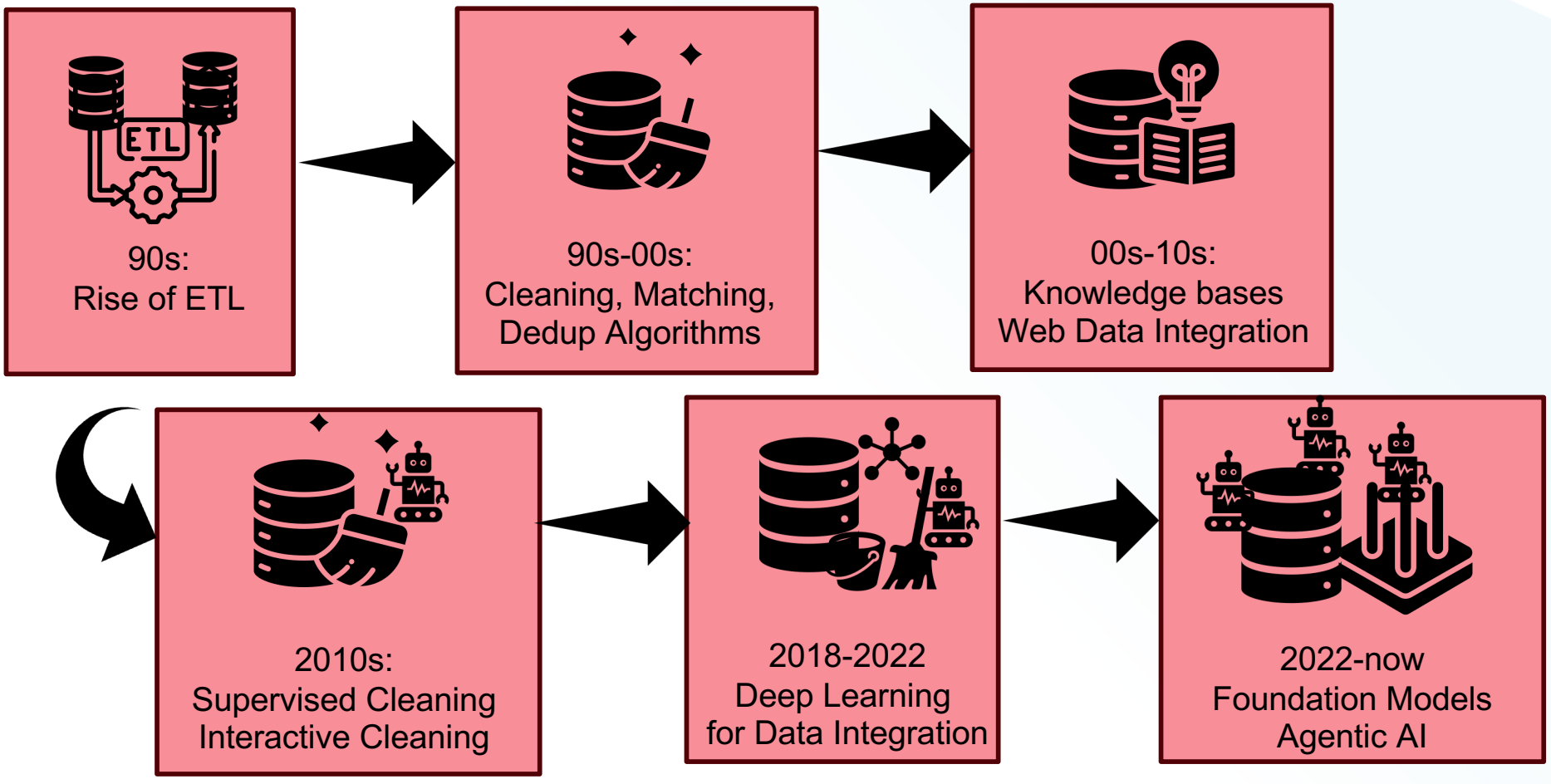
Navigating Disruption: The Impact of AI Technologies on Data Integration Research

Ziawasch Abedjan

www.bifold.berlin

BIFOLD

Data Integration Epochs



The LLM Epoch - Now

Still developing

- New Models come out every few weeks
- A lot of traction with all fields of science
- Reproducibility suffers greatly

Stressful hunt for low-hanging fruits

- There is an LLM angle to nearly any problem
- Very high expectations

Risks and Cost

- Obfuscation of methodology and black box agentic systems
- Intransparent computation costs

Some Anecdotes on how our research was affected

Data Transformation Discovery

- Given a set of input/output examples, identify a

Data Cleaning

- Detecting and repairing errors

Pipeline Generation

- Given a dataset and a task generate an effective pipeline

Data Transformation Discovery

Programming-by Example (PBE)

(Rishabh Singh, Sumit Gulwani:
Learning Semantic String Transformations from Examples. Proc. VLDB Endow. 5(8): 740-751 (2012))

- Synthesizes a series of operators to map each input to ist output
- Works well for syntactic String transformation
- Can do semantic look-ups in small scope

Search-by-Example (SBE)

(Z. Abedjan, J. Morcos, M. N. Gubanov, I. F. Ilyas, M. Stonebraker, P. Papotti, M. Ouzzani:
Dataxformer: Leveraging the Web for Semantic Transformations. CIDR 2015)

- Find resources that implicitly (tables) or explicitly (code, services) represent transformation functions
- Look-up is scalable
- Has difficulties adapting to syntactic transformations

Syntactic transformation

01.01.2025	2025-01-01
12.24.2022	2022-12-24
15.08.2008	2008-08-15

Semantic Transformation

Germany	DE
France	FR
Italy	IT

Bringing SBE and PBE together

Problems to solve:

- How to design the algorithmic interaction between the two approach and the two search spaces?
- How to tokenize the input/output values?

GPT3.5 enters the race:

- Asking a model to find associations by example works like a charm!
- Renders entire though-process for efficient algorithms obsolete

Transform the following list of comma-separated input values the same way "FRA 02/27/1986" is transformed to "Frankfurt 27.02.1986" and "LHR 10/12/2004" is transformed to "London 12.10.2004"

"BER 11/2/2024", "CDG 12/12/1990", "ORD 10/27/2000"

Here is the transformed list:

1. Berlin 02.11.2024
2. Paris 12.12.1990
3. Chicago 27.10.2000

Should we still work on transformation discovery?

YES

Provenance is important
Scalability is important

FlashGPT3

(G. Verbruggen, V. Le, S. Gulwani: Semantic programming by example with pre-trained models. (OOPSLA): (2021))

BIFOLD

Interleaving LLMs with PBE:

- Splits the entire transformation into substrings
- Leverages the LLM as external thesaurus for semantic transformation
- Prefers Syntactic operators over LLM calls.
- This beats all baselines.

The LLM is invoked to find the association between an entire input String and a particular output token.

Two Shortcomings:

- Not every token from an input is relevant,
- The order of output tokens might get lost

Challenging Example for FlashGPT3

CEO Apple Cook ⇨ CEO Apple, AAPL Cook
CFO TSLA Taneja ⇨ CFO Tesla, TSLA Taneja
CTO Amazon Vogels ⇨ ?

- Prefers syntactic operator over semantic operator
- Uses syntactic operator wherever it is possible, even if it will add more semantic operators

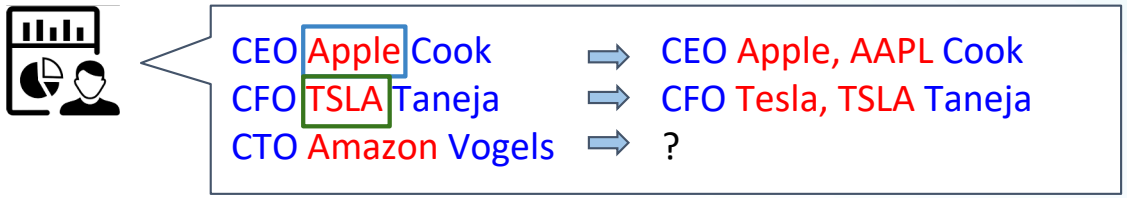
FlashGPT3 Workflow

Query LLM to infer “Apple” from input string

Append a constant comma

Query LLM to infer “AAPL” from input string

Our fix to the problem: identify Non-separable tokens



Our solution works for the specific class of **interchangable transformations:**

- Two values adhere to a **one-to-one relationship**
- Two types of values appear **interchangeably** in the input
- A **unified pattern** involving **both types** of values with **specific order** required for the output

Transformation Discovery: Open Challenges

Scaling is still an issue

- FlashGPT3 reasonably prefers syntactic operators over semantic ones.
- Main problem still: Large space of programs

Inconsistent input examples

- LLMs are confident
- When you ask them to find associations between inconsistent concepts, they will confidently do!

Numeric patterns are still difficult

Opportunities for augmentation via LLMs:

- Treasure trophe for defining transformation tasks.

Data Cleaning

Detecting errors in tabular data and repairing them.

Methods	Strengths	Weaknesses
Unsupervised	No human intervention required High precision	Low recall Limited ability to detect diverse error types
Supervised	Effective for single-table error detection	Require large labelled datasets
Semi-Supervised (e.g., Raha*)	Balance manual effort and automation Require fewer labelled examples than supervised methods	Still require some labelled data

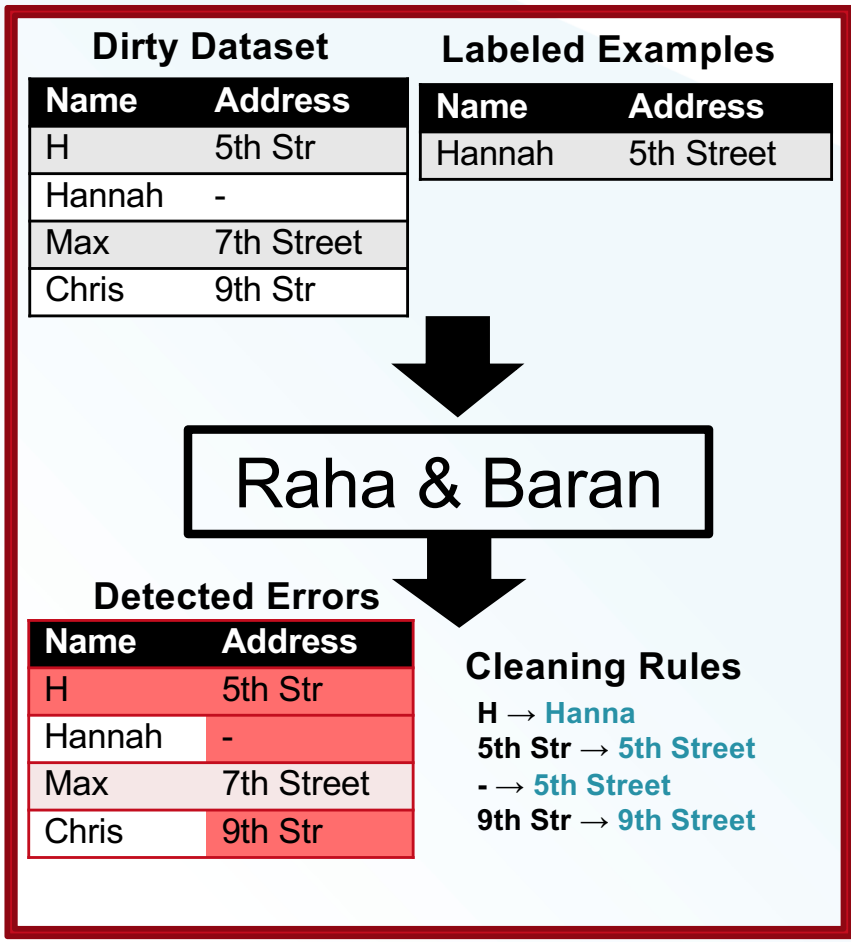
Semi-Supervised Cleaning

Traditional data cleaning:
- Rule-based
- Requires a lot of training data

Challenges for example-based (few-shot) techniques:
- Class imbalance (most of the data is clean) impacts sampling
- Samples must cover different error types
- Corrections must include out-of-vocabulary solutions
- Different modalities of detectors/correctors

Raha for detection:
- Detector-based embedding to distinguish data point quality

Baran for correction:
- Ensemble of correctors that learn from corrections and non corrections



SIGMOD'19, PVLDB'20, CIDR'21

BIFOLD

In Reality we clean multiple Tables

Requires rules or training data per table

No predefined relationships between tables

Challenges in Extending Raha for Multi-Table Error Detection

Error Detection in Raha: A Binary Classification Task – One Classifier per Column

- Too many classifiers, high labeling effort
  - Sharing classifiers
    - Impossible, schema-dependent features
  - Sharing labels
    - When and how?

Multi-Table Error Detection with Matelda

Fatemeh Ahmadi, Marc Speckmann, Malte F. Kuhlmann, Ziawasch Abedjan:
MaTElDa: Multi-Table Error Detection. EDBT 2025: 364-376

BIFOLD

Challenges

Sharing Classifiers

Schema-dependent features

Sharing labels

When and how?

Matelda's Solution

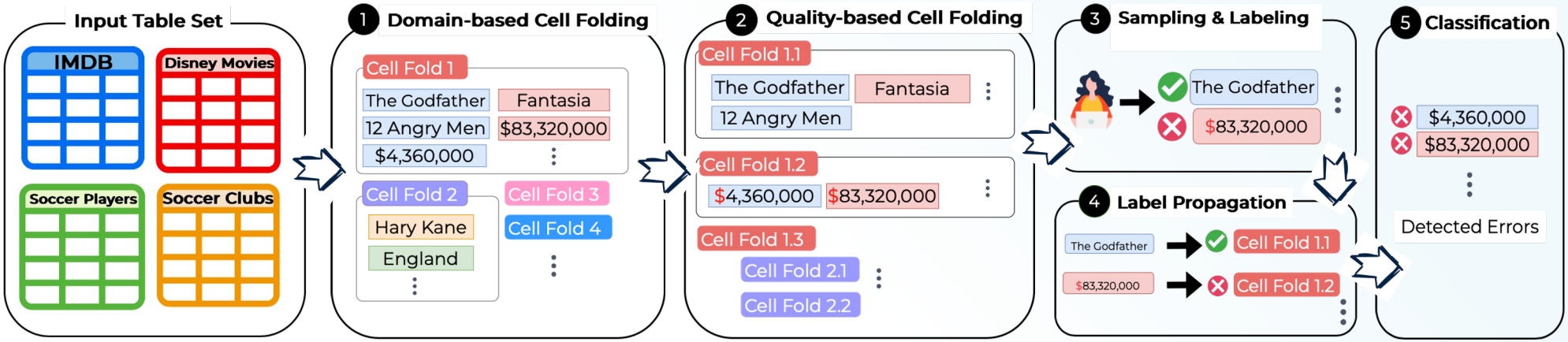
Organizing the table set:

- By semantic similarity
- By quality similarity

Creating a unified feature space

Matelda enables label sharing, reducing manual effort.

Matelda's Workflow



Step 2: Unifying Feature Space

Goal:
Enable cross-table cell value comparison.

How?

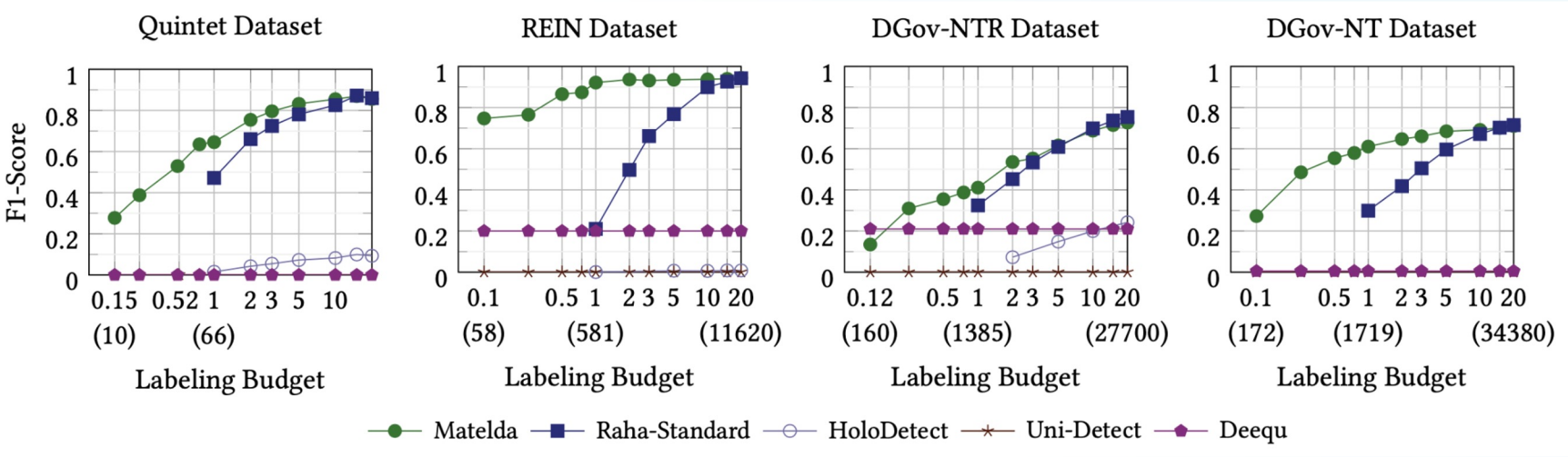
Unifying applies to:

- Pattern violation features
- Functional dependency violation features

Example: Align FDs across tables.

- FD detectors per column:
  - First Column FD.
  - Adjacent Columns FD.
- Aggregated FD Signals

Experiments



Matelda outperforms all competitors until a labeling budget of ten labeled tuples per table.

The Chances with pretrained models and LLMs

Zero-Shot:

- GPT-4 can easily detect spelling issues.
- Detecting outliers needs a set of clean examples.
- Semantic errors are doomed to fail

Few-Shot (No real magic here)

- We really need to pick very good samples: Cover all types of errors
- Picking clean and dirty samples for all types of errors is on its own a challenge.
- Danger of overconfident generalization

Fine-Tuning [TableGPT, SIGMOD'24]

- Helps to better understand table structure
- Still cannot generalize across different error patterns

Pipeline Generation

Natural fit for agentic systems

- Given a dataset and a task, assemble an effective pipeline

Table Understanding is at the core of pipeline generation

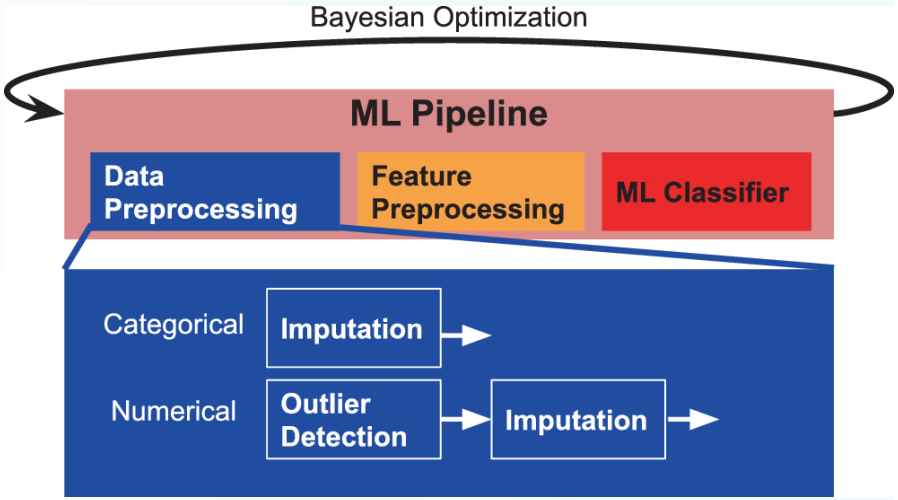
Given a table and a task:

- Identify the most important areas of the table
- Chain of tables inspired by chain of throughts
- Still not really solved

Our Research Pre-LLM

AutoML

- Given a dataset and a task, assemble an effective pipeline



More approaches possible:

- SAGA: Genetic algorithms

S. Siddiqi, R. Kern, M. Boehm: SAGA: A Scalable Framework for Optimizing Data Cleaning Pipelines for Machine Learning Applications. SIGMOD (2023

- AutoPipeline: RL and Search

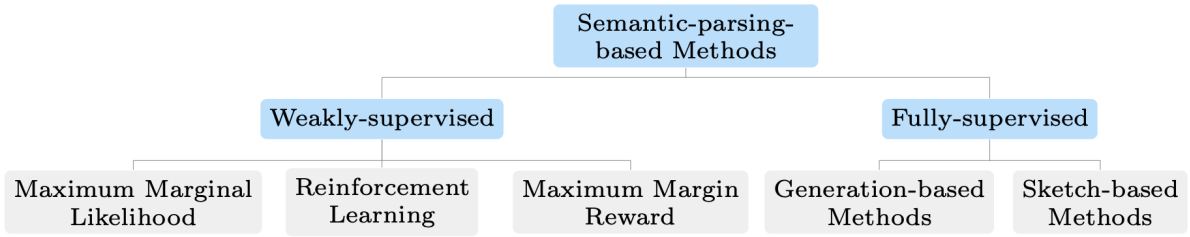
J. Yang, Y. He, S. Chaudhuri: Auto-Pipeline: Synthesize Data Pipelines By-Target Using Reinforcement Learning and Search. PVLDB 2021

Limitations: Do not generalize well, Need explicit training data, limited set of operators

Huge Push towards Semantic Parsing

Semantic Parsing is a simpler problem than ETL pipelines

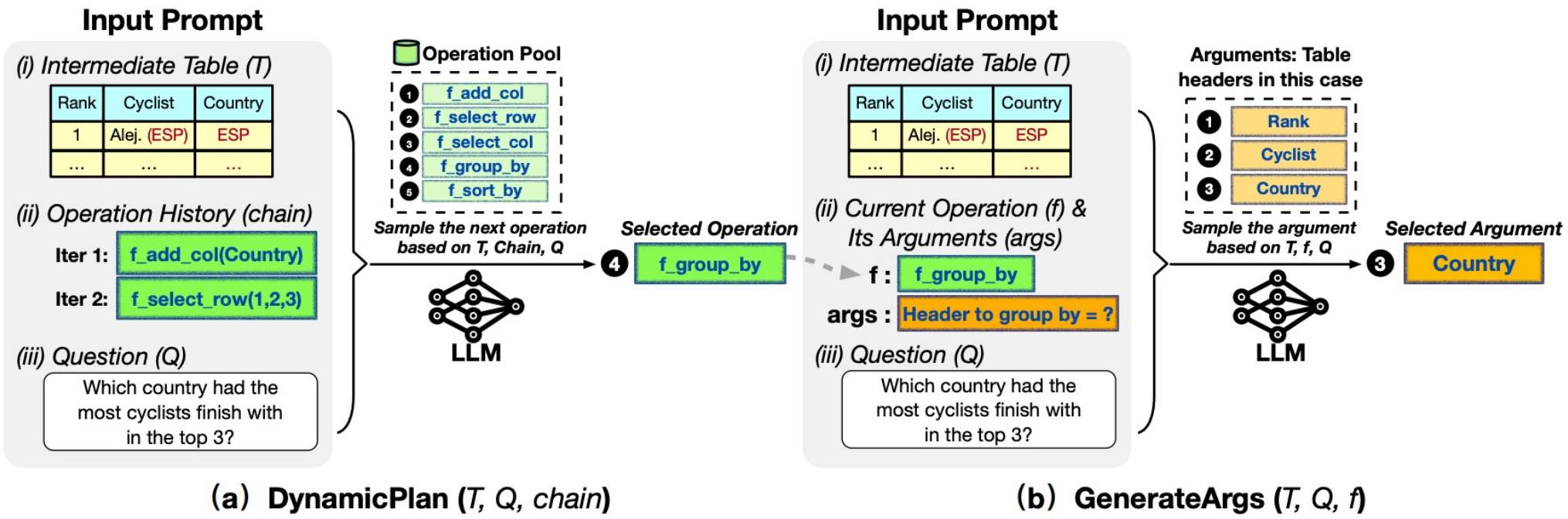
- Given a dataset and a question: Generate a formal query



Chain of Tables:

- Inspired by chain of thoughts
- Let the LLM generate the query step by step
- Generate next operator, generate paramters of operator

Chain of Tables [Wang et. al., 2024]



Our Experience:

Awesome!

- Let us work it out for: ETL approaches, multi-table QA, Large tables etc.

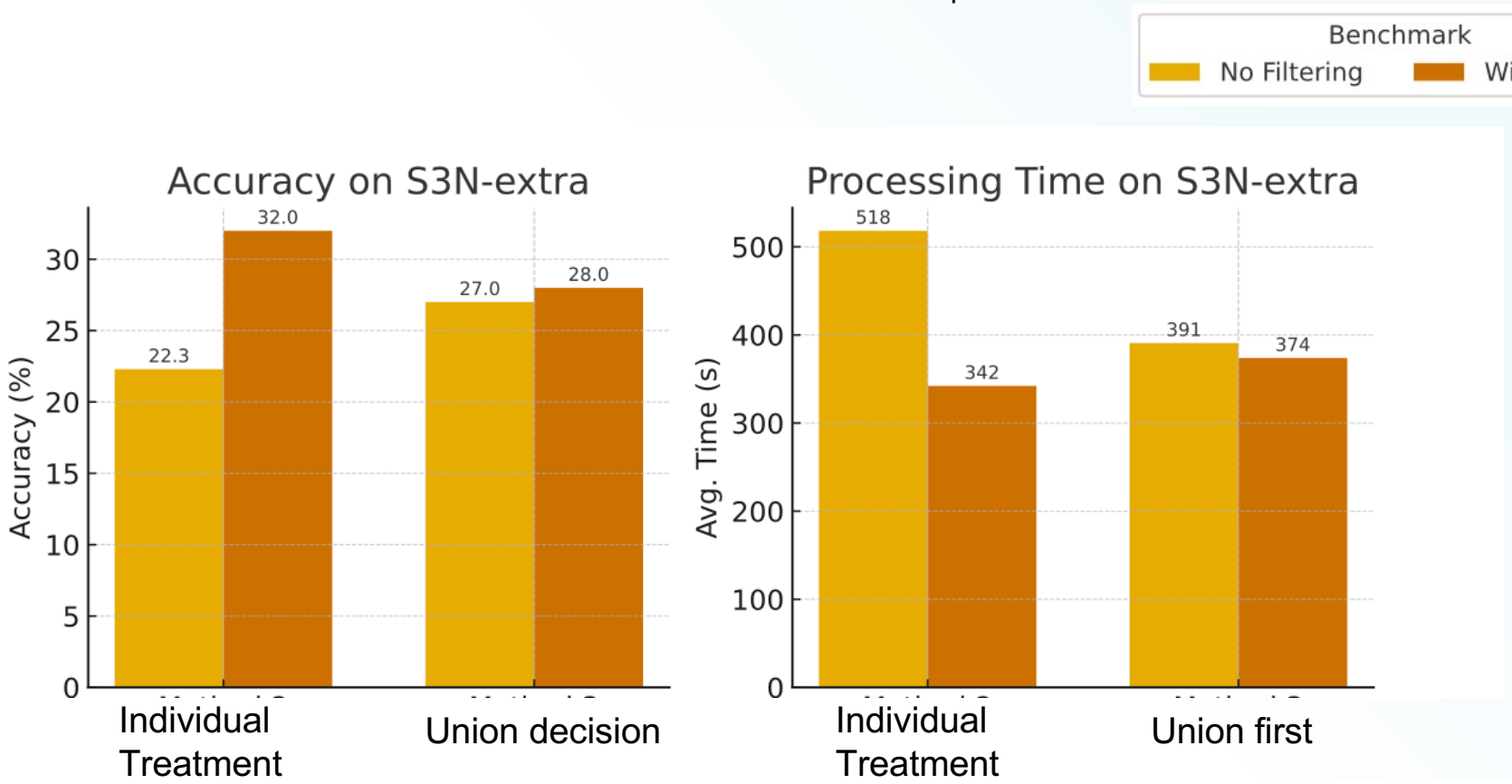
Reality:

- We could not reproduce the results
- Some ad-hoc few-shot paramters and post-processing needed
- The better the model, the more eager it is to answer right away without proposing an operation
- New models have probably seen the entire benchmark

Operation Set	Execution Accuracy (without regex)	Execution Accuracy (with regex)
{Direct Query}	26.78%	18.83%
{Select Rows, Direct Query}	17.27%	8.14%
{Select Columns, Direct Query}	30.56%	16.9%

Multi-Table Scenario
(Decide which tables to use)

S3N-Extra: 3 Relevant tables and 2 irrelevant tables per task



A lot of space to explore

Our original questions still open:

- ETL approaches
- High accuracy for Multi-table QA
- Large tables
- Chain of tables is generally inefficient: 2 prompts per step

Lessons learned so far

You see a new paper solving your problem: **Take a deep breath**. There is no magic.

There are probably **strong assumptions** somewhere

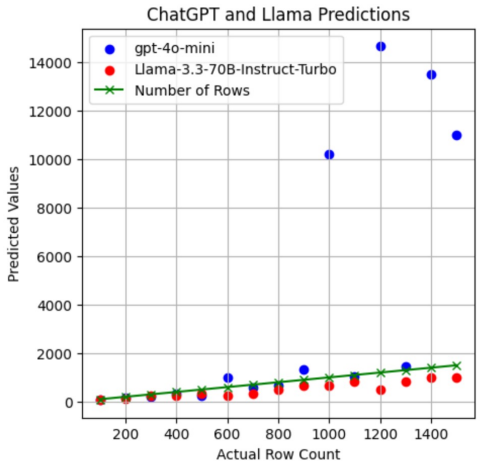
- Focus on methodology instead of black box instrumentation
- Let us understand the capabilities of technologies, such as GenAI, table representation, agentic systems
    - There is **no meaning in competing against a black-box software stack** such as ChatGPT unless you want to show cost-efficiency (Reviewer 2 hates this one)

We are going 2 steps back

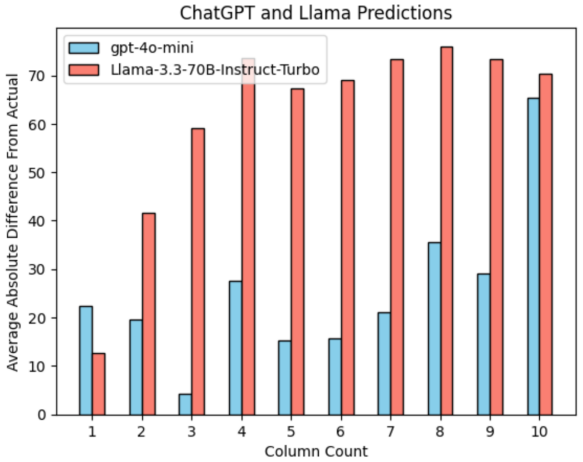
Let us first see how it can solve simple tasks.

- Profiling
- What are the error types it covers?
- How does the accuracy of any task deteriorate with increasing number of rows or columns?

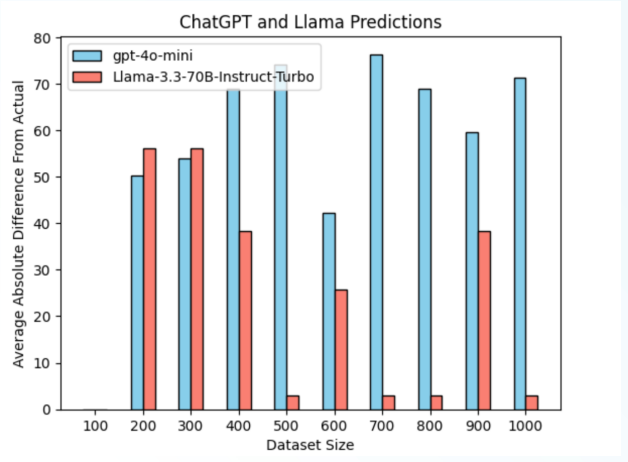
Predict row-count



Count missing values



Count distinct values



Conclusion & Future Directions

Conclusion

- LLMs are still in their infancy regarding tabular data integration
- Useful tool for token-level problems
  - Spelling errors
  - Similarities
- Useful as a human interface
- Leverage the generative nature:
  - Ask for rules, patterns, heuristics → mostly covers head topics, but still useful
  - Ask for augmentation
- TRL needs leaps forward
  - Context of a table should be its task