Demonstration Paper

PROLIT: Supporting the Transparency of Data Preparation Pipelines through Narratives over Data Provenance

Pasquale Leonardo Lazzaro
Università Roma Tre
Italy
pasqualeleonardo.lazzaro@uniroma3.it

Marialaura Lazzaro
Università Roma Tre
Italy
mar.lazzaro1@stud.uniroma3.it

Paolo Missier
University of Birmingham
United Kingdom
p.missier@bham.ac.uk

Riccardo Torlone
Università Roma Tre
Italy
riccardo.torlone@uniroma3.it

ABSTRACT

Establishing trust in the models is a long-standing objective in Machine Learning and AI. Information on how data are manipulated before being used for training is instrumental in such understanding, and data provenance can be used to organize and navigate such information. The **PROLIT** system described in this demo paper is designed to collect, manage, and query the provenance of data as it flows through data preparation pipelines in support of data science analytics and machine learning modeling.

PROLIT extends our prior work on transparently collecting data provenance in several directions. Most notably, it employs a LLM to: (i) automatically rewrite user-defined pipelines in a format suitable for this activity, (ii) segment the code to precisely associate provenance to each code snippet, and (iii) provide human-readable descriptions of each snippet that in turn can be used to generate *provenance narratives*. The demo will showcase these capabilities and offer the opportunity to interact with **PROLIT** on user-defined as well as pre-defined Python code where dataframes are used as the common data abstraction.

1 INTRODUCTION

The provenance of tabular data that flows through a data science pipeline in preparation for use by machine learning algorithms reflects transformations that alter both their schema and values. In this process, every operation performed on the data can have a significant impact on the final results. Thus, understanding the history of each piece of data provides transparency, reproducibility, and control over the quality of both the data used for training and, consequently, also over the final outcome of the ML algorithm trained on those data. For example, provenance helps identify the root cause for errors that manifest themselves at some later stage in the processing.

The **PROLIT** system collects and stores the provenance of data preparation pipelines efficiently and offers users natural language interaction with the provenance storage, both when submitting a question and by returning short *provenance narratives* that incorporate a description of the operations performed on the data. Our approach relies on the PROV provenance model [5] to record the relationships between data and operations, but it also uniquely extends PROV to capture provenance at different levels of granularity within the table. Data transformations are tracked through any operation available in standard data manipulation libraries (e.g., Pandas and Scikit-learn), preserving the paths to data elements at multiple granularities, namely atomic value, row, column, and whole table. This multi-granular view of the data flows provides a balance between flexibility, efficiency, and detail in provenance analysis.

This work follows a long tradition of systems for collecting provenance from scripts and workflows, which has been summarized in multiple surveys over the years, e.g. [4, 7] and with notable examples both old [3] and newer [6, 8]. **PROLIT** follows from its predecessor, DPDS [1, 2], adding multi-granularity support but also integrating large language models (LLMs) to analyze the operations performed on the data and to create a detailed and structured representation of their transformations. This analysis produces a "provenance path" for each data item, which records exactly how and when each value was modified, generated, or deleted. The LLM analyzes the flow of operations and, through the automatic identification of data manipulation patterns, builds a graph that represents the process of creation and modification of each piece of data.

Following PROV, the graphs include Entity and Activity nodes, representing data values and operations respectively, but also new node types to represent table rows and columns. Correspondingly, graph edges represent produce/consume relationships between data and operations, but also structural inclusions for table elements. Users can customize the granularity of provenance representation, allowing them to focus analysis at their preferred level of abstraction while optimizing the volume of stored data.

In the rest of the paper, we briefly present the main features of **PROLIT** and its logical architecture. We then illustrate an outline of our demo proposal and sketch some conclusions.

2 PROLIT OVERVIEW

2.1 Provenance Model

In this work, we focus on data provenance for pipelines that perform typical data manipulation operations on tabular data with a schema, embodied for example by the ubiquitous Python Pandas *dataframes*. A dataframe is a set of *records*, each with atomic *values* for a collection of *attributes* (or *columns*). A *pipeline* is a sequence of data manipulation *operations* on a dataframe that transform, delete, or create values and are designed to reshape the dataframe for downstream uses, typically as training sets or for further analysis.

In this framework, we build on the PROV model [5] to graphically represent provenance information, as shown in Figure 1. The core model describes the effect of data manipulations as a graph consisting of entity and activity nodes and their typed

relationships. The entity nodes represent individual values in the records, while the activity nodes indicate operations on the data. Pre-defined types relationships are used to capture data consumption (used), creation (GeneratedBy), and removal (InvalidatedBy), as well as the derivation (DerivedFrom) of an entity (column) from another entity (column) in the case of a data transformation.

We extend PROV to also include Column nodes, allowing provenance representation at a coarser granularity. Correspondingly, we introduce the belongsTo relationship that associates dataframe cells to a Column. We also introduce explicit sequencing between activities using the new next relationship.

This approach provides a complete representation of data provenance, independently of the type of operators in the pipeline and the specific library used to implement them.

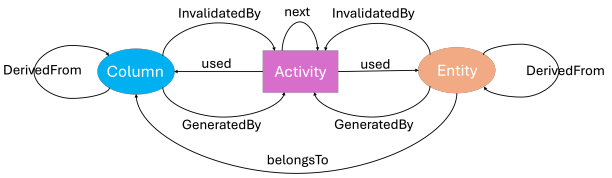


Figure 1: The data provenance model

2.2 Data provenance granularity

In building our system, we start from the observation that a primary limitation of data provenance systems, especially those that track individual elements of a dataset, is the excessive volume of provenance data, which complicates the readability and interpretability of the provenance graph. **PROLIT** addresses this problem by enabling customization of the level of granularity to which the provenance is collected and queried. Provenance can thus be captured and observed at the desired level of detail, as described next.

- The Sketch Level provides a high-level view of the provenance, showing activities and a single entity node for each relationship; useful for a quick overview of how a pipeline operates on data.
- Derivation Level incrementally adds, at the sketch level, DerivedFrom links between entities, allowing a selective exploration of dependencies by expanding specific nodes.
- Full Level offers the most detailed view of the provenance, including all entities, activities, and relationships between them.
- Columns Level captures transformations at the schema level, showing only activity nodes, column nodes, and relationships between them, thus disregarding the effect of the operations on individual data items.

Recording provenance at the full level results in a complex and potentially overwhelming representation of the data history and lineage. In contrast, less detailed levels offer more readable representations that let users focus on essential details, simplifying analysis or understanding.

2.3 The role of LLMs in PROLIT

PROLIT leverages Large Language Models (LLMs) in several tasks of the overall process of data provenance collection and management, as described next.

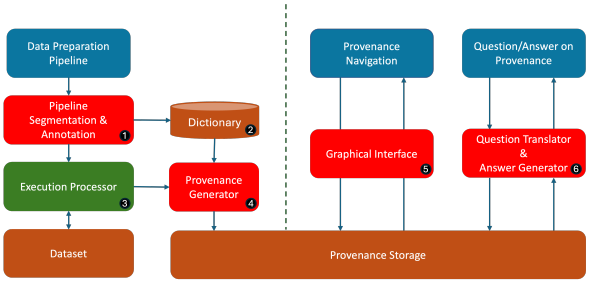


Figure 2: PROLIT architecture

(1) Firstly, a LLM is used to transform the input pipeline into a standardized structure, segmenting the pipeline into distinct activities and adding detailed, inline comments that describe each data manipulation operation within the code. This activity facilitates the identification of the operations and the collection of the provenance they produce. It also provides a clear and explicit explanation of each pipeline activity.

(2) Secondly, the LLM identifies columns used in each activity, a task often challenging to achieve by merely analyzing the pipeline code, and automatically generates and stores, in a dictionary, high-level descriptions of each operation, explaining its intent and impact. This task ensures the availability of comprehensive documentation that accurately reflects the operations performed, significantly enhancing the reproducibility and transparency of the pipeline.

(3) Finally, the LLM is also used to provide a high-level interaction with the system based on natural language questions and answers. This follows the Retrieval-Augmented Generation (RAG) approach by translating user-defined questions into queries over the data provenance collected during pipeline execution and generating an answer by using the query result in the context of the LLM.

In the following section, we describe how the various features described above are implemented in **PROLIT**.

3 SYSTEM DESCRIPTION

3.1 System Architecture

The logical architecture of **PROLIT** is shown in Figure 2.

The input to the system is a script s that implements a pipeline of operations. The first component ❶ performs, with the help of an LLM, a rewriting of s where the various activities are identified and documented by annotating the original script. This component also produces an internal dictionary ❷ that includes a detailed description of each operation involved in the pipeline. The new script is then executed in the running environment ❸ for which it was originally developed. The provenance generator ❹ analyzes the effect of each operator in the pipeline at execution time by: (i) comparing the input(s) and output datasets for that command; and (ii) producing a provenance graph fragment that captures the dependencies for the data elements that have changed, reflecting the specific change pattern and injecting the operation descriptions stored in the dictionary; and (iii) writing the resulting provenance fragment into a graph database (in this implementation we use Neo4j[1]).

[1]https://neo4j.com/

The complete provenance graph that becomes available at the end of execution can be analyzed using native, direct access to Neo4j ❺ and associated graph visualization, but also using a new component ❻ that translates user questions expressed in natural language into Cypher queries over the underlying graph database and returns the result in the form of textual *narratives* about provenance.

In components ❶ and ❻ of **PROLIT** we have used llama-3.1-8B as Large Language Model and Langchain[2] as mediator between the LLM and the rest of the system, facilitating both the collection and analysis of provenance data.

3.2 PROLIT in Action

Let us consider, as a simple example, the following (toy) dataset in Figure 3, where we ultimately wish to use Country, Age, and Salary to predict whether or not a customer has made a Purchase. Missing values are present, such as the age for the third row.

Country	Age	Salary	Purchased
France	44.0	72000.0	No
Spain	27.0	48000.0	Yes
Germany	—	54000.0	No
Italy	—	27000.0	No

Figure 3: Sample Dataset

The following typical sequence of operations, simplified here for the sake of the example, first creates a new dataframe that only contains the predictors, and then adjusts this new dataframe by imputing the missing values for Age.

```
# Separate features and target variable
df = df.iloc[:, :-1]
# Impute missing values in the Age column
df['Age'].fillna(df['Age'].mean(), inplace=True)
```

The result is shown in Figure 4.

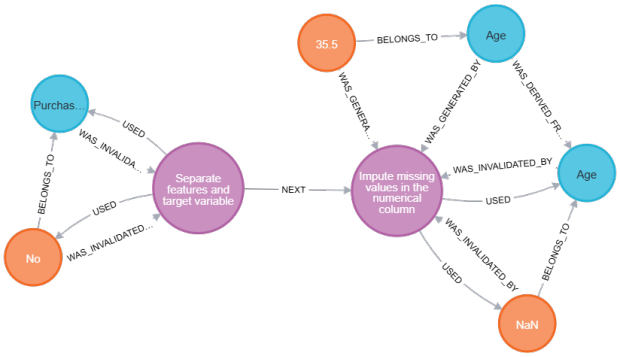
Country	Age	Salary
France	44.0	72000.0
Spain	27.0	48000.0
Germany	35.5	54000.0
Italy	35.5	27000.0

Figure 4: Sample Dataset after pipeline execution

Figure 5 shows fragments of the resulting provenance graphs at each level of detail. Through this visual interface, users can choose their preferred level of detail and incrementally inspect the graph by selectively expanding on each of the nodes.

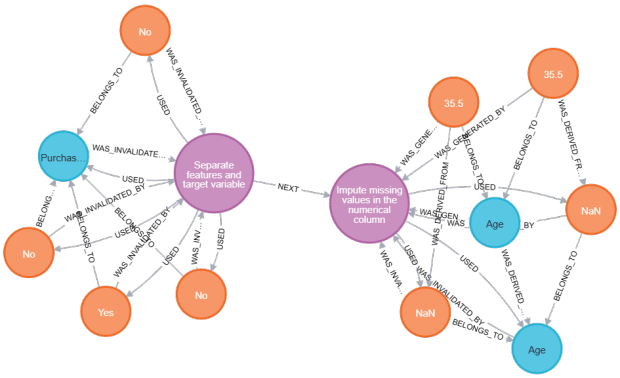
In addition, Figure 6 shows an example of **PROLIT**'s ability to interact with the provenance graph using natural language and to obtain detailed explanations. This capability is particularly valuable as it allows users to obtain insights into the data manipulation process in an intuitive and accessible manner, making it comprehensible even for individuals with limited technical expertise.

[2]https://www.langchain.com/langchain

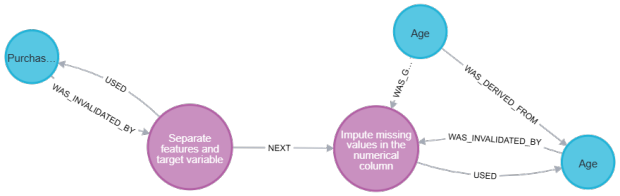


(a) Sketch-Level Provenance: High-level overview, showing activities and single entity nodes per relationship.

(b) Derivation-Level Provenance: Expands the sketch level by adding DerivedFrom links between entity nodes.



(c) Complete Provenance: Full provenance with all entities, activities, and relationships.



(d) Columns-Level Provenance: Focuses on schema-level transformations.

Figure 5: Granularity levels for the data provenance of the example pipeline. Orange nodes represent entities, blue nodes represent columns, and purple nodes represent activities. Nodes can be further inspected by clicking on them.

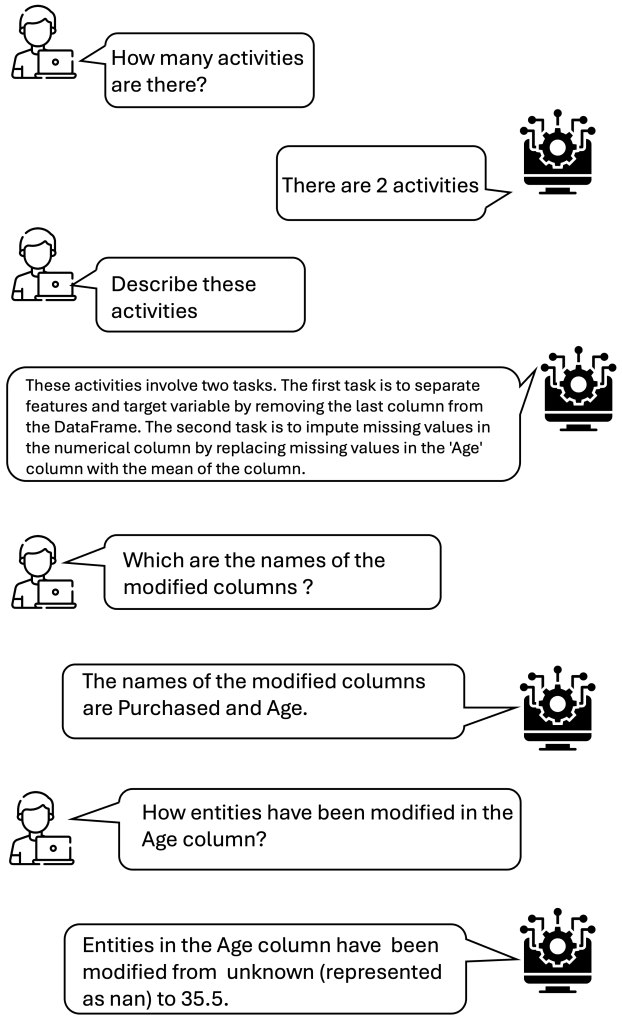


Figure 6: An example of how PROLIT provides insights on a pipeline through narratives over provenance data.

4 DEMO OUTLINE

In our demonstration, we will simulate typical scenarios of data preprocessing in data science aimed at giving a comprehensive view of **PROLIT** and leading to discussions on supporting data science processes with provenance data.

Scenario A: Provenance Capture. In this scenario, we demonstrate how **PROLIT** captures provenance for some predefined processing pipelines. The audience will be invited to choose one of those pipelines and see how the provenance graph is automatically built with negligible overhead. This demonstrates how transparently and efficiently provenance can be easily integrated into data flows with our system.

Scenario B: Configuring Provenance Granularity. Building on the provenance collected from Scenario A, this scenario allows users to adjust the granularity of the provenance, as discussed in the previous sections. Users can explore how different levels of detail impact the structure and depth of the provenance graph, allowing for a tailored approach based on the specific needs of the analyst.

Scenario C: Interactive Provenance Narratives. In this scenario, we showcase the capability of **PROLIT** to produce narratives over the provenance graph. Users will be invited to interact with the system by asking specific questions about the effect of the input pipeline and commenting on the responses provided by the system. This interactivity allows the audience to find out how provenance data can provide meaningful insights into pipelines and comprehensible explanations of their effect on data.

The audience will be able to iterate through Scenarios A, B, and C to explore the impact of the choices made in Scenario A to Scenarios B and C.

5 CONCLUSION

In this paper, we have illustrated the development of a system, called **PROLIT**, for collecting and managing data provenance in data preparation pipelines to support the subsequent activity of machine learning. **PROLIT** addresses some of the critical needs for transparency and interpretability in machine learning systems, as those become key requirements for MLOps and AI deployment.

PROLIT is designed to support the most common data manipulation operators, as found in the most spread data manipulation libraries for Python. It is designed to be extensible and flexible, with support from pre-trained LLM to identify logical units of computation within a script and provide narratives about provenance. **PROLIT** offers a way to track data manipulations at three different levels of granularity. These are recorded into a Neo4J graph database, which can then be queried to generate suitable data-centric explanations.

Our demonstration aims to show the distinctive capabilities that **PROLIT** can offer to data scientists, its minimal impact on the data science process, and its ability to provide insight and explanations about data manipulation pipelines innovatively and easily.

ACKNOWLEDGMENTS

This work is partially supported by the PNRR-MUR project PE0000013-FAIR and by the PRIN-MUR project 2022XERWK9-S-PIC4CHU.

REFERENCES

[1] Adriane Chapman, Luca Lauro, Paolo Missier, and Riccardo Torlone. 2024. Supporting Better Insights of Data Science Pipelines with Fine-grained Provenance. *ACM Trans. Database Syst.* 49, 2, Article 6 (2024). https://doi.org/10.1145/3644385

[2] Adriane Chapman, Paolo Missier, Luca Lauro, and Riccardo Torlone. 2022. DPDS: Assisting Data Science with Data Provenance. *PVLDB* 15, 12 (2022), 3614 – 3617. https://doi.org/10.14778/3554821.3554857

[3] Boris Glavic and Gustavo Alonso. 2009. Perm: Processing Provenance and Data on the Same Data Model through Query Rewriting. In *Proc. of ICDE*. IEEE Computer Society, 174–185. https://doi.org/10.1109/ICDE.2009.15

[4] Melanie Herschel, Ralf Diestelkämper, and Houssem Ben Lahmar. 2017. A survey on provenance: What for? What form? What from? *The VLDB Journal* 26, 6 (2017), 881–906. https://doi.org/10.1007/S00778-017-0486-1

[5] Paolo Missier, Khalid Belhajjame, and James Cheney. 2013. The W3C PROV family of specifications for modelling provenance metadata. In *Proc. of EDBT (Tutorial)*. ACM. https://doi.org/10.1145/2452376.2452478

[6] Mohammad Hossein Namaki, Avrilia Floratou, Fotis Psallidas, Subru Krishnan, Ashvin Agrawal, Yinghui Wu, Yiwen Zhu, and Markus Weimer. 2020. Vamsa: Automated Provenance Tracking in Data Science Scripts. In *Proc. of KDD*. ACM, 1542–1551. https://doi.org/10.1145/3394486.3403205

[7] João Felipe Pimentel, Juliana Freire, Leonardo Murta, and Vanessa Braganholo. 2019. A survey on collecting, managing, and analyzing provenance from scripts. *Comput. Surveys* 52, 3 (2019), 1–38. https://doi.org/10.1145/3311955

[8] Sebastian Schelter, Joos-Hendrik Böse, Johannes Kirschnick, Thoralf Klein, and Stephan Seufert. 2018. Declarative metadata management: A missing piece in end-to-end machine learning. In *Proc. of SysML Conference*.