CSV-PM-LLM-Parsing: Automatic Ingestion of CSV Event Logs for Process Mining using LLMs

Alessandro Berti[1,2,*], Wil M.P. van der Aalst[1,2]

[1]Process and Data Science Chair, RWTH Aachen University, Aachen, Germany
[2]Fraunhofer FIT, Sankt Augustin, Germany

Abstract

This demo paper introduces *CSV-PM-LLM-Parsing*, a Python library to automatically transform unstructured event logs contained in CSV files into process mining event logs by using Large Language Models (LLMs). In particular, we implement modules for three problems: 1) identification of the separator and quote character, 2) selection of the essential process mining columns, and 3) timestamp format detection. Our library is compatible with any LLM supporting the OpenAI's API specification. Our experiment shows that some LLMs (such as *gpt-4o* or the open-source *Qwen/Qwen2-72B-Instruct*) can effectively tackle all three problems. The source code is publicly available in the Git repository https://github.com/fit-alessandro-berti/csv-pm-llm-parsing.

Keywords

Object-Centric Anomaly Detection, Object-Centric Feature Extraction, Procurement Processes, Large Language Models

Process mining is a branch of data science aiming to infer process-related insights from the event data recorded by the information systems supporting the execution of such processes. The most important artifact in process mining is the *event log*, which contains a collection of events extracted from the information systems and allows the application of techniques such as process discovery, conformance checking, model enhancement, and predictive analytics. Standard formats have been proposed for process mining, including XES [1] for traditional event logs (in which an event is related to a single case) and OCEL 2.0 [2] for object-centric event logs (in which an event is related to different objects of different object types). However, in practice, CSV files are often the main data source for process mining, leading to challenges in the detection of the encoding, fundamental parameters (separator and quote character), challenges in the identification of the main columns (case identifier, activity, and timestamp), and challenges in the detection of the timestamp format.

In this demo paper, we propose a Python library to overcome the challenges of importing a CSV for process mining purposes using Large Language Models (LLMs). LLMs, being trained on a vast corpus of data, can resolve the aforementioned challenges. LLMs have been successfully applied to other process mining contexts requiring domain knowledge [3, 4], and can be seen as a zero-shot [5] or few-shot [6] executors for mainstream tasks. The energy consumption of LLMs is concerning and poses sustainability challenges [7], which may discourage their application for tasks that can be resolved with other techniques. However, their application is justified for problems requiring domain knowledge, which also include CSV parsing for process mining purposes.

1. Modules of the Library

In this section, we introduce the three modules of the library. First, we explain the LLM-based approach to identify the separator and quote characters of the CSV file. Then, we describe the LLM-based identification of the case identifier, activity, and timestamp columns. Finally, we discuss the automatic detection of the timestamp column format.

Proceedings of the Best BPM Dissertation Award, Doctoral Consortium, and Demonstrations & Resources Forum co-located with 22nd International Conference on Business Process Management (BPM 2024), Krakow, Poland, September 1st to 6th, 2024.
*Corresponding author.
a.berti@pads.rwth-aachen.de (A. Berti); wvdaalst@pads.rwth-aachen.de (W. M.P. v. d. Aalst)
0000-0002-3279-4795 (A. Berti); 0000-0002-0955-6940 (W. M.P. v. d. Aalst)

Listing 1: Prompt for the identification of the case identifier, activity, and timestamp column in a CSV file.

```
Given the dataframe with the following columns:

Data columns (total 5 columns):
 # Column Non-Null Count Dtype
--- ------ -------------- -----
 0 case_id 6 non-null int64
 1 task 6 non-null object
 2 completion_time 6 non-null object
 3 resource 6 non-null object
 4 cost 6 non-null int64
dtypes: int64(2), object(3)


Can you suggest some columns for the case identifier, activity, and completion timestamp?
Please produce a JSON containing as keys: 'caseid', 'activity', 'timestamp'
Each key should be associated with the name of the column.
```

1.1. Identification of the Separator and the Quote Character

Identification of the separator and the quote character is critical for accurate CSV parsing. The separator distinguishes different fields within a row, while the quote character allows the inclusion of separators within field values. Misidentifying these characters can lead to incorrect data parsing, causing errors in subsequent data processing tasks.

For humans, identifying the separator and quote character can be challenging due to the variability in CSV formats. Common separators include commas, semicolons, tabs, and spaces, while common quote characters include double quotes and single quotes. Users may need to examine the CSV file closely to determine which characters are used, a task that can be time-consuming and prone to error, especially with complex datasets.

Implementing an automatic detector, or *sniffer*, for these characters involves several challenges. The sniffer must accurately analyze the CSV content to infer the correct the separator and quote characters without human input. This requires robust algorithms capable of handling diverse formats and edge cases, such as fields containing separators or quote characters within quoted strings.

In our library, we provide the initial characters of the CSV file to the LLM to identify the separator and quote character. These are used to parse the CSV file into a Pandas dataframe. If the parsing fails, the prompt is executed again with the recorded error until valid separator and quote characters are provided.

However, in some situations, no suitable separator and quote character could be discovered because the CSV is malformed. In that situation, the parsing can be attempted by ignoring the malformed rows.

1.2. Identification of the Main Attributes (Case Identifier, Activity, and Timestamp)

Detecting which columns correspond to the case identifier, activity, and timestamp when ingesting a CSV for process mining is crucial because these elements are the foundation for creating an event log. The case identifier groups events into specific instances of a process, the activity column specifies the actions taken, and the timestamp provides the chronological order of events.

End users may face several challenges in accurately identifying the case identifier, activity, and timestamp columns in a CSV file. One common difficulty is the variability in naming conventions, where column names may not be intuitive or standardized, making it hard to discern their purposes. Additionally, users might not have domain expertise, leading to confusion about which columns should serve as case identifiers or timestamps, especially if the data includes multiple date or ID fields.

Current automated techniques to detect the case identifier, activity, and timestamp columns in a CSV file for process mining primarily rely on two approaches. The first approach uses regular expression matching to identify these columns based on their names, attempting to match patterns that suggest they represent case IDs, activities, or timestamps. For the second category, different techniques have been proposed in the literature. For example, [8] and [9] focus on the identification of the case identifier column based on statistical techniques. In [10] and [11], deep-learning-based techniques have been used to detect the case identifier, activity, and timestamp columns. However, the computational cost of

these models is high. In [12], a meta-model is built starting from a relational database that allows the discovery/recommendation of a case notion. However, the technique is not applicable to standard CSVs.

In our library, we provide the name and data type of the columns to the LLM, requesting identification of the case identifier, activity, and timestamp columns (a corresponding prompt is shown in Fig. 1). If the recognition fails, the prompt is executed again until a valid combination of columns is identified.

1.3. Timestamp Format Detection

In process mining, converting timestamp strings to timestamp objects is essential because it ensures accurate temporal analysis and sequencing of events within the process. This conversion allows for the calculation of durations, intervals, and the order of activities, which are critical for identifying process inefficiencies and optimizing workflows.

However, end users often struggle to specify the correct format of timestamp columns due to variations in date and time representations, leading to errors in data interpretation.

Implementing automated approaches to address this challenge is also difficult, as it requires robust algorithms capable of correctly identifying and parsing a wide range of timestamp formats, accounting for regional differences, inconsistencies, and potential ambiguities in the data. For instance, libraries like *dateutil* in Python and Java's *java.time* package provide built-in functions to parse multiple date and time formats. However, they fail on non-common formats.

In our library, we provide the top values of the timestamp column to the LLM, which proposes a format. The timestamp column is then parsed according to this proposed format. If parsing fails, the resulting error is given to the LLM, which then proposes an alternative format.

The proposed approach fails when the timestamp column contains values of mixed types (for example, some values follow ISO8601, and some follow RFC1123). In that case, a line-by-line parsing needs to be adopted, which is impractical on LLMs.

2. Installation

The library can be installed in Python $\geq$ 3.9.x executing the command **pip install -U csv-pm-llm-parsing**. The source code is publicly available in the Git repository https://github.com/fit-alessandro-berti/csv-pm-llm-parsing. The library can use as backend LLM any advanced LLM exposing the OpenAI's API specification.

To setup the connection to the LLM, the `openai_api_url`, `openai_api_key`, and `openai_model` should be provided to the methods of the library. Alternatively, the parameters could be set up in the system environment variables **OPENAI_API_URL**, **OPENAI_API_KEY**, and **OPENAI_MODEL**.

Some example settings are provided:

- OpenAI's GPT-4O:
  `openai_api_url` = 'https://api.openai.com/v1'; `openai_api_key` = 'replace'; `openai_model` = 'gpt-4o';
- Locally run Ollama[1]:
  `openai_api_url` = 'http://127.0.0.1:11434/v1'; `openai_api_key` = 'replace'; `openai_model` = 'qwen2:7b-instruct-q6_K';
- DeepInfra's open-source model[2]:
  `openai_api_url` = 'https://api.deepinfra.com/v1/openai/'; `openai_api_key` = 'replace'; `openai_model` = 'Qwen/Qwen2-72B-Instruct';

[1] https://ollama.com/library/qwen2:7b-instruct-q6_K
[2] https://deepinfra.com/Qwen/Qwen2-72B-Instruct

3. Initial Experiments

We generated some synthetic datasets that can be used to test the library. In particular:

- https://github.com/fit-alessandro-berti/csv-pm-llm-parsing/tree/main/testfiles/sep_detection contains some examples of common choices for separator and quote character (i.e., command and double quote, semicolon and single quote, tab and single quote, ...).
- https://github.com/fit-alessandro-berti/csv-pm-llm-parsing/tree/main/testfiles/cid_acti_timest contains some test cases for the selection of the case identifier, activity, and timestamp columns.
- https://github.com/fit-alessandro-berti/csv-pm-llm-parsing/tree/main/testfiles/timest_format contains some test cases for the identification of the timestamp format (containing formats such as ISO8601, UNIX, RFC1123, ...).
- https://github.com/fit-alessandro-berti/csv-pm-llm-parsing/tree/main/testfiles/overall contains more complex test cases to test the overall functioning of the library.

We propose in the folder https://github.com/fit-alessandro-berti/csv-pm-llm-parsing/tree/main/examples some examples that use the library for every one of the provided test cases.

In general, we found that larger LLMs (such as *gpt-4o* and *Qwen/Qwen2-72B-Instruct*) can execute the tasks satisfactorily, while smaller LLMs (such as *qwen2:7b-instruct-q6_K*), while still competent, fail on different test cases.

A video showcasing the library and its functioning on some test cases is available at https://youtu.be/L21sapgCzbM.

References

[1] M. T. Wynn, W. M. P. van der Aalst, E. Verbeek, B. N. D. Stefano, The IEEE XES standard for process mining: Experiences, adoption, and revision [society briefs], IEEE Comput. Intell. Mag. 19 (2024) 20–23.

[2] I. Koren, N. Adams, A. Berti, OCEL 2.0 resources - www.ocel-standard.org, CoRR abs/2403.01982 (2024). arXiv:2403.01982.

[3] A. Berti, D. Schuster, W. M. P. van der Aalst, Abstractions, scenarios, and prompt definitions for process mining with llms: A case study, in: BPM 2023 Workshops, volume 492, Springer, 2023, pp. 427–439.

[4] A. Berti, H. Kourani, H. Hafke, C. Yun-Li, D. Schuster, Evaluating Large Language Models in Process Mining: Capabilities, Benchmarks, Evaluation Strategies, and Future Challenges, in: Proceedings of the BPM-DS 2024 Working Conference, Springer, 2024.

[5] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa, Large language models are zero-shot reasoners, in: S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, A. Oh (Eds.), NeurIPS 2022, 2022.

[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. K. et al., Language models are few-shot learners, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[7] A. S. Luccioni, S. Viguier, A. Ligozat, Estimating the carbon footprint of bloom, a 176b parameter language model, J. Mach. Learn. Res. 24 (2023) 253:1–253:15. URL: http://jmlr.org/papers/v24/23-0069.html.

[8] A. A. Andaloussi, A. Burattin, B. Weber, Toward an automated labeling of event log attributes, in: J. Gulden, I. Reinhartz-Berger, R. Schmidt, S. Guerreiro, W. Guédria, P. Bera (Eds.), BPMDS 2018 and EMMSAD 2018, Held at CAiSE 2018, Proceedings, volume 318 of *Lecture Notes in Business Information Processing*, Springer, 2018, pp. 82–96.

[9] A. Burattin, R. Vigo, A framework for semi-automated process instance discovery from decorative attributes, in: CIDM 2011 Proceedings, IEEE, 2011, pp. 176–183.
[10] S. Sim, R. A. Sutrisnowati, S. Won, S. Lee, H. Bae, Automatic conversion of event data to event logs using CNN and event density embedding, IEEE Access 10 (2022) 15994–16009.
[11] K. Toyoda, R. G. K. Ying, A. N. Zhang, P. S. Tan, Identifying the key attributes in an unlabeled event log for automated process discovery, IEEE Trans. Serv. Comput. 17 (2024) 74–81.
[12] E. G. L. de Murillas, H. A. Reijers, W. M. P. van der Aalst, Case notion discovery and recommendation: automated event log building on databases, Knowl. Inf. Syst. 62 (2020) 2539–2575. URL: https://doi.org/10.1007/s10115-019-01430-6.