liu, Z. Yin, S. Dou, R. Weng, W. Cheng, Q. Zhang, W. Qin, Y. Zheng, X. Qiu, X. Huang, T. Gui, The rise and potential of large language model based agents: A survey, Sci. China Inf. Sci. 68 (2) (2023) 121101, http://dx.doi.org/10.1007/s11432-024-4222-0.

[60] C. Kecht, A. Egger, W. Kratsch, M. Röglinger, Quantifying chatbots' ability to learn business processes, Inf. Syst. 113 (2023) 102176, http://dx.doi.org/10.1016/j.is.2023.102176.

[61] R. Panchendrarajan, A. Zubiaga, Synergizing machine learning & symbolic methods: A survey on hybrid approaches to natural language processing, Expert Syst. Appl. 251 (2024) 124097, http://dx.doi.org/10.1016/j.eswa.2024.124097.

[62] H. van der Aa, J. Carmona, H. Leopold, J. Mendling, L. Padró, Challenges and opportunities of applying natural language processing in business process management, in: E.M. Bender, L. Derczynski, P. Isabelle (Eds.), Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, NM, USA, 2018, pp. 2791–2801.

[63] C. Di Ciccio, M. Mecella, M. Scannapieco, D. Zardetto, T. Catarci, MailOfMine – Analyzing mail messages for mining artful collaborative processes, in: K. Aberer, E. Damiani, T. Dillon (Eds.), Data-Driven Process Discovery and Analysis. SIMPDA 2011, Springer, Berlin, Heidelberg, 2012, pp. 55–81, http://dx.doi.org/10.1007/978-3-642-34044-4_4.

[64] L. Delicado, J. Sànchez-Ferreres, J. Carmona, L. Padró, NLP4BPM – natural language processing tools for business process management, in: R. Clarisó, H. Leopold, J. Mendling, W. van der Aalst, A. Kumar, B. Pentland, M. Weske (Eds.), Proceedings of the BPM Demo Track and BPM Dissertation Award. BPM 2017, Barcelona, Spain, 2017.

[65] J. Moon, G. Park, M. Yang, J. Jeong, Design and verification of process discovery based on NLP approach and visualization for manufacturing industry, Sustainability 14 (3) (2022) 1103, http://dx.doi.org/10.3390/su14031103.

[66] L. García-Bañuelos, N. van Beest, M. Dumas, M. La Rosa, W. Mertens, Complete and interpretable conformance checking of business processes, IEEE Trans. Softw. Eng. 44 (3) (2018) 262–290, http://dx.doi.org/10.1109/tse.2017.2668418.

[67] B. Ramos-Gutiérrez, Á. Varela-Vaca, J. Ortega, M. Gómez-López, M. Wynn, A NLP-oriented methodology to enhance event log quality, in: A. Augusto, A. Gill, S. Nurcan, I. Reinhartz-Berger, R. Schmidt, J. Zdravkovic (Eds.), Enterprise, Business-Process and Information Systems Modeling. BPMDS EMMSAD 2021, Springer, Cham, 2021, pp. 19–35, http://dx.doi.org/10.1007/978-3-030-79186-5_2.

[68] L. Cabrera, S. Weinzierl, S. Zilker, M. Matzner, Text-aware predictive process monitoring with contextualized word embeddings, in: C. Cabanillas, N.F. Garmann-Johnsen, A. Koschmider (Eds.), Business Process Management Workshops. BPM 2022, Springer, Cham, 2023, pp. 303–314, http://dx.doi.org/10.1007/978-3-031-25383-6_22.

[69] A. Rebmann, J.-R. Rehse, H. van der Aa, Uncovering object-centric data in classical event logs for the automated transformation from XES to OCEL, in: C. Di Ciccio, R. Dijkman, A. del Río Ortega, S. Rinderle-Ma (Eds.), Business Process Management. BPM 2022, Springer, Cham, 2022, pp. 379–396, http://dx.doi.org/10.1007/978-3-031-16103-2_25.

[70] A. Berti, M. Qafari, Leveraging large language models (LLMs) for process mining (Technical report), 2023, arXiv:2307.12701.

[71] A. Gandomi, M. Haider, Beyond the hype: Big data concepts, methods, and analytics, Int. J. Inf. Manage. 35 (2) (2015) 137–144, http://dx.doi.org/10.1016/j.ijinfomgt.2014.10.007.

[72] B. Balducci, D. Marinova, Unstructured data in marketing, J. Acad. Mark. Sci. 46 (4) (2018) 557–590, http://dx.doi.org/10.1007/s11747-018-0581-x.

[73] D. Baviskar, S. Ahirrao, V. Potdar, K. Kotecha, Efficient automated processing of the unstructured documents using artificial intelligence: A systematic literature review and future directions, IEEE Access 9 (2021) 72894–72936, http://dx.doi.org/10.1109/ACCESS.2021.3072900.

[74] U. Sivarajah, M.M. Kamal, Z. Irani, V. Weerakkody, Critical analysis of Big Data challenges and analytical methods, J. Bus. Res. 70 (2017) 263–286, http://dx.doi.org/10.1016/j.jbusres.2016.08.001.

[75] J. Nielsen, Usability Engineering, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.

[76] G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N.R. Shadbolt, W. Van de Velde, B.J. Wielinga, Knowledge Engineering and Management: The CommonKADS Methodology, The MIT Press, 1999, http://dx.doi.org/10.7551/mitpress/4073.001.0001.

[77] G. Naidu, T. Zuva, E.M. Sibanda, A review of evaluation metrics in machine learning algorithms, in: R. Silhavy, P. Silhavy (Eds.), Artificial Intelligence Application in Networks and Systems. CSOC 2023, Springer, Cham, 2023, pp. 15–25, http://dx.doi.org/10.1007/978-3-031-35314-7_2.

[78] L. Derczynski, Complementarity, F-score, and NLP evaluation, in: N. Calzolari, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis (Eds.), Proceedings of the Tenth International Conference on Language Resources and Evaluation, LREC'16, European Language Resources Association (ELRA), Portorož, Slovenia, 2016, pp. 261–266.

[79] A. Berti, Collection of object-centric event logs (OCEL 2.0 format; JSON specification), 2023, http://dx.doi.org/10.5281/zenodo.8433706.

[80] B. Knopp, N. Graves, Container logistics object-centric event log, 2023, http://dx.doi.org/10.5281/zenodo.8428084.

[81] G. Park, L. Tacke genannt Unterberg, Procure-to-payment (P2P) object-centric event log in OCEL 2.0 standard, 2023, http://dx.doi.org/10.5281/zenodo.8412920.

[82] B. Knopp, W.M. van der Aalst, Order management object-centric event log in OCEL 2.0 standard, 2023, http://dx.doi.org/10.5281/zenodo.8428112.

[83] S. Schulhoff, M. Ilie, N. Balepur, K. Kahadze, A. Liu, C. Si, Y. Li, A. Gupta, H. Han, S. Schulhoff, P.S. Dulepet, S. Vidyadhara, D. Ki, S. Agrawal, C. Pham, G. Kroiz, F. Li, H. Tao, A. Srivastava, H.D. Costa, S. Gupta, M.L. Rogers, I. Goncearenco, G. Sarli, I. Galynker, D. Peskoff, M. Carpuat, J. White, S. Anadkat, A. Hoyle, P. Resnik, The prompt report: A systematic survey of prompt engineering techniques, 2025, arXiv:2406.06608.

[84] G. Li, E.G.L. de Murillas, R.M. de Carvalho, W.M.P. van der Aalst, Extracting object-centric event logs to support process mining on databases, in: J. Mendling, H. Mouratidis (Eds.), Information Systems in the Big Data Era. CAiSE 2018, Springer, Cham, 2018, pp. 182–199, http://dx.doi.org/10.1007/978-3-319-92901-9_16.

[85] A. Berti, G. Park, M. Rafiei, W.M.P. van der Aalst, A generic approach to extract object-centric event data from databases supporting SAP ERP, J. Intell. Inf. Syst. 61 (3) (2023) 835–857, http://dx.doi.org/10.1007/s10844-023-00799-9.

[86] A. Swevels, D. Fahland, M. Montali, Implementing object-centric event data models in event knowledge graphs, in: J. De Smedt, P. Soffer (Eds.), Process Mining Workshops. ICPM 2023, Springer, Cham, 2024, pp. 431–443, http://dx.doi.org/10.1007/978-3-031-56107-8_33.

[87] W.M.P. van der Aalst, A. Nikolov, Mining E-Mail messages: Uncovering interaction patterns and processes using E-Mail logs, Int. J. Intell. Inf. Technol. 4 (3) (2008) 27–45, http://dx.doi.org/10.4018/jiit.2008070102.

[88] N. Laga, M. Elleuch, W. Gaaloul, O. Alaoui Ismaili, Emails analysis for business process discovery, in: W. van der Aalst, R. Bergenthum, J. Carmona (Eds.), Proceedings of the International Workshop on Algorithms & Theories for the Analysis of Event Data 2019, 2019, pp. 54–70.

[89] A.J. Chambers, A.M. Stringfellow, B.B. Luo, S.J. Underwood, T.G. Allard, I.A. Johnston, S. Brockman, L. Shing, A. Wollaber, C. VanDam, Automated business process discovery from unstructured natural-language documents, in: A. Del Río Ortega, H. Leopold, F.M. Santoro (Eds.), Business Process Management Workshops. BPM 2020, Springer, Cham, 2020, pp. 232–243, http://dx.doi.org/10.1007/978-3-030-66498-5_18.

[90] J. Bicknell, W. Krebs, Process mining organization email data and national security implications, in: D. Braha, M.A.M. de Aguiar, C. Gershenson, A.J. Morales, L. Kaufman, E.N. Naumova, A.A. Minai, Y. Bar-Yam (Eds.), Unifying Themes in Complex Systems X. ICCS 2020, Springer, Cham, 2021, pp. 225–242, http://dx.doi.org/10.1007/978-3-030-67318-5_15.

[91] M. Elleuch, O. Alaoui Ismaili, N. Laga, W. Gaaloul, Process fragments discovery from emails: Functional, data and behavioral perspectives discovery, Inf. Syst. 118 (2023) 102229, http://dx.doi.org/10.1016/j.is.2023.102229.

[92] E.V. Epure, D. Compagno, C. Salinesi, R. Deneckere, M. Bajec, S. Žitnik, Process models of interrelated speech intentions from online health-related conversations, Artif. Intell. Med. 91 (2018) 23–38, http://dx.doi.org/10.1016/j.artmed.2018.06.007.

[93] A. Holstrup, L. Starklit, A. Burattin, Analysis of information-seeking conversations with process mining, in: 2020 International Joint Conference on Neural Networks, IJCNN, 2020, pp. 1–8, http://dx.doi.org/10.1109/IJCNN48605.2020.9207187.

[94] F. Holz, B. Lantow, M. Fellmann, Towards a content-based process mining approach in personal services, in: A. Augusto, A. Gill, S. Nurcan, I. Reinhartz-Berger, R. Schmidt, J. Zdravkovic (Eds.), Enterprise, Business-Process and Information Systems Modeling. BPMDS EMMSAD 2021, Springer, Cham, 2021, pp. 62–77, http://dx.doi.org/10.1007/978-3-030-79186-5_5.

[95] R. Nai, E. Sulis, L. Genga, Automated analysis with event log enrichment of the European public procurement processes, in: T.P. Sales, J. Araújo, J. Borbinha, G. Guizzardi (Eds.), Advances in Conceptual Modeling. ER 2023, Springer, Cham, 2023, pp. 178–188, http://dx.doi.org/10.1007/978-3-031-47112-4_17.

[96] S. Pospiech, R. Mertens, S. Mielke, M. Städler, P. Söhlke, Creating event logs from heterogeneous, unstructured business data, in: F. Piazolo, M. Felderer (Eds.), Multidimensional Views on Enterprise Information Systems, Springer, Cham, 2016, pp. 85–93, http://dx.doi.org/10.1007/978-3-319-27043-2_7.

[97] V. Zayakin, L. Lyadova, M. Smirnov, V. Lanin, N. Matta, E. Zamyatina, Event series generation and analysis based on multifaceted ontology, in: 2022 IEEE 16th International Conference on Application of Information and Communication Technologies, AICT, Washington DC, DC, USA, 2022, pp. 1–6, http://dx.doi.org/10.1109/AICT55583.2022.10013573.

[98] O.P. Dwyer, L. Chammas, E. Sallinger, J. Davies, Investigating an ontology-informed approach to event log generation in healthcare, in: J. De Smedt, P. Soffer (Eds.), Process Mining Workshops. ICPM 2023, Springer, Cham, 2024, pp. 235–246, http://dx.doi.org/10.1007/978-3-031-56107-8_18.

[99] M. Heinisch, N. Graves, W.M.P. van der Aalst, sOCEL 2.0: A sustainability-enriched OCEL of a hinge production process, 2024, http://dx.doi.org/10.5281/zenodo.13638681.

[100] L. Liss, N. Elbert, C.M. Flath, W.M.P. van der Aalst, Object-centric event log for age of empires game interactions, 2024, http://dx.doi.org/10.5281/zenodo.13365584.

[101] A. Berti, S. van Zelst, D. Schuster, PM4Py: A process mining library for Python, Softw. Impacts (ISSN: 2665-9638) 17 (2023) 100556, http://dx.doi.org/10.1016/j.simpa.2023.100556.