Automatic data repair for ML pipelines

Parth Asawa, Shreya shankar, professor Aditya g. parameswaran

Developing a low-latency, scalable solution using LLMs to automatically perform data repair on tabular data.

Background

- **Problem**
  - Data is often **dirty** in tabular ML pipelines due to:
    - Distribution shift
    - Corruptions in data ingestion
    - Violating basic constraints
    - Suffering from a software bug
  - There's a lot of **manual work** that ML engineers take on to enumerate different constraints or heuristics for data cleaning in existing pipelines and this doesn't always result in **improved performance**.
- **Goal**
  - How do we match that performance or exceed it with **minimal human intervention**?
  - → Evaluate the utility of **LLMs** as a method to present an automated and inexpensive solution for enabling tabular data repair in the best way possible.

Prior Data Repair Work

Dependency Networks	Probabilistic Methods	Other
ERACER	SCARE (SCalable Automatic REpairing)	BoostClean
KATARA	HoloClean	BigDansing

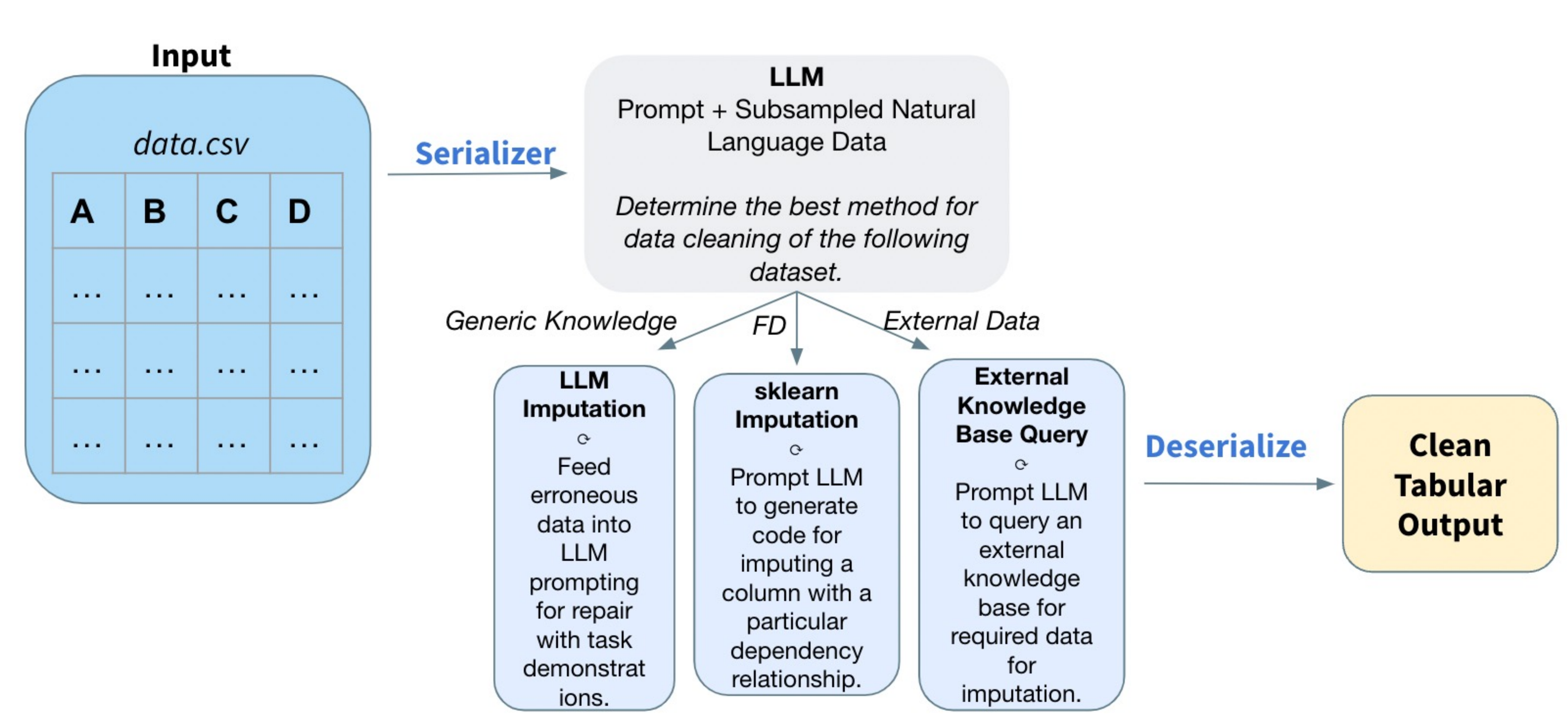
- **Key Takeaways for Good Performance**
  - **Scalability Constraints**: Many systems contain a human-in-the-loop component and are designed with an "on-call engineer" in mind.
  - Reliance on **external knowledge bases.**
  - **Minimality** in data repair: less edits is preferable.
  - **Partitions**: ML pipelines experience a continuous inflow of data. Creating partitions of data and evaluating partitions over time can be more robust.
  - **Temporal robustness:** datasets often contain temporal patterns whether by seasons, weeks, days, etc. that data repair algorithms can falsely try to correct.

LLMs x Tabular Data

- **Key benefits** (Narayan et al., 2022):
  - Task-agnostic architecture
  - Encoded knowledge
  - Limited to no labeled data required
- Applying LLMs to Data Tasks consists of 3 main steps:
  - **Tabular Data Serialization:** adapting structured data inputs to textual inputs.
  - **Converting Data Cleaning/Integration Tasks** to Natural Language Tasks.
  - **Task Demonstrations**: constructing optimal demonstrative task examples to help the FM learn new data tasks (or fine tuning).
- Limitations
  - Different prompts lead to high variance in performance for different tasks.
  - Lack of domain specificity.
  - Cost and privacy.

LLMs x Data Cleaning Architecture

- We propose a high level architecture that follows these key principles to applying LLMs to Data Tasks to solve a general data cleaning problem for a nonspecific dataset.



Research Directions

- **Prompts**
  - Effective and Smaller Prompts: Provide more information than just the column types when it comes to metadata, e.g. leverage column names, partition summaries, etc.
- **Task Demonstration Selection**
  - Context of few-shot learning
  - Determine systematic ways to select small samples of data cleaning task examples that should be passed into an LLM in the prompt as context (ex. kNN).
- **Chain of Thought Reasoning**
  - Using an LLM directly isn't always the best way to data clean, when there is for example a complex FD.
  - → Prompt LLMs to determine which method of data imputation would be best for a particular dataset. Use the resulting answer to generate relevant repair code or directly impute the value with an agent.

Other Considerations

- **Cost:** we seek to explore the cheapest and most democratizable ways LLMs can be leveraged.
- **Privacy:** open source models that can be locally hosted pose a lower risk to privacy than that of large models only accessible through API calls.
- **Time:** we hope to reduce the time required in the overall data cleaning process.