Disentangling Event Logs With Large Language Models

Gregory Benton
g.benton@celonis.com
Celonis
New York, New York, USA

Cong Yu
co.yu@celonis.com
Celonis
New York, New York, USA

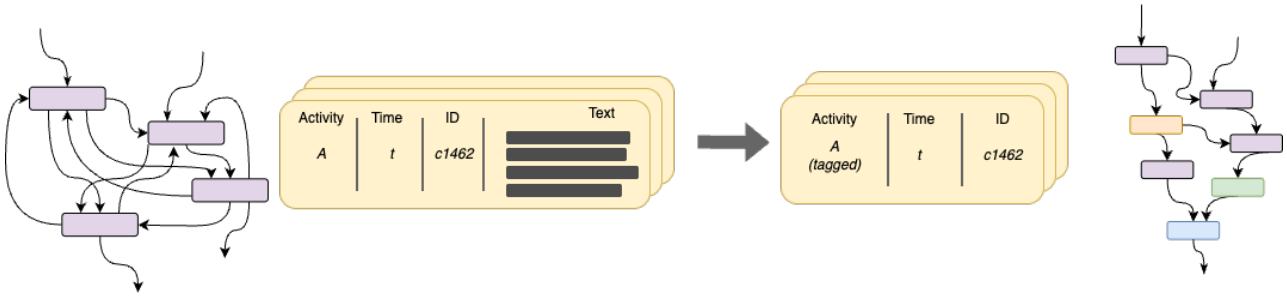


Figure 1: When key aspects of activities are left in secondary textual attributes processes become opaque, with insights and inefficiencies hidden by overloaded labels. By using LLMs for *event log annotation* we increase the resolution of activities and discover a more realistic and accurate process.

Abstract

Process mining is a powerful tool for discovering insights from records of occurrences called event logs. With complex activities, however, event logs may only contain partial records. Key information is often contained in secondary attributes accompanying events. To incorporate this important but overlooked information, we explore *event log annotation* as a method for combining large language model annotations and process mining. Using constructed examples and benchmark datasets we find that annotating event logs can unlock new insights, enabling process mining to better align process discovery with reality.

Keywords

LLMs, Process Mining, Data Annotation, Process Discovery, Large Language Models

ACM Reference Format:
Gregory Benton and Cong Yu. 2025. Disentangling Event Logs With Large Language Models. In *Proceedings of SIGMOD LLM-DPM (SIGMOD '25 LLM-DPM Workshop)*. ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

In many processes, activities in the event log are under-specified, reducing the insights that can be gained through traditional process mining. Consider activities such as *update correspondence* or *receive ticket*; without additional context critical distinctions get masked under broad labels. Is an update correspondence activity indicating that an issue has been resolved, or that more information is needed? Is a received support ticket merely asking for a feature to be enabled, or is a customer experiencing a critical outage?

Without additional context distinctions remain hidden and mined processes become distorted — showing self-loops and redundant activities that don't actually exist, or overlooking sequential steps. To resolve this, we leverage unstructured text at the activity level to annotate and disentangle these ambiguous events, aligning the discovered process to fully reflect the nuance of complex activities.

Process mining's strength lies in discovering meaningful insights directly from data. However, to unlock its full potential, the data must be enriched with the right contextual information. While historically contextual information hidden in unstructured text would be difficult to incorporate into an event log, the introduction of Large Language Models (LLMs) makes such a task straightforward.

In this work we explore *event log annotation* with LLMs. More specifically, we examine the applicability of LLMs for relabeling activities in an event log given additional unstructured information. Through constructed examples, and combinations of real and synthetically generated data we highlight scenarios in which event log annotations can play a key role in process discovery.

Importantly, using annotated event logs to enhance process discovery is fully compatible with existing process mining algorithms. Recent methods, including those designed for use with process mining, have identified the need for data enrichment and use LLM annotations to add additional columns to data schemas. However, since process mining algorithms are typically agnostic to secondary data sources for process discovery. incorporating such information directly would require significant changes to the underlying algorithms themselves [1, 12]. By looking directly to event logs, we are the first to leverage both LLM annotations and process mining algorithms in a single framework.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGMOD '25 LLM-DPM Workshop, Berlin, Germany
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/2025/06
https://doi.org/XXXXXXX.XXXXXXX

Our summarized contributions are as follows

(1) Proposing a framework for the synthesis of LLM-enriched event logs based on unstructured context about events that allows us to improve the granularity of process discovery using existing process mining algorithms (Sections 3.1 and 3.2).
(2) Providing worked examples of event log annotation based on constructed and real-world benchmark event log data (Sections 3.3 and 4).
(3) Formalizing the graphical structures that lead to ambiguities in process models and how they can be resolved with event log annotation (Section 4.2).

2 Related Work

Large Language Models for Annotation. LLMs have proliferated as a tool for data annotation and labeling in both the research and industry communities. Tan et al. [12] and Pavlovic and Poesio [10] both offer comprehensive surveys on LLMs as annotators. Pavlovic and Poesio [10] offer an overview of recent studies on using LLMs for annotation, and additionally provide an exploration of LLM annotation agreement with human judgment for subjective tasks. Tan et al. [12] investigate LLM-based annotation generation specifically as part of an *annotation* → *annotation assessment* → *annotation utilization* pipeline.

Perhaps closest to our own line of inquiry is Chen et al. [5], who examine event extraction using LLMs. Chen et al. [5] work specifically on extracting event labels from unstructured text, but they do not investigate annotating pre-existing event logs as we do here. Nor do they look into the inclusion of LLM annotations as a step in process mining in the presence of unstructured data.

In industrial applications LLMs are also being increasingly deployed for data annotation, curation, and moderation. In content moderation, OpenAI shows that GPT-4 is comparable to human moderators with light training across several categories [9]. The broad utility of automated data annotation has also led to the development of several annotation platforms, such as Scale AI [11].

Large Language Models for Process Mining. Outside of focusing on annotations, several works have investigated the applicability of LLMs to process mining. Berti and Qafari [3] look at leveraging LLMs for process mining, focusing on extracting text summaries of process mining artifacts. They then apply these summaries for LLM powered question and answering, as well as producing LLM generated database queries.

Berti et al. [2] continues the work of Berti and Qafari [3], evaluating the performance of LLMs for process mining across insight generation, code generation, and hypothesis suggestion. Specifically, given process mining data, they report on LLMs ability to contribute to process mining, and the LLM capabilities necessary to for process mining on LLMs.

To better aid users in parsing the information extracted via process mining Kermani et al. [7] propose an LLM integration with process mining tools. Similar to the above works on LLMs for process mining, the LLM in Kermani et al. [7] is used to digest process mining analytics and data and for query generation.

While many works have advanced both LLMs for annotations and for process mining, we believe we are the first to explicitly join these directions and investigate how LLM annotations can enhance process mining.

3 Event Log Annotation with Large Language Models

Here we formalize the notion of event log annotation with a brief introduction to event logs and process mining.

3.1 Event Logs, Process Mining, and Notation

We define an *event* e as a record containing a timestamp t, a case identifier c, and an activity a from the universe of activities A. For many processes, events will also contain some additional attribute dimensions $d_1, \ldots, d_n$ which can be continuous, categorical, or unstructured. We are focused on extracting information from unstructured text attributes, such as messages accompanying correspondence events or notes in hospital treatment processes. Thus for simplicity we consider just one single attribute d_{text} (which may be empty) and write

$$e = (a, c, t, d_{text}), \quad a \in A. \tag{1}$$

While we focus on the parsing of text based features, the multimodality of frontier models means that in principle these features d_{text} could be of much broader range of forms, including images, audio, or structured data [6].

An event log E is simply a collection of events, $E = \{e_1, \ldots, e_N\}$. While process mining is a a broad field that includes conformance checking, and process improvement, we are particularly interested in process discovery [13]. Process discovery is a set of algorithms that parse an event log to generate a graphical representation of the underlying process such as a Petri net, or business process model. By default, the common process discovery algorithms such as α-miner or inductive miner consider the activity, case identifier, and timestamp of an event. The result is that for these algorithms to consider any additional information, the information must first be somehow encoded into these three event attributes.

3.2 Event Log Annotations

Given the structured data associated with process mining and event logs and the need to annotate the events themselves, standard LLM annotation approaches are not out-of-the-box sufficient for all process mining applications. In particular, simply writing an annotation to a new column of the data still leaves underspecified activities opaque from the process perspective. We refer to an activity as *overloaded* or *under-specified* if the activity name does not fully describe the activity. That is, just by knowing two events both occur with activity a, we still believe these events may be distinct in some important sense (which is typically described in d_{text}). If we extend our events to include new annotations $d_{annot.}$ based on d_{text} such that $e = (a, c, t, d_{text}, d_{annot.})$ the context provided by $d_{annot.}$ is not included in the key columns for discovery. Thus the process discovery algorithm that infers the universe of activities A from event log $E = \{e_1, \ldots, e_N\}$ will still discovery the same A had the annotations not existed.

However, if we instead create a new activity column with $e = (a^*, c, t, d_{text})$ where a^* is overwritten with the annotation based on d_{text} then our new universe of activities A^* accounts for the

contents of d_{text}. Through this process-focused annotation, we not only have extracted structure from the unstructured information in d_{text}, but the graphical representation learned through the process discovery algorithm now accounts for the annotation and the mined process is fundamentally changed.

Supposing we want to annotate single type of activity, a_{target}, we can write the event log annotation as the following map:

$$a^* = \begin{cases} a, & \text{if } a \neq a_{\text{target}} \\ f_{\text{LLM}}(d_{text}), & \text{if } a = a_{\text{target}}. \end{cases} \tag{2}$$

When a is an instance of the a_{target} activity, we apply an annotation based on some textual feature d_{text}, otherwise we leave a untouched.

This distinction between adding a new attribute and altering the event log matters ultimately because process mining attempts to infer the underlying structure of a real-world process. If we only create a new attribute through annotations we still group distinct real-world events to the same activity label during mining. Furthermore, if we use this new attribute to filter our data downstream, while we may reduce the collisions of distinct events to the same label, we then lose our ability to see the full process by masking out activities. It is only through annotating our event log, the central data source in process mining, that we can disentangle overloaded and underspecified events.

3.3 Illustrative Example

To elucidate the idea of event log annotations and to demonstrate practical applications we give an example event log in Table 1. This table shows a few events from a hypothetical employee payroll correction process in which one activity is merely "update correspondence" accompanied by a *Message* field (d_{text} in this case) and an associated LLM annotation $d_{text} \rightarrow \{\text{resolved}, \text{hold}\}$. The LLM annotation maps the unstructured text in d_{text} to one of either *resolved* or *hold*, that the message represents a hold being on the correction, or the correction being resolved.

Table 2 shows the proposed event log annotation where we change the *Update Correspondence* activities to be the new annotated activities from A^*, the expanded set of annotation expanded activities. Thus we have

$$\begin{aligned} A^* = A \cup \{&\textit{Update Correspondence (Hold)}, \\ &\textit{Update Correspondence (Resolved)}\}, \\ &\setminus\{\textit{Update Correspondence}\}. \end{aligned} \tag{3}$$

While the difference between Tables 1 and 2 may seem minor, given the deep dependence of process mining on the *Activity*, *Case ID*, and *Timestamp* columns, the precise location of where we include the information from the *Message* field has large implications on the mined process.

Figure 2 continues our example, highlighting the distinction between what standard process mining would yield for the given event log, what we could obtain after using LLM based annotations from the *Message* field, and what can be found when annotating the event log itself. While the original process visualization indicates self loops in the *Update Correspondence* activity and connections from *Form Completion* back to *Update Correspondence*, these connections are misleading. If we use the *Annotation* column from Table 1 to filter down based on the parsed messages, we start to see a clearer picture, but it is still incomplete. The self loop in *Update Correspondence* is gone, as is the connection from *Form Completion* to *Update Correspondence*. However, these options only show half the true picture.

Activity	Case ID	Time-stamp	Message	Annotation
Update Corr.	1	1/4/2025	We have submitted your request and you should receive payment shortly	resolved
Close Case	1	1/7/2025	–	–
Update Corr.	2	1/8/2025	We require an additional form before payment can be sent.	hold
Form Sent	2	1/8/2025	-	-
Update Corr.	2	1/9/2025	Form received, payment will be sent.	resolved
Close Case	2	1/9/2025	–	–
⋮	⋮	⋮	⋮	⋮

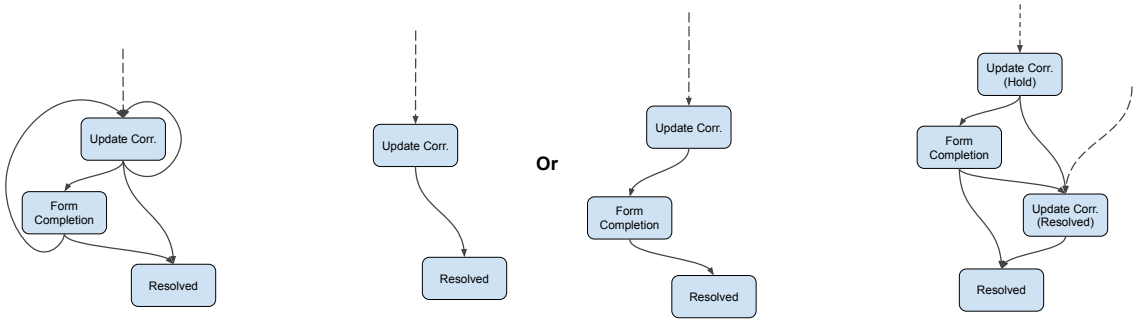
Table 1: Sample event log for a payroll correspondence process with hypothetical annotations for the messages attached to the *update correspondence* activity.

By augmenting our universe of activities A to include the new LLM generated activities allows us to visualize a new process that reveals a more accurate set of insights about how our cases flow. The self-loop and the *Update Correspondence* $\rightarrow$ *Form Completion* $\rightarrow$ *Update Correspondence* in Figure 2a are merely an artifacts of not being able to disentangle overloaded activities. When we properly split out *hold* and *resolved* correspondences, the true sequence of activities emerges.

While the ability to visualize the process accurately is itself an important component of process mining, the true use of process mining is in providing actionable insights around process improvement and automation. In the process mined from the event log in Table 1 bottlenecks cannot be correctly inferred, nor can the process be automated at the *Update Correspondence* step. Without amending the event log itself all algorithms that are based on the event log will fail to capture the nuances of the true process.

4 Case Study: Sepsis Treatment Event Log

We now move to realistic evaluation, applying annotations atop real benchmark data. For data, we use the publicly available sepsis treatment event log, tracking sepsis patients through ER visits [8]. The activities include *ER Triage*, *Admission NC* (normal care), *Admission IC* (intensive care), *Release*, and *Return ER*. Importantly,



(a) Default process map associated with the event log.

(b) Possible visualization of the process when annotating cases and applying filtering.

(c) The true underlying process visualized by annotating the event log prior to mining the process.

Figure 2: (*Left*) Excerpt of the process map given the original event log. While this map aligns with the original activities in the event log, the entanglement of activities creates challenges in generating insights and automations. (*Middle*) By using standard annotation techniques and filtering the event log, we can begin to see clearer patterns based on specific subtypes of activities based on the annotations. (*Right*) By performing event log annotation we form a clear picture of the interaction between events. Paths that initially seem like self loops and rework may be in fact collisions between distinct types of activities. Here we see that the overloading of the Update Correspondence activity creates a confused perspective in the map to the left, which are made clear in the right-most perspective.

Activity	Case ID	Time-stamp	Message
Update Corr. (Resolved)	1	1/4/2025	We have submitted your request and you should receive payment shortly
Close Case	1	1/7/2025	–
Update Corr. (Hold)	2	1/8/2025	We require an additional form before payment can be sent.
Form Sent	2	1/8/2025	-
Update Corr. (Resolved)	2	1/9/2025	Form received, payment will be sent.
Close Case	2	1/9/2025	–
⋮	⋮	⋮	⋮

Table 2: The event log annotated by directly updating the *update correspondence* activity. With the annotation taking over the activity column of the event log, we have a fundamentally new process with a new universe of activities that can be mined.

however, the data do not make any further distinction as to the patient's health status than normal care, intensive care, and releasing a patient.

To adapt this dataset for evaluation of annotations, we first create admission notes to accompany each *Admission NC* event using GPT-4o [6]. These admission notes are then treated as the text features of the event log, d_{text}, and event log annotation is used to parse the admission note into a classification of *low*, *medium*, or *high* — making

$$\begin{aligned} A^* = A \cup \{&\textit{Admission NC (low)},\\ &\textit{Admission NC (medium)},\\ &\textit{Admission NC (high)}\}\\ \setminus\{&\textit{Admission NC}\}. \end{aligned} \tag{4}$$

We give system instructions to the LLM to classify the risk based on admission note, and d_{text} is passed as the chat message. The LLM output is then $d_{LLM} \in \{low, medium, high\}$ and the annotation is written into the event log as *Admission IC* (d_{LLM}). For the full prompts and admission note generation see Appendix A.

4.1 Annotation Analysis

Using Celonis process mining software, we visualize the original (non-annotated) process in Figure 3 [4]. What we observe is a complex set of interactions between *Admission NC* and *Admission IC*. Some cases flow from normal care to intensive care, then back again before being released, but without tracking these cases individually we cannot understand where in the process this transfer from IC to NC happens, or if it is happening multiple times before a patient gets released.

In Figure 4 we show the same events but using the annotated set of *Admission NC* activities. This process tells a different and more complete story than that of Figure 3, allowing us to see that risk levels during admission to normal care are highly predictive of outcomes. Loops appearing in Figure 3 can now be interpreted as a continuum from increased risk levels, to intensive care, to lower

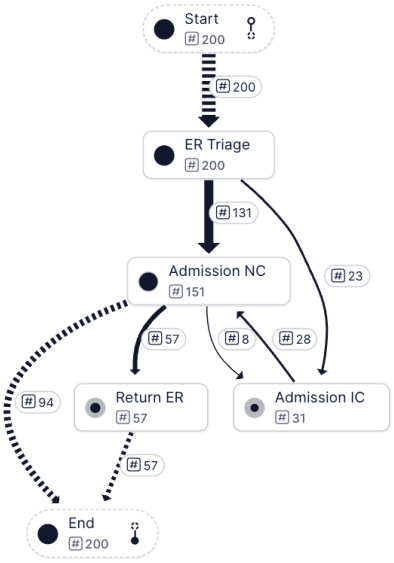


Figure 3: Process visualization associated with the non-annotated sepsis treatment data. Without understanding the details of admissions to normal care, we cannot parse the relationships between normal care, intensive care, and patient releases.

risk levels. Early in the process we can understand the likelihood a patient will need intensive or repeated care. Without distinguishing between risk levels information that may normally be hidden in admission notes, the utility of process mining to help allocate resources or automate monitoring or intervention is diminished.

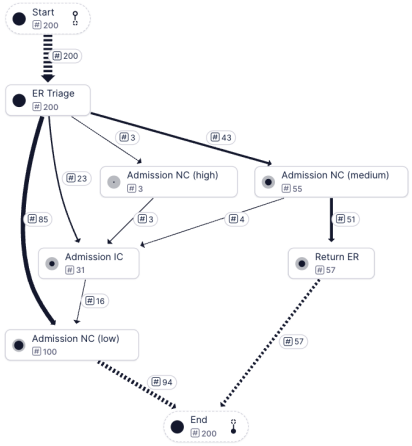


Figure 4: Process visualization associated with the annotated sepsis treatment data. Through categorizing the normal care admissions by risk according to (synthetic) medical notes we see

Beyond visualization improvements, these annotations provide critical context for downstream conformance checking and predictive analytics. For example, by distinguishing between admission risk categories, conformance checking can explicitly highlight deviations from hospital care protocols: Are high-risk patients consistently getting the requisite testing? Do low-risk patients experience unnecessary interventions? With clearer activities, optimization efforts become more precise. Resources can be targeted specifically where the risk annotations show they are most needed, reducing both false positives in conformance checks and inefficient resource allocation. In short, precise annotations directly strengthen process analytics, yielding insights that can better power conformance checks and automations.

4.2 Ontology of Structural Patterns

Given the above example, we now provide an overview of the structural patterns in graphical representations of processes that can be ambiguous due to overloaded activities. We group these structures into three types of ambiguities: rework, fan-in, and fan-out ambiguities. To define these ambiguities we consider the activities from the original event log $\{a, b, c\} \subset A$ and annotated activities $\{a_1, \ldots a_n\, b, c\} \subset A^*$.

Rework and Self-Loop Ambiguity. The first structure that can lead to ambiguity without proper annotation is any type of rework or self-loop in the process. An n-hop path $a \rightarrow a$ may be confused with $a_i \rightarrow a_j$ for any i, j pair. Thus rework may be masking the occurrence of distinct activities.

We observe this pattern in the Sepsis data, where repeated admissions to normal care in Figure 3 are masking a continuum of care from high risk, to low risk, to release (Figure 4).

Fan-In Ambiguity. Fan-in ambiguity occurs when multiple activities converge to a shared downstream activity, $b \rightarrow a$ and $c \rightarrow a$. Without annotation this convergence can be masking either distinct or imbalanced paths. With an annotated event log we may observe $b \rightarrow a_1$ and $c \rightarrow a_2$ or imbalances such that $p(a_1|b) \neq p(a_1|c)$ and $p(a_2|b) \neq p(a_2|c)$. In the annotated sepsis data of Figure 4 we observe with *Admission IC* only being upstream of *Admission NC (low)*, and *ER Triage* having very different probabilities of flowing to the various levels of normal care admissions. This change in the annotated version of events is an instance of fan-in ambiguity in the original event log.

Fan-Out Ambiguity. Similar to fan-in ambiguity, fan-out ambiguity occurs when a single overloaded activity has multiple outcomes, $a \rightarrow b$ and $a \rightarrow c$. Without annotation this divergence can be masking distinct paths $a_1 \rightarrow b$ and $a \rightarrow c$, or imbalances such as $p(b|a_1) \neq p(c|a_1)$ and $p(b|a_2) \neq p(b|a_2)$. In Figure 3 we see that *Admission NC* can flow to *Return ER*, *Admission IC*, or *End*. However, in the annotated version of Figure 4 we observe that only *Admission NC(medium)* leads to *Return ER*, and *Admission NC (low)* only leads to *End*. These changes in the annotated process graph are instances of fan-out ambiguity in the original event log.

5 Discussion

Our work is an early stage investigation, and as such leads to several open questions. Key among these is the impact of activity annotation on process analysis. Through activity annotation we modify the space of activities, and in particular change the cardinality of

this space. Modifying the number of possible activities has meaningful implications on how interpretable a discovered process is — too few activities and critical details become hidden, too many and key patterns are diffused.

In this work, we focus only on data where the desired annotations are known, and the event log is already in place. Through limiting ourselves to these cases, we avoid the degenerate modes of either collapsing or blowing up the activity space. We believe the most critical area of future interest for this work is in relaxing these assumptions and developing a methodology for not only annotation but *discovering* the correct annotations from data. With such a procedure, we could enhance the granularity of process discovery with minimal human intervention and without domain expertise. Such an approach would need much closer scrutiny on its impacts on process visibility. We intend to include such algorithmic development in future versions of this work.

As an additional direction for future work, we aim to explore further evaluation of event log annotation. While we have extended the public Sepsis dataset to include text features for certain events, we intend to do more thorough evaluations on real-world data which can be released with a an accompanying full-length paper.

We have given an initial investigation into event log annotation with LLMs. While process mining alone can be a powerful tool, as can LLM powered annotation, we believe there is overlooked potential at the intersection of these directions. By modifying the event log itself with annotations, the downstream process mining algorithm is able to fully capture nuance that is hidden in secondary attributes. Critically, with the correct setup, event log annotation is compatible with process mining algorithms without modification. Therefore, one can easily apply event log annotation as a preprocessing step in standard process mining pipelines, making it an easy tool for increasing the resolution of process discovery.

Acknowledgments

We thank Stephan Rossbauer, Stefano Pizzoli, Snigdha Shah Deo, Teodora Lata, and Kevin Yang for their helpful discussions.

References

[1] [n. d.]. Celonis Annotation Builder. https://docs.celonis.com/en/annotations-agent.html. Accessed: 2025-03-25.
[2] Alessandro Berti, Humam Kourani, Hannes Häfke, Chiao-Yun Li, and Daniel Schuster. 2024. Evaluating large language models in process mining: Capabilities, benchmarks, and evaluation strategies. In *International Conference on Business Process Modeling, Development and Support*. Springer, 13–21.
[3] Alessandro Berti and Mahnaz Sadat Qafari. 2023. Leveraging large language models (llms) for process mining (technical report). *arXiv preprint arXiv:2307.12701* (2023).
[4] Celonis. 2025. Celonis Process Mining Software. https://www.celonis.com/ Accessed: 2025-03-06.
[5] Ruirui Chen, Chengwei Qin, Weifeng Jiang, and Dongkyu Choi. 2024. Is a large language model a good annotator for event extraction?. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 17772–17780.
[6] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276* (2024).
[7] Mehrdad Agha Mohammad Ali Kermani, Hamid Reza Seddighi, and Mehrdad Maghsoudi. 2024. Revolutionizing process mining: A novel architecture for ChatGPT integration and enhanced user experience through optimized prompt engineering. *arXiv preprint arXiv:2405.10689* (2024).
[8] Felix Mannhardt et al. 2016. Sepsis cases-event log. *Eindhoven university of technology* 10 (2016).
[9] OpenAI. 2023. Using GPT-4 for Content Moderation. https://openai.com/index/using-gpt-4-for-content-moderation/. Accessed: 2025-03-11.
[10] Maja Pavlovic and Massimo Poesio. 2024. The effectiveness of LLMs as annotators: A comparative overview and empirical analysis of direct representation. *arXiv preprint arXiv:2405.01299* (2024).
[11] Scale AI. 2025. *Scale AI Data Engine.* https://scale.com/data-engine
[12] Zhen Tan, Alimohammad Beigi, Song Wang, Ruocheng Guo, Amrita Bhattacharjee, Bohan Jiang, Mansooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. 2024. Large language models for data annotation: A survey. *arXiv e-prints* (2024), arXiv–2402.
[13] Wil Van Der Aalst and Wil van der Aalst. 2016. *Data science in action.* Springer.

A Sepsis Annotation Details

Admission Note Generation. For the generation of synthetic admission notes we use the following system prompt and user message with OpenAI's GPT-4o [6]. Note that these are only for synthetic data generation and tagging, thus classification accuracy is not critically important, and we have not spent meaningful time optimizing these prompts.

System Prompt

```
Your task is to generate a short synthetic text
    snippet to accompany benchmark data on a
    sepsis treatment event log.

The aim is to create synthetic text that will
    accompany the event "Admission to normal care
    ". These texts will later be categorized as
    low, medium, or high probability of needing
    further care.

Your generated note should just be one to two
    sentences. e.g. :
"Patient's ___ is alarming. Observe ____ for
    potential transfer to IC" as high probability
    of needing further care
"Patient stable, give low dose of ___ and release
    in 24h" as a low probability of needing
    further care.
"Patient's condition is improving, consider ___
    for further treatment" as medium probability
    of needing further care.

For each case you will be given 3 things.
i) the number of times the case was admitted to
    intensive care afer this admission to normal
    care (normally 0-2 or 3)
ii) the number of times the case was admitted to
    the ER again after this admission to normal
    care (normally 0-1)
iii) a seed for randomness. Use this to make your
    responses diverse and sometimes not match
```

User message

```
Future IC admissions: <INT>
Future ER returns <INT>
Seed: <INT 0-20>
```

Annotation Generation. For the generation of annotations based on admission notes we use the following system prompt and user message with OpenAI's GPT-4o [6].

```
Your task is to take a note that accompanies
    admission to normal care for a sepsis patient
    and categorize the likelihood that the patient
     needs further care (either moved to IC from
    normal care, or readmission to ER).

If the text indicates that the patient is likely
    to need further care it's probably medium or
    high, if indicating release soon, it's
    probably low.

You should categorize these texts into `low`, `
    medium`, and `high` based on probability of
    increased future care.

Do not output _any other text_. Just `low`, `
    medium` or `high` with no punctuation or
    quotation marks or anything. just text.
```

The user message is simply d_{text} as the admission note.