

GRAM: Generative Retrieval Augmented Matching of Data Schemas in the Context of Data Security

Xuanqing Liu
xuanqing@amazon.com
Amazon Web Services
Seattle, Washington, USA

Yang Song
patsong@amazon.com
Amazon Web Services
Seattle, Washington, USA

Runhui Wang
runhuiw@amazon.com
Amazon Web Services
Seattle, Washington, USA

Luyang Kong
luyankon@amazon.com
Amazon Web Services
Seattle, Washington, USA

SIGKDD '24

Presenter: Joohyung Yun

2024.9.5

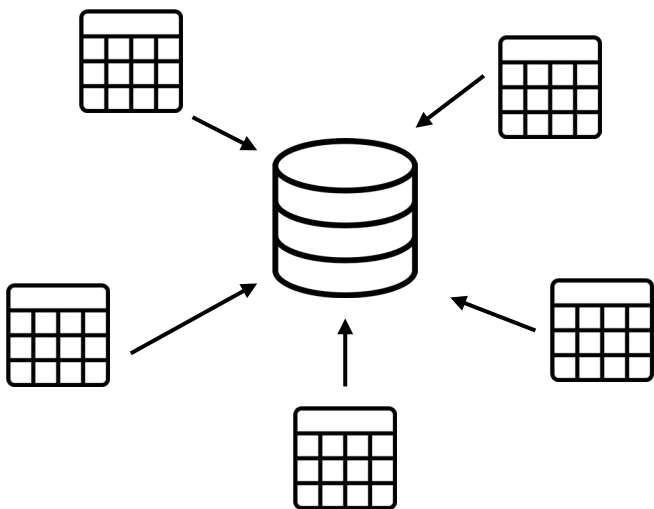
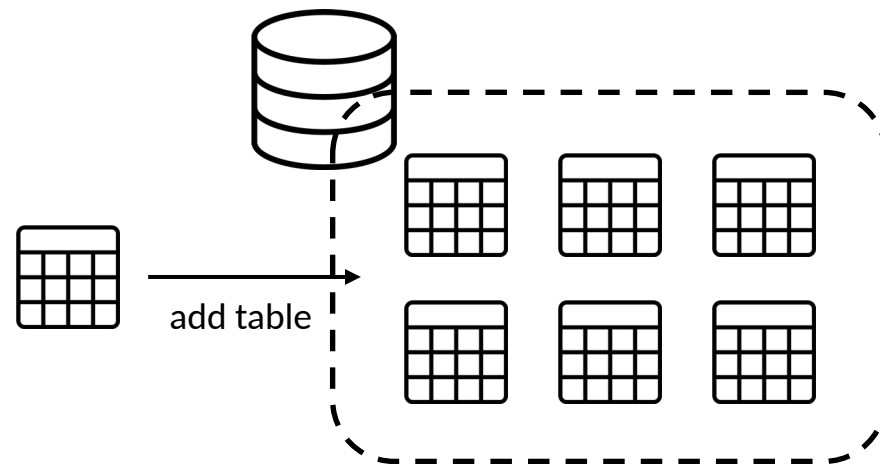
Overview

- Introduction
- Problem Definition
- Existing Works
- Methodology
- Evaluation
- My Thoughts

Introducing Data Ingestion

Data ingestion: a process that encompasses the extraction, loading, and transformation of data.

- **Applications:**
 - Integrating multiple tables to form a database
 - Adding a new table to an existing database



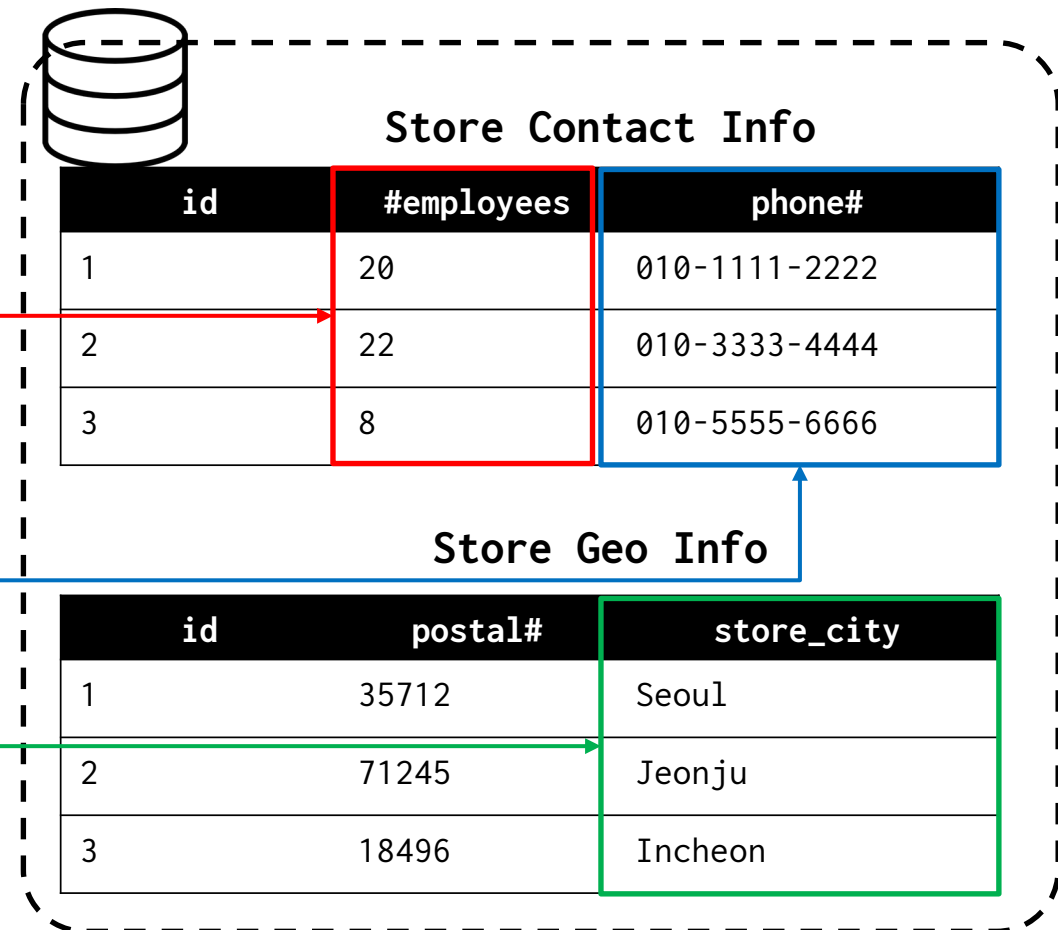
Motivation: Inhomogeneous Schemas

Schemas (or attribute names) may differ among different tables used among different users.

The attributes that share the same semantic, but with different names should be found!

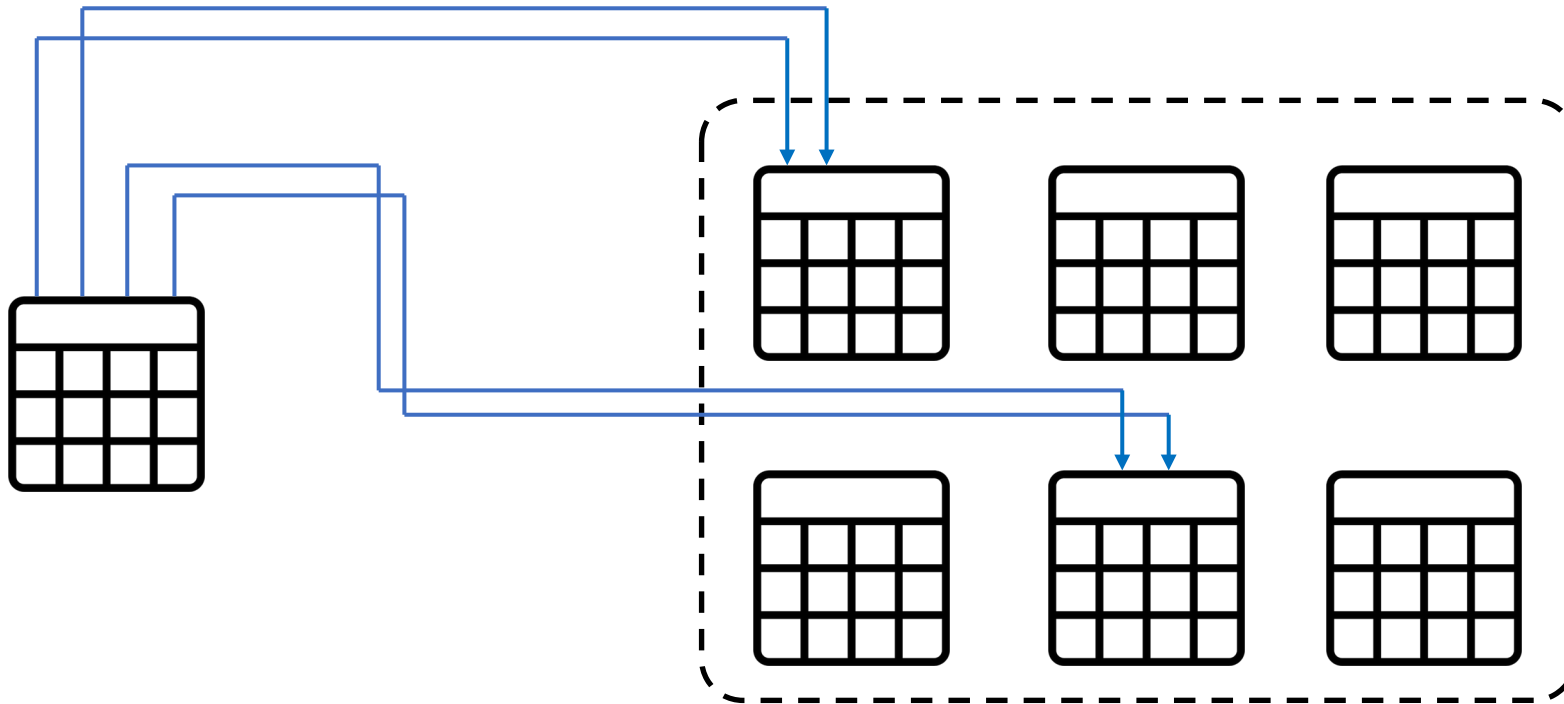
< Table to be added >

num_employees	mailing_address_city	business_phone
10	Pohang	010-1234-5678
7	Gyeongju	010-2345-6789



Description of Schema Matching

Schema matching: The problem of matching attributes of a table to attributes among a set of tables.



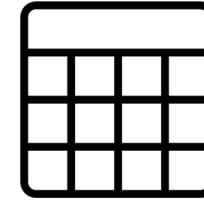
Overview

- Introduction
- **Problem Definition**
- Existing Works
- Methodology
- Evaluation
- My Thoughts

Notations

Schema S : A set of attributes

- Attribute A : 6-tuple of key-value pairs
 - Column name (*required*)
 - Data type
 - Nullable
 - Column meaning
 - Values
 - Length



An icon for
a schema



An icon for
an attribute

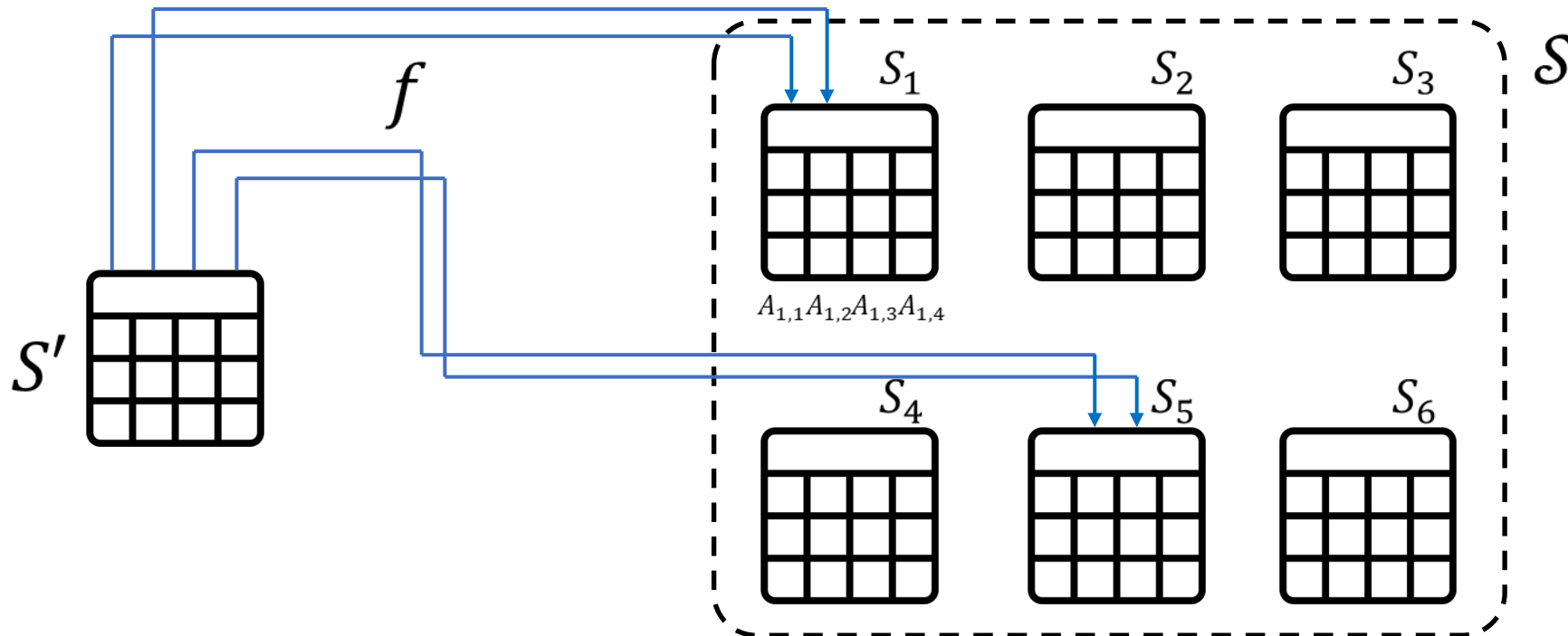
Column name:	<i>order_cancelled</i>
Data type:	<i>boolean</i>
Nullable:	<i>False</i>
Column meaning:	<i>A variable indicating if the order has been cancelled by customer</i>
Values:	[False, False, True, ...]
Length:	112,000,000

An example of an attribute

Schema Matching

Given a source schema S' and a set of target schemas $\mathcal{S} = \{S_1, \dots, S_n\}$, correctly infer a mapping $f: S' \rightarrow \bigcup_{i=1}^n S_i$.

- $S_1 = \{A_{1,1}, A_{1,2}, \dots, A_{1,m}\}$

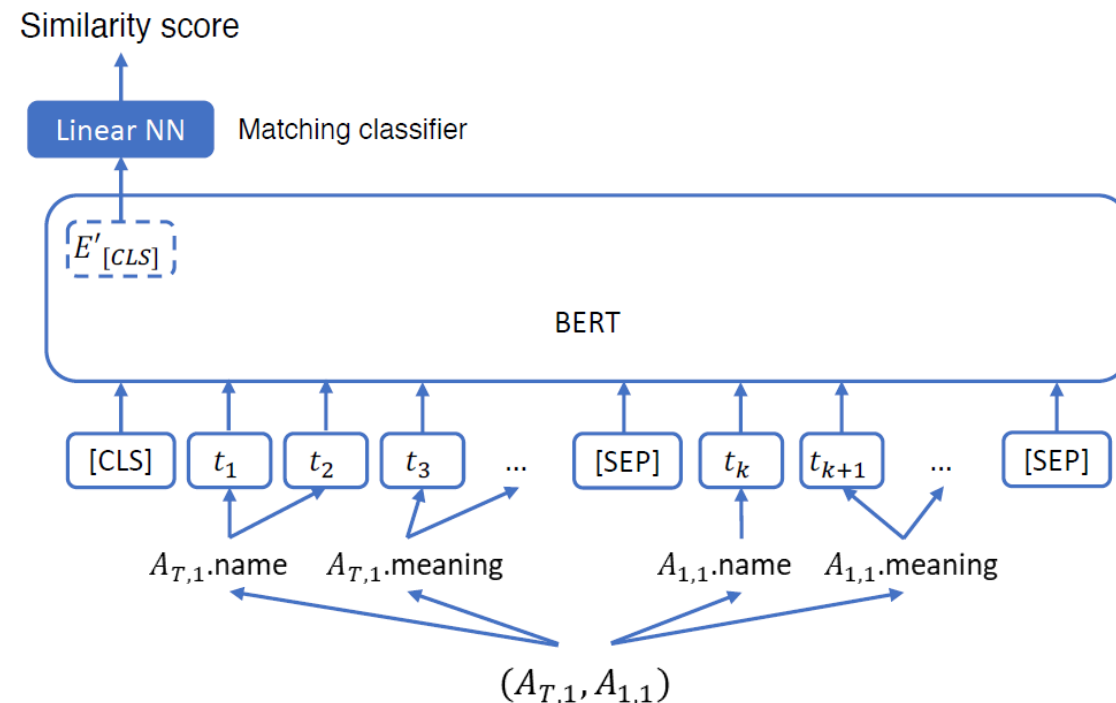


Overview

- Introduction
- Problem Definition
- **Existing Works**
- Methodology
- Evaluation
- My Thoughts

LSM: Prior Schema Matching Work

- Collapses S' and S to a set of attributes
- Uses BERT-based featurizer to calculate similarity between two attributes.
 - Fine-tuned using human-provided labels.



Contributions

- We address the **automated schema matching problem within the context of data privacy**, employing a novel perspective that incorporates zero-shot and few-shot predictions.
- Our solution seamlessly **integrates the recent surge in large language models**. We conduct a comprehensive benchmarking exercise across various open-source and proprietary LLMs to assess their performance.
- Introducing a **dynamic prompt selection** method based on input characteristics, our approach not only **enhances inference speed** but also **augments the in-context learning accuracy of LLMs**.

Contributions (Cont.)

- Beyond the conventional scope of schema matching, our solution **incorporates object type detection and unique key detection**. These additional components transform the standalone schema matching module into a more feature-complete data-table ingestion service.
- We **rigorously benchmark the accuracy of our methodology against relevant approaches** using both public and production-quality, synthetic datasets. Particularly noteworthy is the utilization of datasets designed to mirror realistic workloads in various industrial applications.

Overview

- Introduction
- Problem Definition
- Existing Works
- **Methodology**
- Evaluation
- My Thoughts

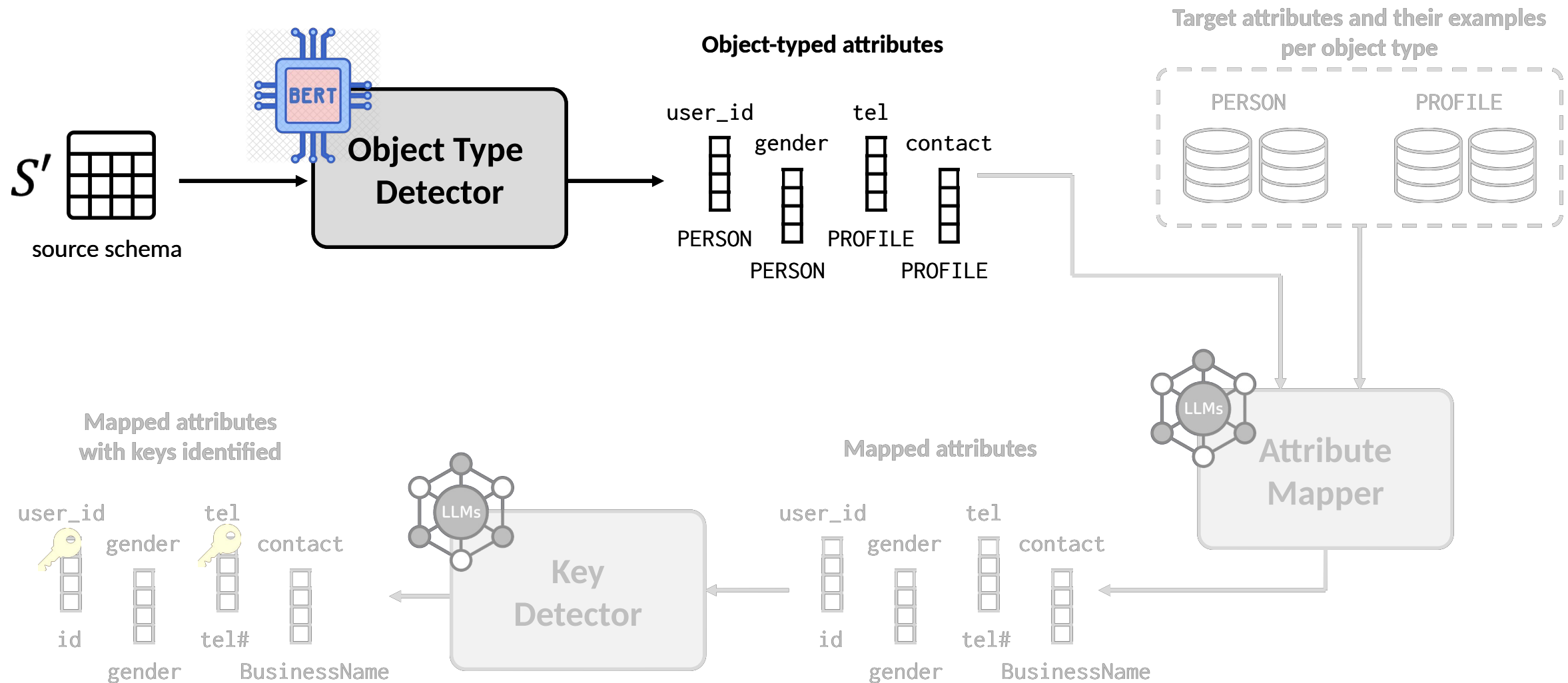
Global Example

S'

user_id	gender	tel	contact
JE92K	M	193019120	Amazon.com Inc.
0F29S	F	192928919	null

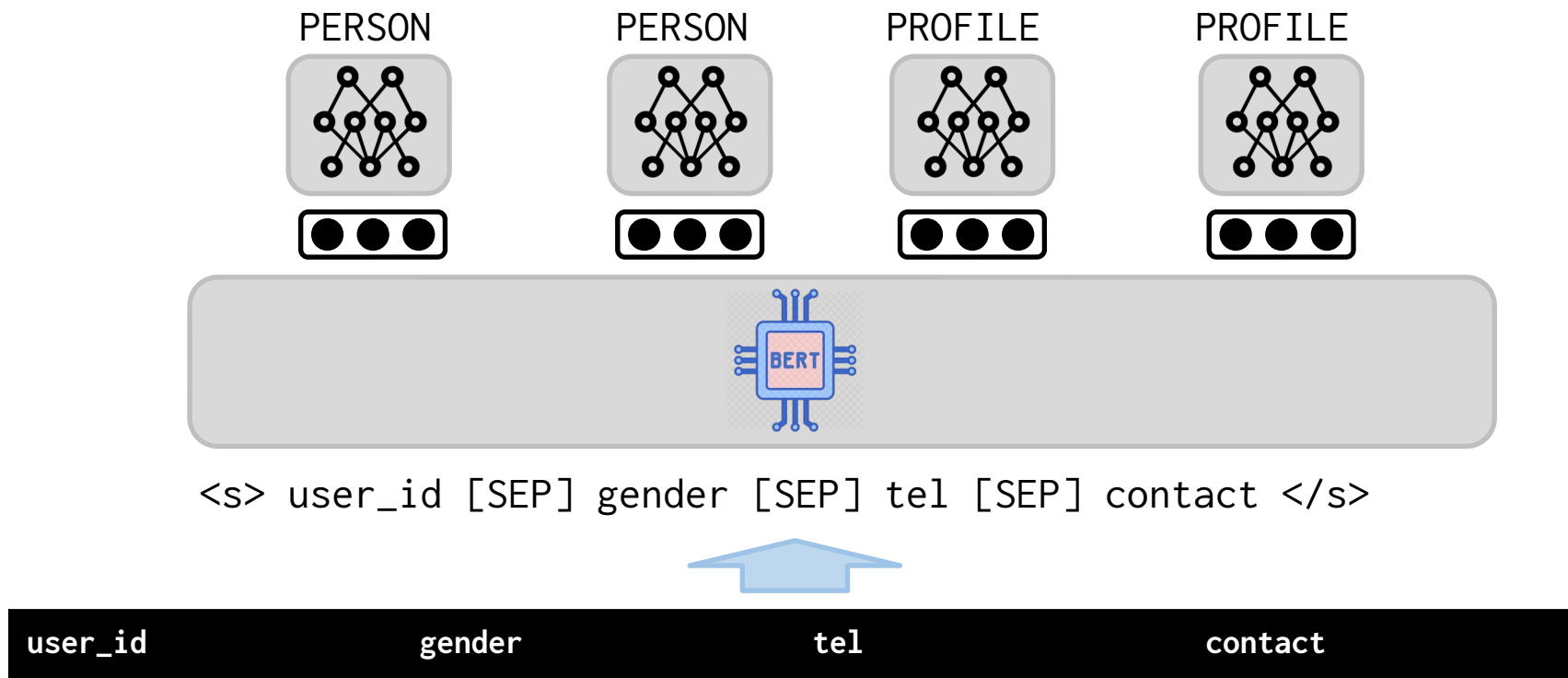
source schema

GRAM Overview

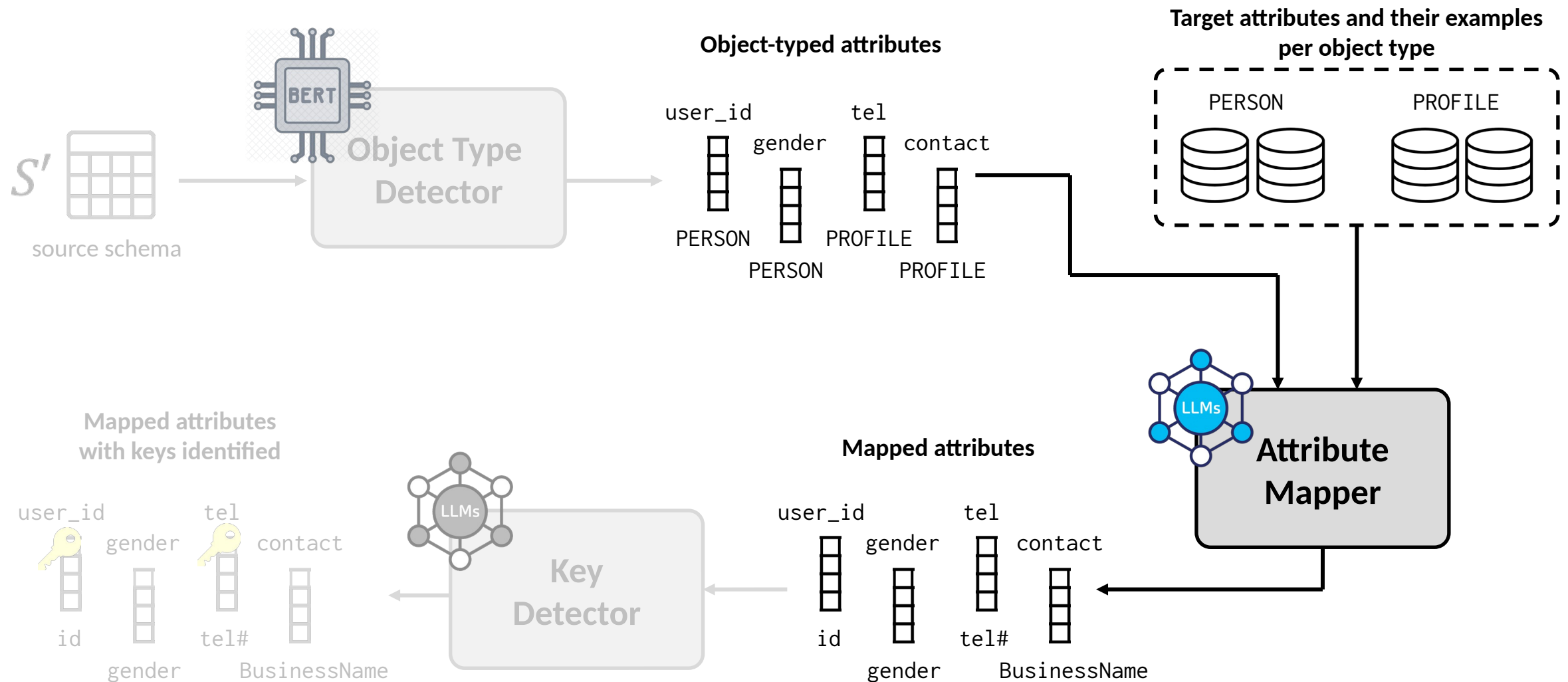


Object Type Detector

- Narrow down the candidate attributes using grouping criteria.
 - Classify each column name into predefined group labels using BERT-based multi-class classifier.



GRAM Overview (Revisited)



Reducing the Problem to Attribute Mapping

- How to treat each attribute

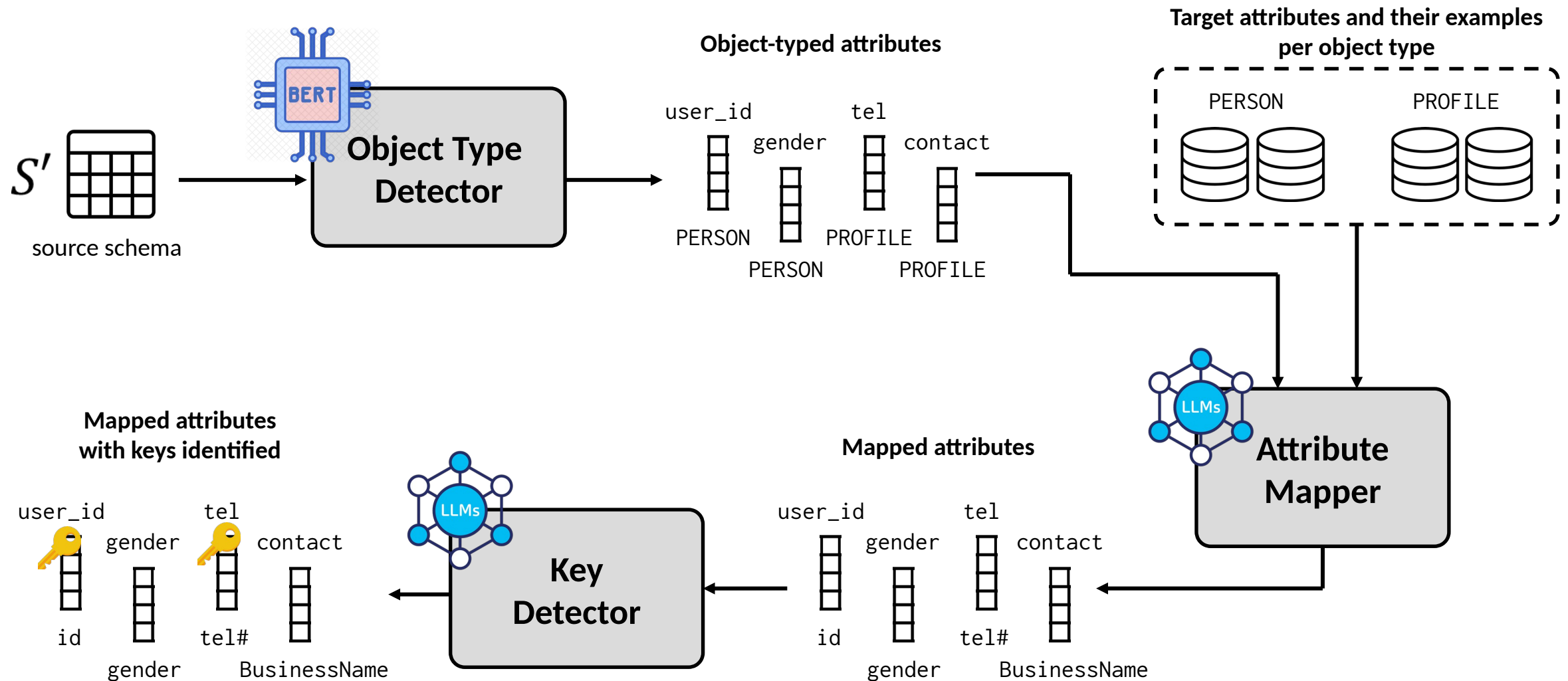
- Prompt example + Problems

- Lower level architecture

- NER-based destination filter

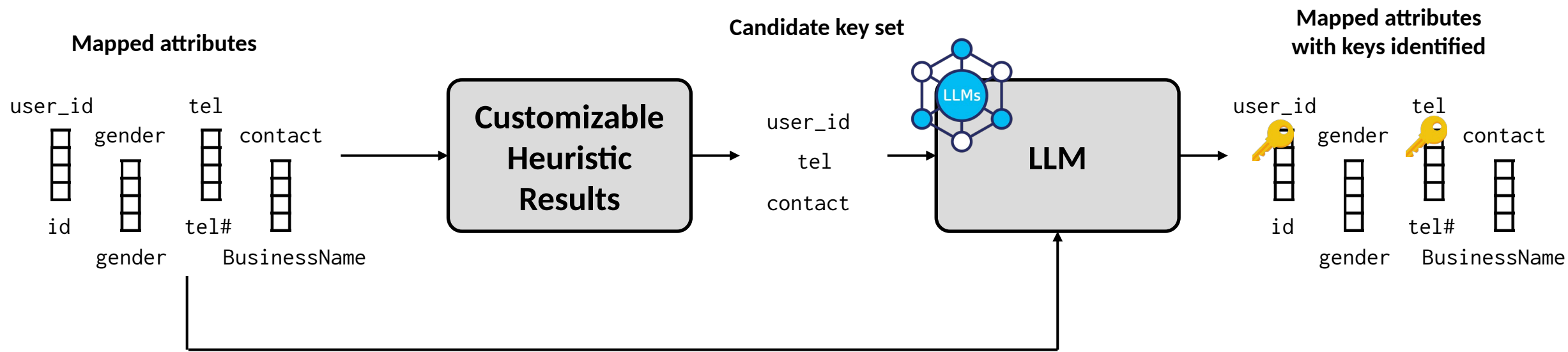
- Double-RAG filter

GRAM Overview (Revisited)



Key Detector

- Postprocessor for the attribute mapper
- **Role:** Find keys among source attributes.



Overview

- Introduction
- Problem Definition
- Existing Works
- Methodology
- **Evaluation**
- My Thoughts

Objectives of Evaluation

- **Accuracy comparison:** how does this retrieval-augmented LLM solution compare with traditional solutions in terms of accuracy?
- **Ablation study:** What benefits are observed in throughput when incorporating the prompt compression techniques?
- **Parameter sensitivity:** How does the number of k -shot examples influence end-to-end accuracy?
- **LLM-size sensitivity:** What is the most practical choice among LLM backbones of different sizes?

Experimental Settings

- **Competitors:** LSM, LSD, CUPID, COMA 3.0
- **Datasets**
 - Personal Contacts: 1400 columns
 - Sales: 400 columns
 - Products: 200 columns
 - Issue Tickets: 330 columns
- **Used models**
 - LLM: FLAN-T5 model
- **Hardware settings**
 - 4x Nvidia A10 GPUs, 24 CPU cores, 192GB memory

Ablation Study

Algorithms	Mean accuracy (%) in domain				
	Person	Sales	Products	Tickets	Avg.
LSD [10]	73.0	63.6	61.8	74.7	68.3
CUPID [21]	52.2	50.6	39.8	62.7	51.3
COMA 3.0 [1]	56.6	48.7	69.0	50.6	56.2
LSM [37]	81.0	78.5	70.2	71.4	75.3
GRAM (ours)	91.9	80.3	92.3	90.3	88.7

Table 1: Comparing the accuracy numbers across different domains among traditional algorithms, deep neural nets based algorithms, and our LLM based algorithm.

- GRAM outperforms all existing algorithms in terms of accuracy, with 17% increase on average compared to SOTA.
 - Embedding-based cosine similarity complements lexical similarity metrics.

Ablation Study

Settings	Mean accuracy (%) in domain				
	Person	Sales	Products	Tickets	Avg.
No filter	83.0	78.7	81.1	90.5	83.3
+NER	92.7	86.6	88.1	85.4	88.2
+Double-RAG	89.4	74.9	89.6	91.6	86.4
+Both filters	91.9	80.3	92.3	90.3	88.7

Table 2: Comparing the testing accuracy with and without filters. Filters do not change model accuracy in a consistent direction.

- Imposing one or both of the filters show increase in accuracy.
 - The authors explain this as removal of hard negatives given to LLM.
- Imposing both filters show about 80% increase in matching throughput.

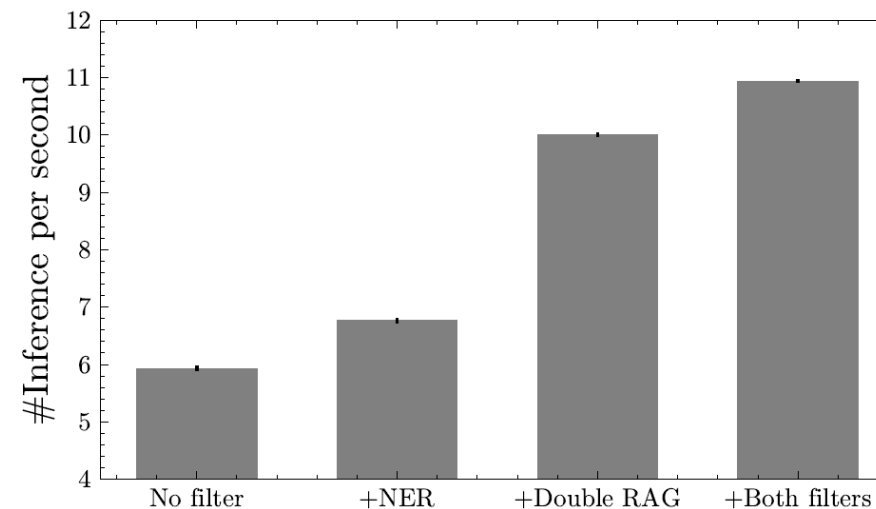


Figure 5: Inference speedup due to NER and double-RAG filters. With double-RAG filter, we keep $k_{\text{opt}} = 4$ options and $k_{\text{ex}} = 1$ examples for each of the 4 options. Error bars are provided but barely visible.

Parameter Sensitivity

k : the number of examples given per each option

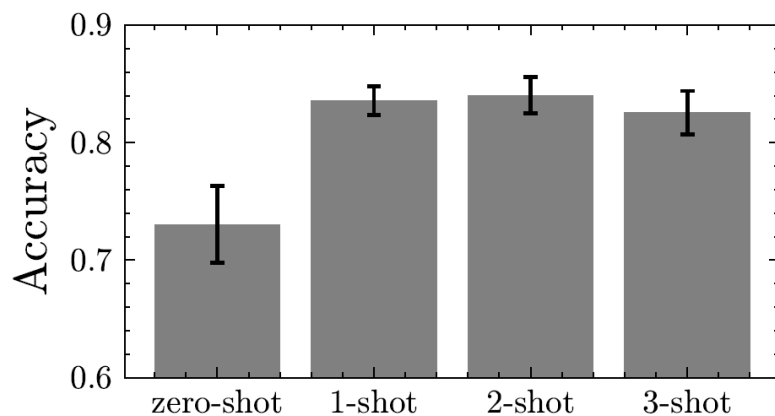


Figure 7: Comparing matching accuracy under varying k -shot examples in the prompt.

- Rapid increase in accuracy from zero-shot (73.05%) to 1-shot (83.58%).
- Error drops from 3-shot in-context learning.
 - *The authors did not explain this phenomenon.*

LLM-Size Sensitivity

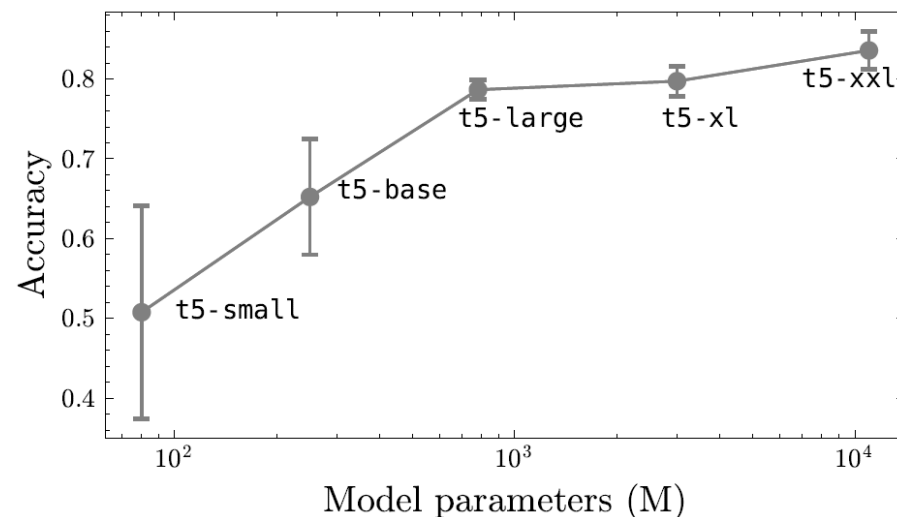


Figure 6: Comparing the matching accuracy among different sizes of instruction finetuned models, accuracy is averaged across domains and datasets therein.

- Consistent increase in accuracy as the number of model parameter increases.

My Thoughts about This Paper

- May adopt the idea of prompt shortening for efficiency in our research
- Insufficient reasoning for each decision
- Insufficient analysis on experiment results
 - Reasons for some tendencies (e.g., decrease in k-shot example) are not given.
- Poor writing
 - Methodology is presented in a bottom-up manner.
 - Examples do not align the text.