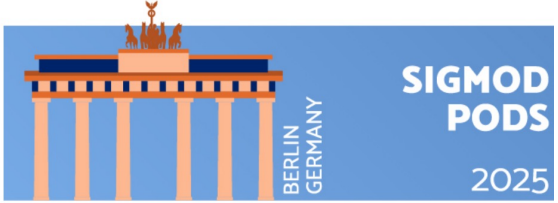


Data+AI: LLM4Data and Data4LLM

Guoliang Li, Jiayi Wang, Chenyang Zhang

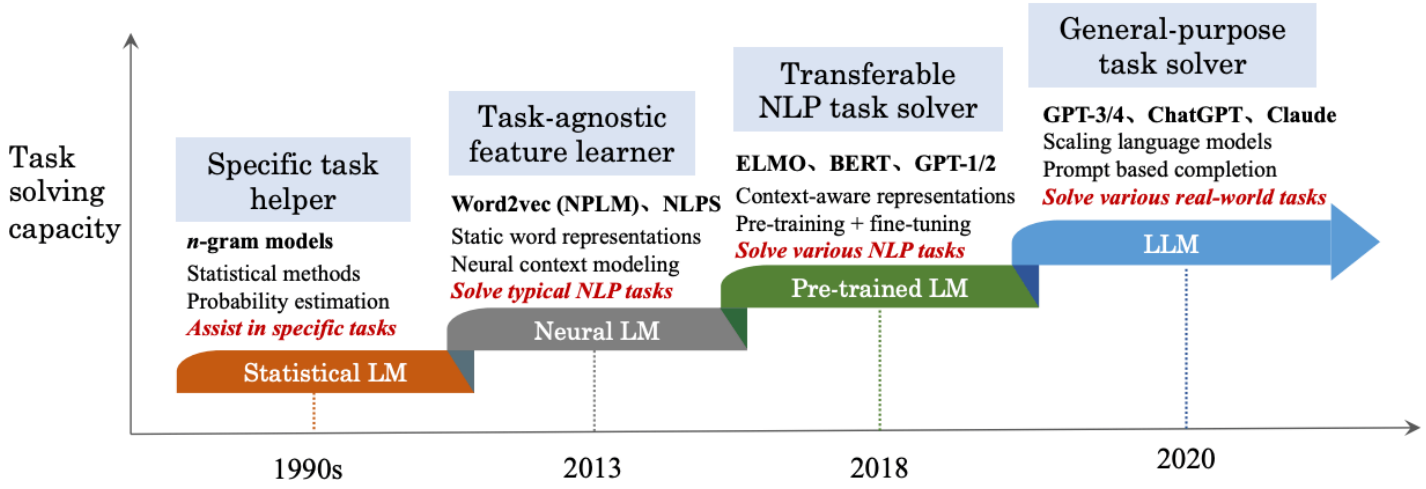
Tsinghua University

Jiannan Wang

Simon Fraser University

LLMs Are Revolutionizing Data/Database Systems

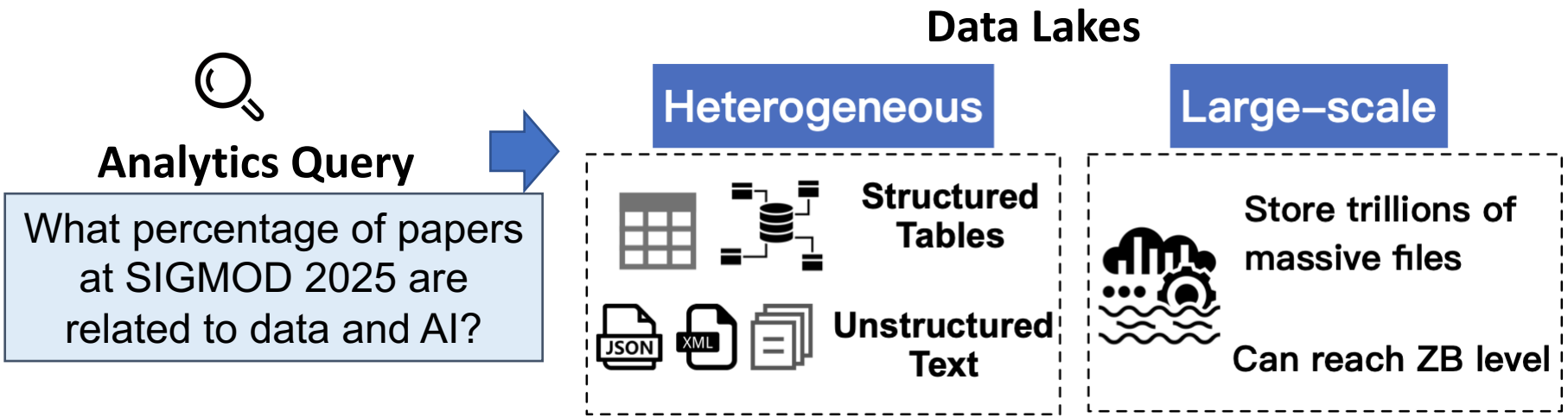
- **LLMs are revolutionizing data management systems due to their:**
  - Text → Semantics: Semantic understanding capabilities
  - Retrieval → Reasoning: Reasoning and planning ability
  - Vertical domains → Multiple domains: Adaptability for supporting various tasks
  - Closed World → Open World: Generalization capabilities



https://klu.ai/glossary/large-language-model

LLM4Data: LLM Capabilities – Semantic Processing

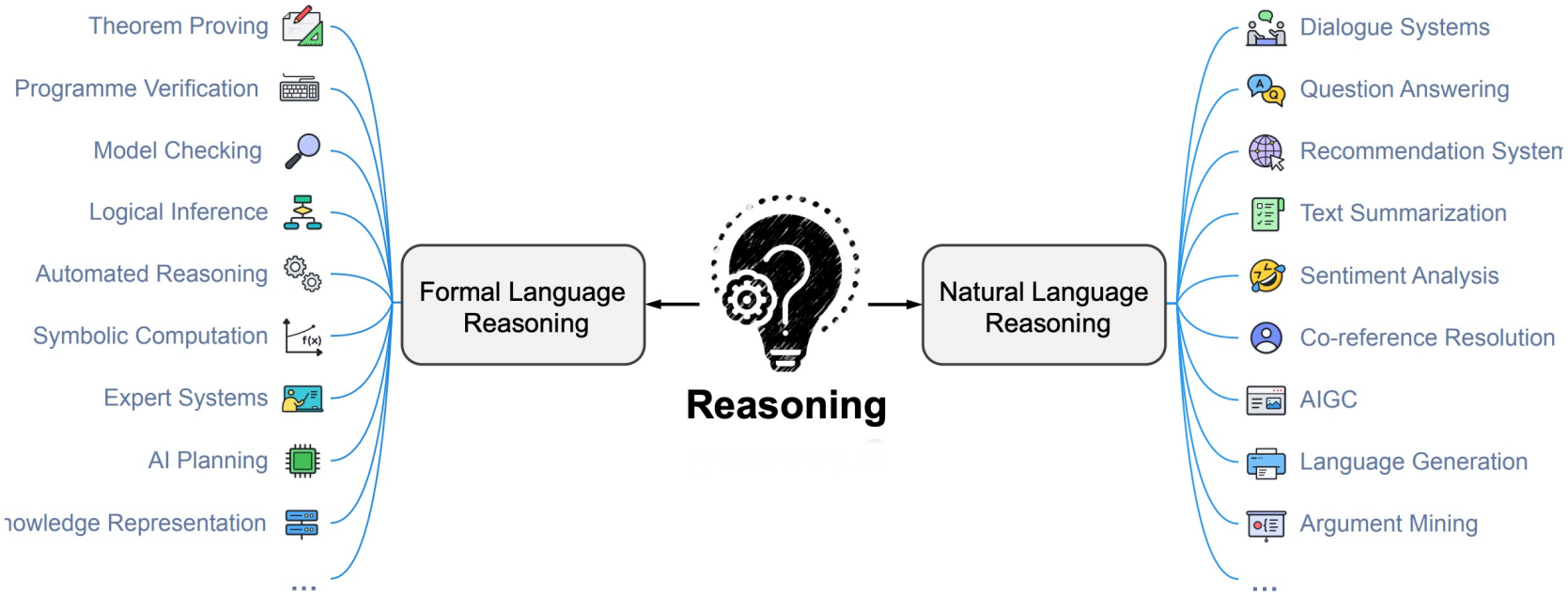
- ❑ Traditional data management can only get results exactly in database
- ❑ However, semantic processing is crucial to discern nuances, context and subtleties that are typically challenging for traditional ML models



J. Wang, G. Li, and J. Feng. idatalake: An llm-powered analytics system on data lakes. Data Engineering, 2025

LLM4Data: LLM Capabilities – Reasoning (Inference)

- ❑ Conduct multi-step reasoning
- ❑ Perform better on logical, mathematical or programmatic tasks



Sun J, Zheng C, Xie E, et al. A survey of reasoning with foundation models[J]. arXiv preprint arXiv:2312.11562, 2023.

LLM4Data: LLM Capabilities – Adaptability (Knowledge)

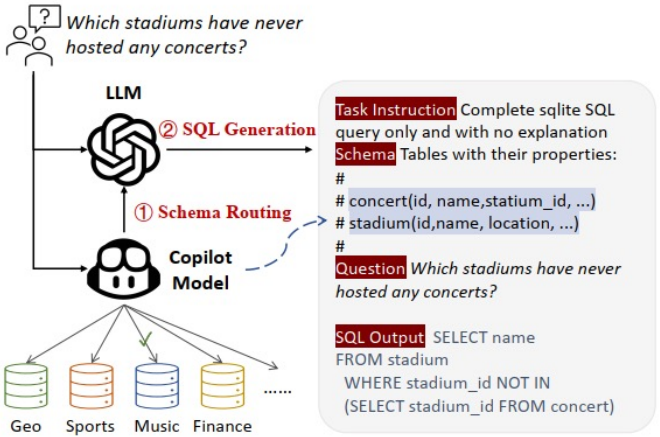
- ❑ Extensive knowledge coverage due to diverse datasets
- ❑ Enable LLMs to understand and process various queries and tasks

	Wikipedia	Books	Journals	Reddit links	CC	Other	Total
GPT-1		*4.6*					4.6
GPT-2				**40**			40
GPT-3	*11.4*	*21*	*101*	*50*	**570**		753
The Pile v1	**6**	**118**	**244**	**63**	**227**	**167**	825
Megatron-11B	**11.4**	*4.6*		**38**	**107**		161
MT-NLG	*6.4*	*118*	*77*	*63*	*983*	*127*	1374
Gopher	**12.5**	**2100**	*164.4*		*3450*	*4823*	10550

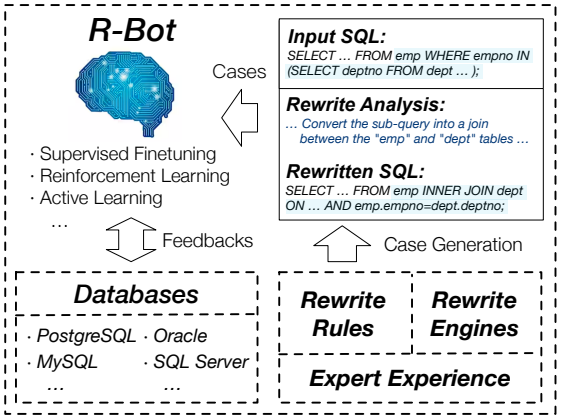
https://hub.baai.ac.cn/view/24150

LLM4Data: LLM Capabilities – Understanding & Generation

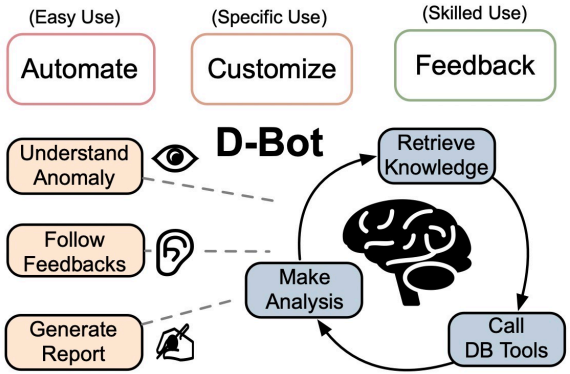
- ❑ **Beyond comprehension, LLMs are capable of generation**
- ❑ **LLMs can create human-like text in response to prompts**
  - Can be utilized in data management for generating reports, automating data documentation, and even crafting queries in natural language



Text2SQL



Query Rewrite



Diagnosis

Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models. CoRR abs/2309.01219 (2023)
https://promptdrive.ai/llm-limitations/

LLM4Data: LLM Capabilities – In-context Learning

❑ **High-Quality Prompt can instruct LLMs to optimize DB tasks without training**

➢ **Zero-shot Prompting**

- Input LLM with a **task description**, without training over labeled data

- **Instruction Prompting**
  - Input LLM with **explicit instructions** on approaching the task, e.g., detailing the format, tone, or type of output response

➢ **Few-shot Prompting**

- Provide LLM with a few **examples** of the task within the prompt to guide the model on how to generate responses

Prompt of Query Rewrite

Task Description
Write an equivalent SQL query that can be executed on a Postgres database with decreased latency.

Instruction
1. Ensure output query is semantical-equivalent to the input query …

Example Input
```
select ... from t1 where t1.a=(select avg(a) from t3 where t1.b=t3.b);
```
Example Output
```
select … from t1 inner join (select avg(a) avg,t3.b from t3 group
by t3.b) as t3 on (t1.a=avg and t1.b=t3.b);
```

Input
```
select t1.* from t1 where t1.col1>(
   select max(t2.col2) from t2 where t2.col1 in (
      select t1.col1 from t1 where t1.col1=t2.col1));
```

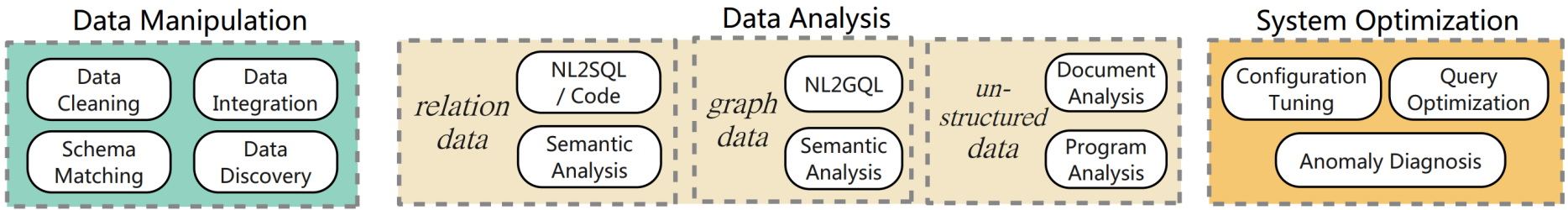
Output
```
select t1.* from t1 inner join (
   select max(t2.col2) max, t2.col1 from t2
      group by t2.col1) as t2 on (
         t1.col1=t2.col1)
where t1.col1>max;
```

Xuanhe Zhou, Zhaoyan Sun, Guoliang Li. DB-GPT: Large Language Model Meets Database. Data Science and Engineering 2023.

LLM4Data: Motivation and Opportunities

❑ Opportunities of LLM for data management

- Automatic planning for data preparation
  - Discovery, cleaning, integration, mixing, standardization
- Semantic data analytics of unstructured data, structured data, data lakes.
  - Natural language based query optimizations
  - Data interpretation and insights
- Data/Database System optimization
  - Tuning, Diagnosis, Optimization



Zhou X, et al. A Survey of LLM x DATA. arXiv, 2025

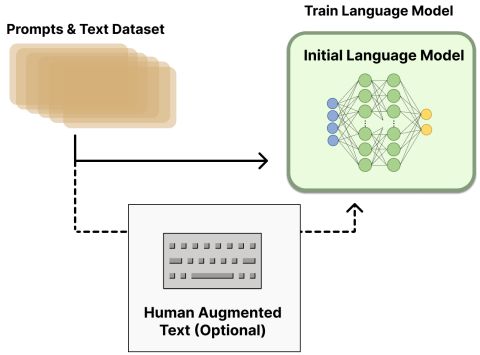
LLM4Data: Challenges and Solutions

Inconsistency	Give conflicting outputs for very similar prompts Task decomposition; Prompt for multiple times and Vote; Self-Reflection …
Hallucination	Generate text that seems realistic and plausible but is actually inaccurate RAG, Write instructive prompts to ask for source/evidence or call tools …
Lack of long-term memory	Cannot automatically retain information from previous chats or update in time Cache and reuse historical messages …
Limited reasoning	Struggle with tasks requiring complex reasoning, multi step problem-solving, … Task decomposition; Provide reasoning process examples; Prompt engineering …
Outdated information	The knowledge LLM used can be out-of-date, because the new knowledge is learned in batch for traditional model finetuning RAG …
Low parameter efficient	Billions of parameters to update → LoRA; RAG …
Resource constraints	Have memory limits on how much text they can process at once Chunking; Embedding; Prompt Compression; RAG + Vector Databases …

Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models. CoRR abs/2309.01219 (2023)
https://promptdrive.ai/llm-limitations/

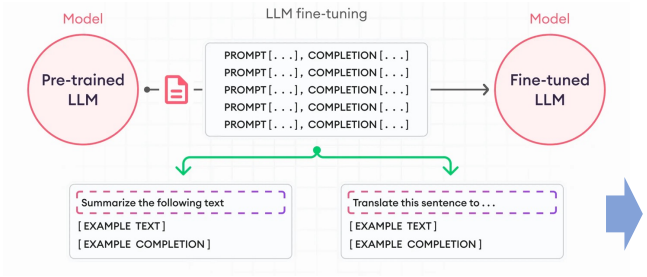
Data4LLM: Different Stages of LLM

1. (Incremental) Pretraining



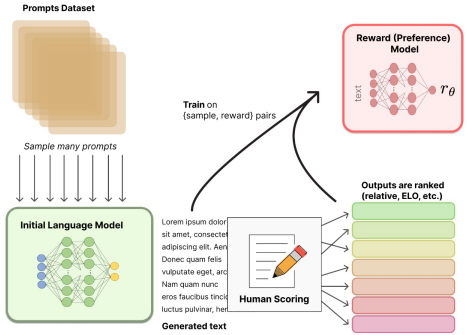
- *Common Knowledge Acquisition*
- *Understanding Diverse Texts*

2. (SFT/RLHF) Finetuning



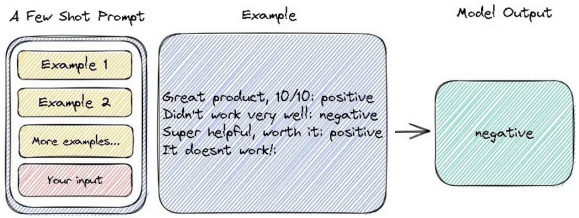
- *Instruction Following*
- *Task Adaption like Traslation/Q&A*
- *Align with human preferences*

3. (RL) Post-training



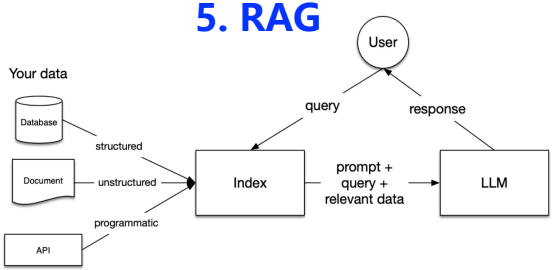
- *Slow thinking*
- *Robustness Enhancement*

4. Prompting



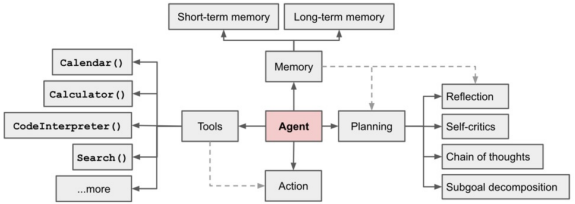
- *Context Comprehension*
- *Learn from demo examples*

5. RAG



- *External Knowledge Integration*
- *Contextual Relevance / QA Accuracy*

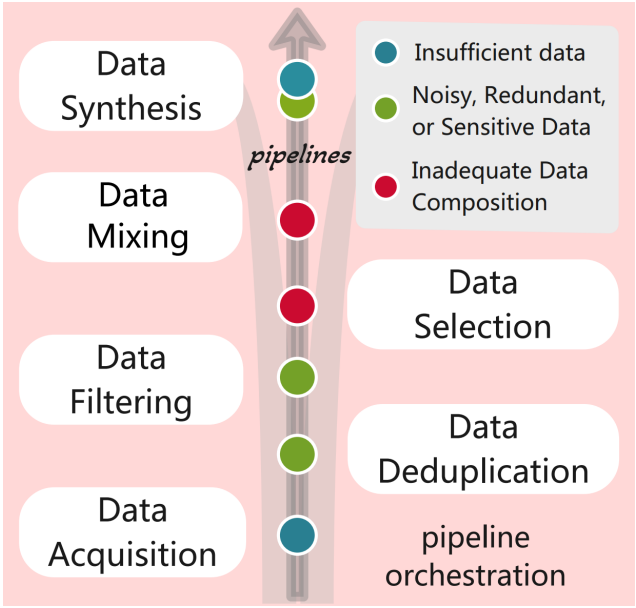
6. Agent



- *LLM system equipped with reasoning, tools, and memory*

Data4LLM: Data Management Can Benefit LLMs

- **The LLM life-cycle includes pretraining, fine-tuning (SFT and RLHF), prompting, RAG, Agent**
- **Effective data management is fundamental to the scalable development and deployment of LLMs**
  - Data Preparation
    - Data Discovery
    - Data Selection
    - Data Cleaning
    - Data Augmentation
    - Data Labeling
    - Data Synthesis
    - Data Processing
    - Data Optimization
    - Data Storage
  - LLM Training
  - LLM Serving (Inference)

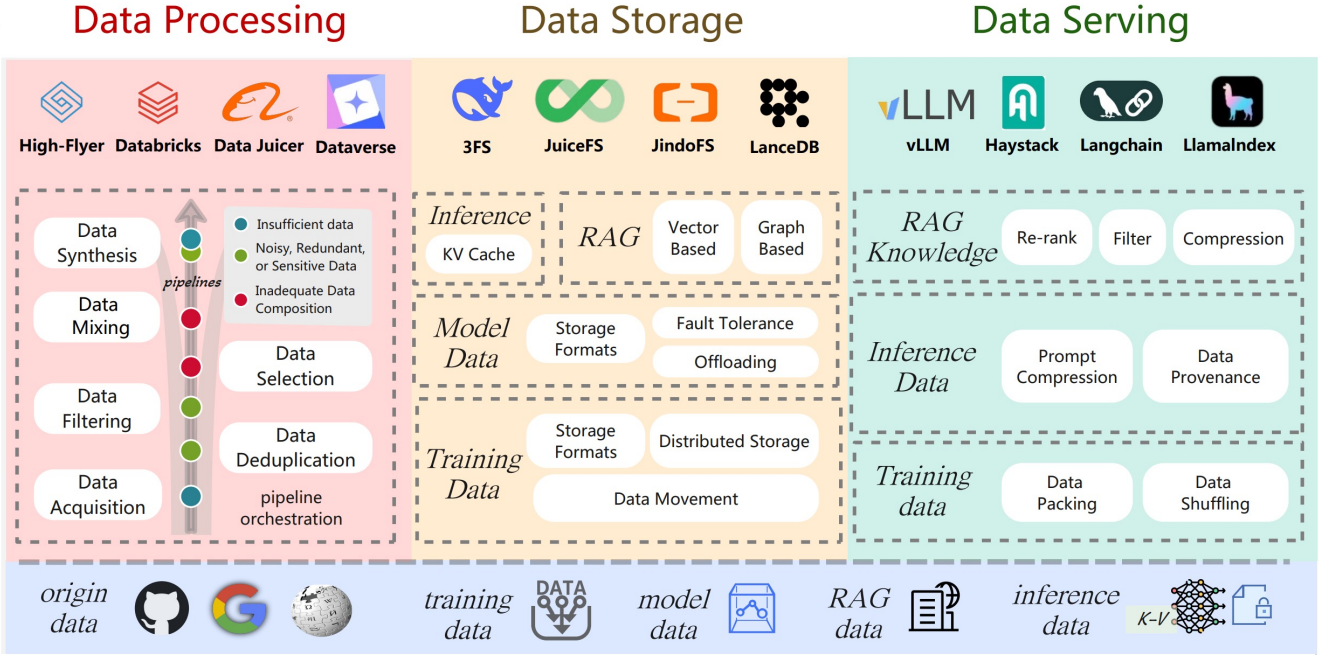


https://klu.ai/glossary/large-language-model

Data4LLM: Motivation and Opportunities

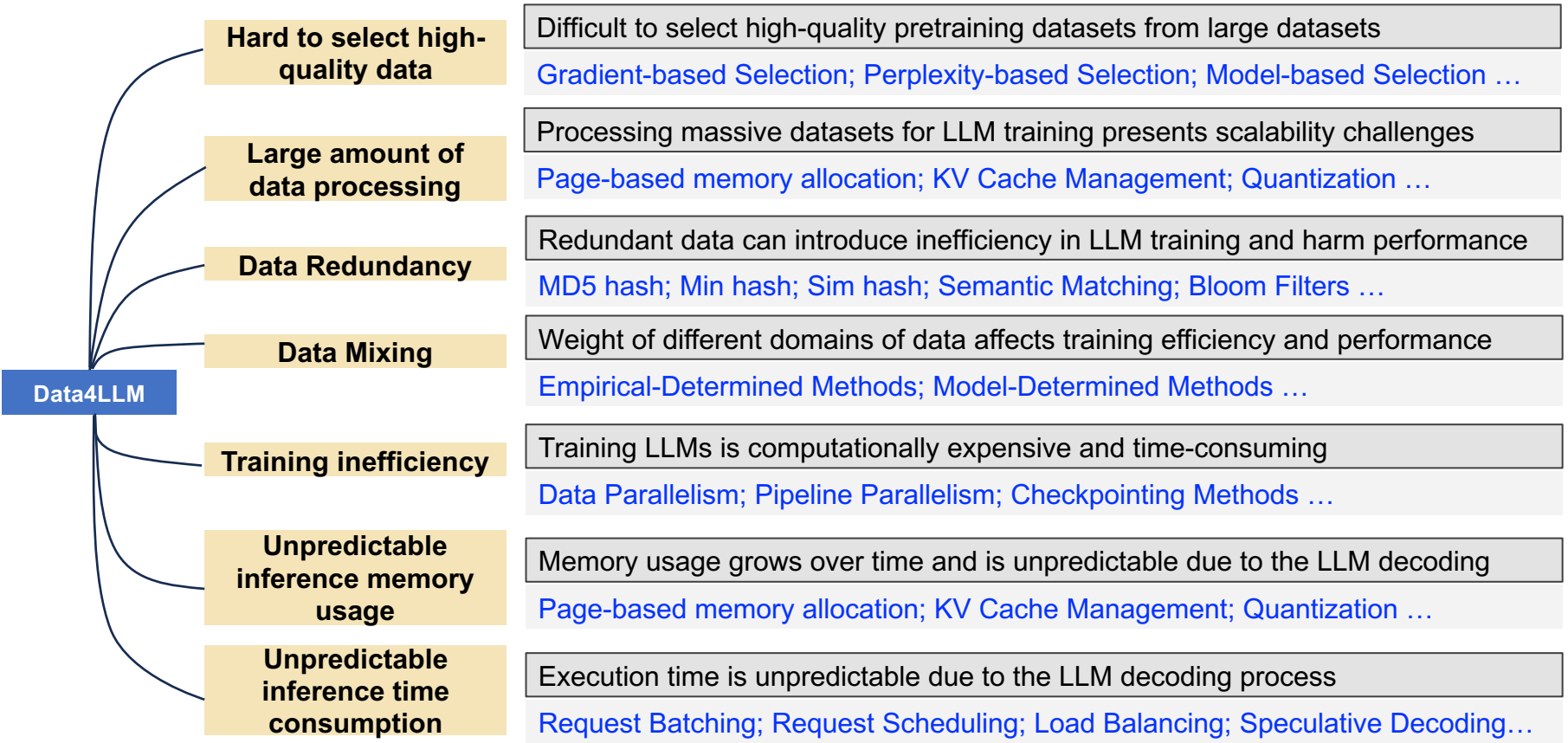
❑ Opportunities of Data4LLM

- Improved Training Efficiency and Cost
- Improved Inference Efficiency



Zhou X, et al. A Survey of LLM x DATA. arXiv, 2025

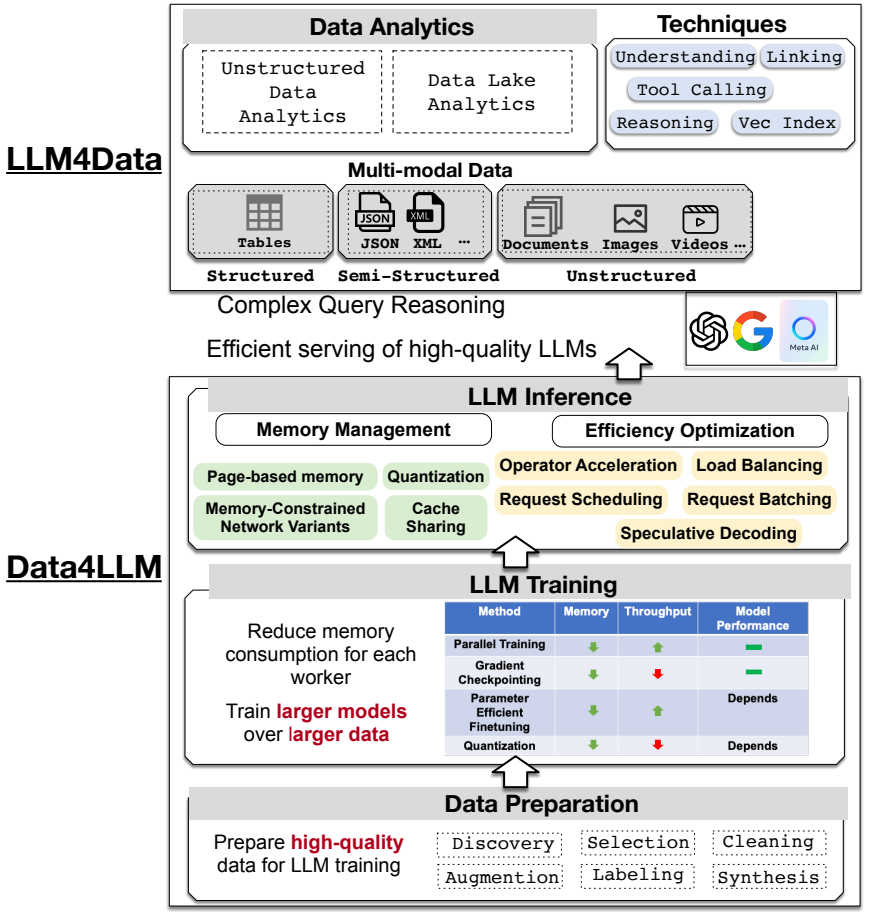
Data4LLM: Challenges and Solutions



Zhou X, et al. A Survey of LLM x DATA. arXiv, 2025

Outline of LLMxData

- ❑ **LLM4Data Techniques**
  - LLM Prompting
  - RAG & Vector DB
  - Data Agents
    - Unstructured Data Analytics
    - SQL + Semantics
    - Data Lake Analytics
- ❑ **Data4LLM Techniques**
  - Data Preparation
  - LLM Inference
  - LLM Training
- ❑ **Open Challenges**



Outline of LLMxData

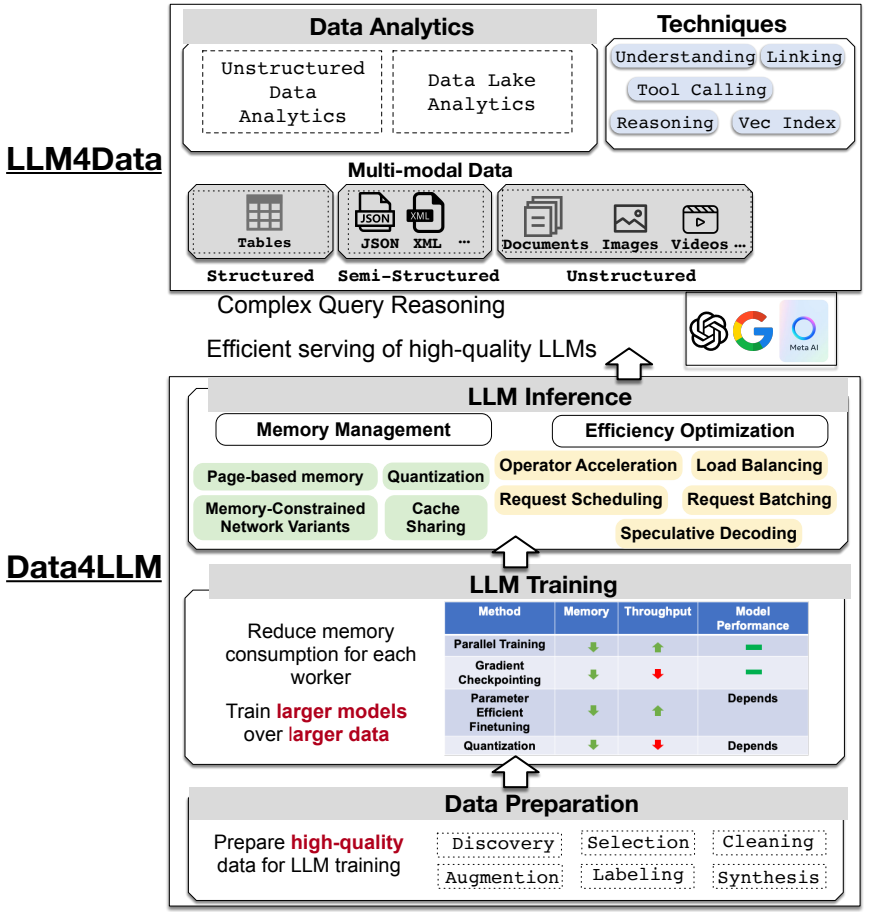
❑ LLM4Data Techniques

- LLM Prompting
- RAG & Vector DB
- Data Agents
  - Unstructured Data Analytics
  - SQL + Semantics
  - Data Lake Analytics

❑ Data4LLM Techniques

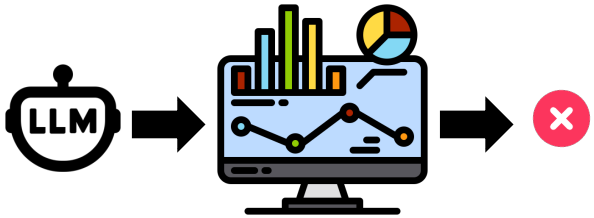
- Data Preparation
- LLM Inference
- LLM Training

❑ Open Challenges



Challenges of LLM4Data

❑ Low Accuracy

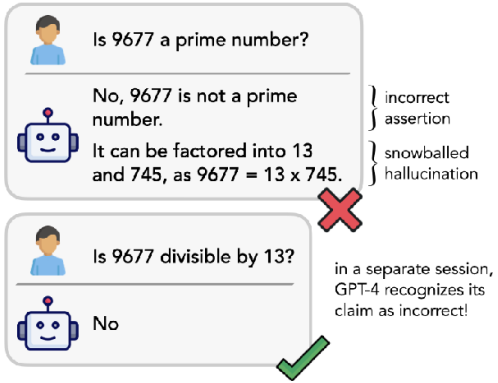


- **Hard for complex tasks**

❑ High Cost

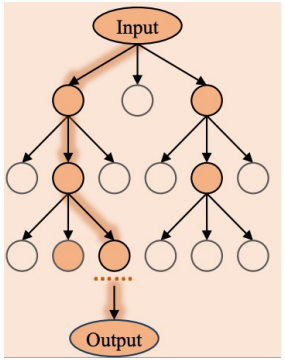
- **Large number of LLM invocations**

❑ Hallucination



- **LLMs may output factual errors**

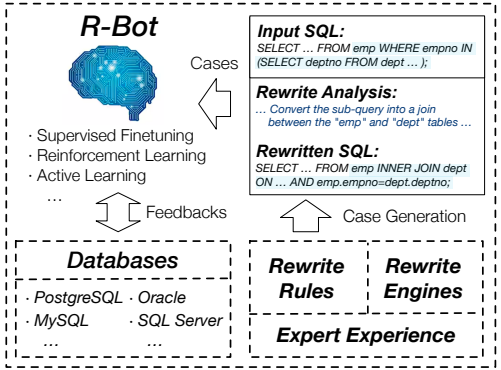
❑ Limited Reasoning



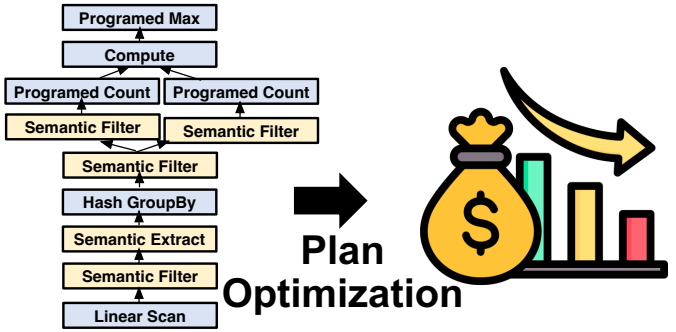
- **Require multi-step reasoning**

Principles of LLM4Data

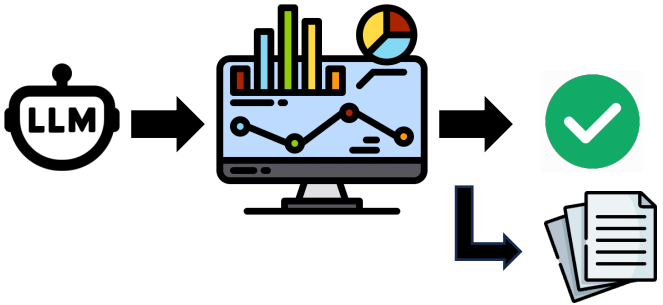
❑ Involving Domain Knowledge



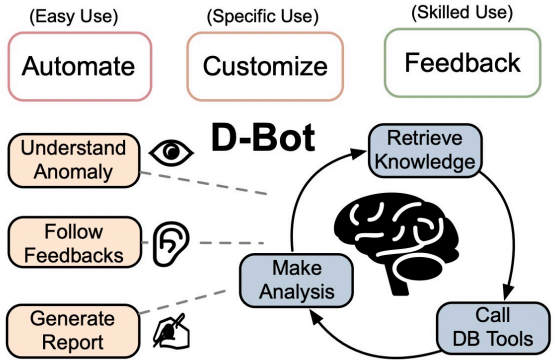
❑ Cost-Efficiency Optimization



❑ Verification and Reliability



❑ Reasoning and Self-Reflection



Technical Solutions

Approach	Definition	Purpose	Advantages	Examples
Pre-training	Initial training on large, diverse datasets to learn general patterns.	Establish foundational knowledge	Efficient learning; broad applicability	LLMs like GPT, DeepSeek
Fine-tuning	Additional training on task-specific datasets to refine model performance.	Adaptation to specific tasks	Improved accuracy for specific applications	Image classification, sentiment analysis
Post-training (RL)	Further training to refine strategies and performance.	Optimize decision-making	Enhanced strategy refinement; improved robustness	Game playing, autonomous driving
Prompting	Guiding model behavior using specific input formatting or instructions.	Directs model output without retraining	Flexible interaction; reduced need for labeled data	Interactive assistant tasks
RAG	Combines retrieval of relevant documents with generation tasks.	Enhances information retrieval	Access to external data sources; improved relevance	Knowledge-based question answering
Agent	Autonomous systems that perceive, reason, and act.	Decision-making in complex scenarios	Real-time interaction; adaptive strategies	Robotics, automated trading systems

Background of Unstructured Data/Data Lake Analytics

☐ Large-scale raw data in data lakes

- **Structured:** relational databases
- **Semi-Structured**: CSV, JSON, XML
- **Unstructured**: emails, documents, PDFs

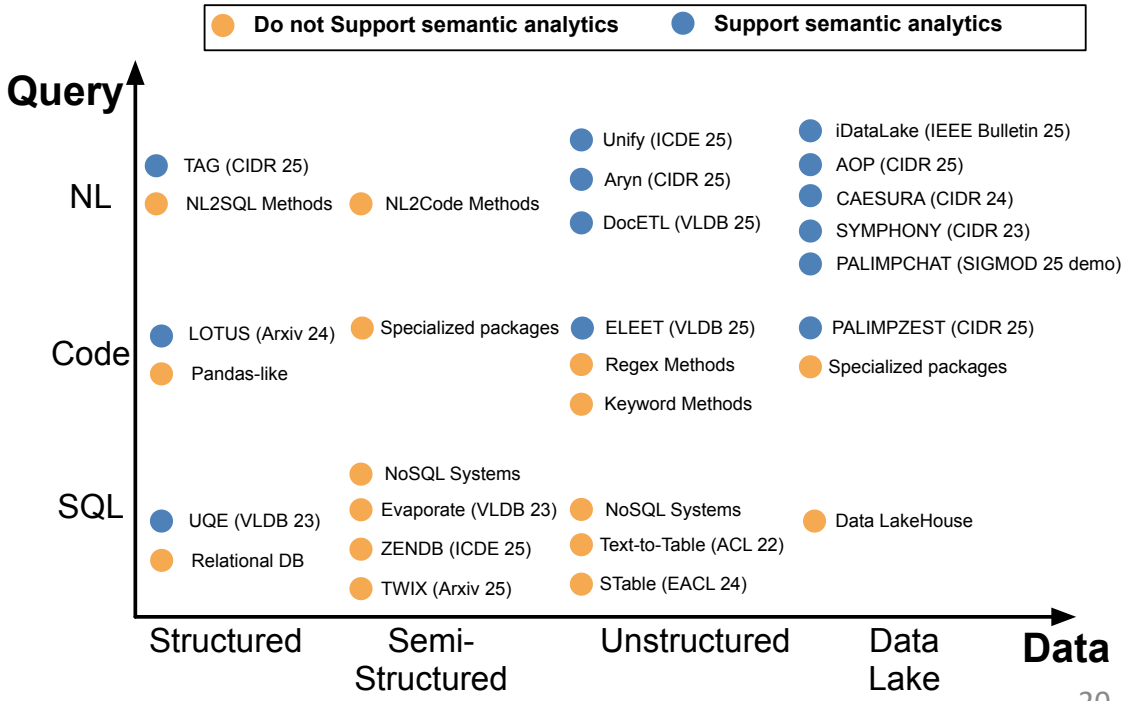
☐ Challenges

- No schema, hard to analyze
- Hard to understand data semantics
- No plan, hard to conduct data analytics

Difficult to conduct data analytics over data lakes

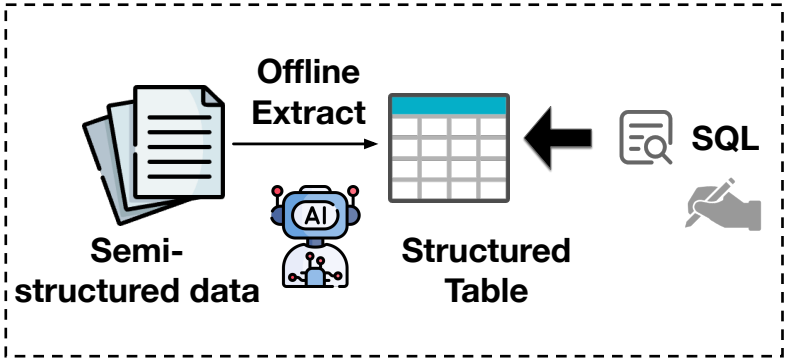
Summary of Different Data Analytics Methods

- **LLMs enable semantic data analytics over complex data**
  - **Understand, planning, reasoning**
- **Queries**
  - **NL:** Flexible, can express semantic conditions
  - **SQL:** Precise with strict syntax, hard to express semantic conditions
  - **Code:** Precise with strict syntax, hard to write
- Data
  - Textual Embedding
  - Extraction (Unstructure2Structure)

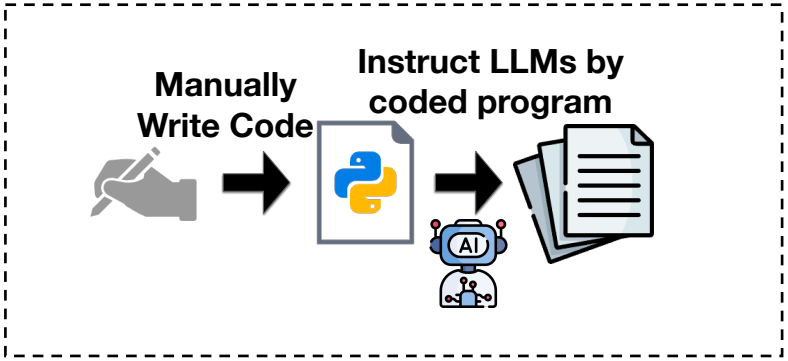


Classification of Unstructured Data/Data Lake Analytics Methods

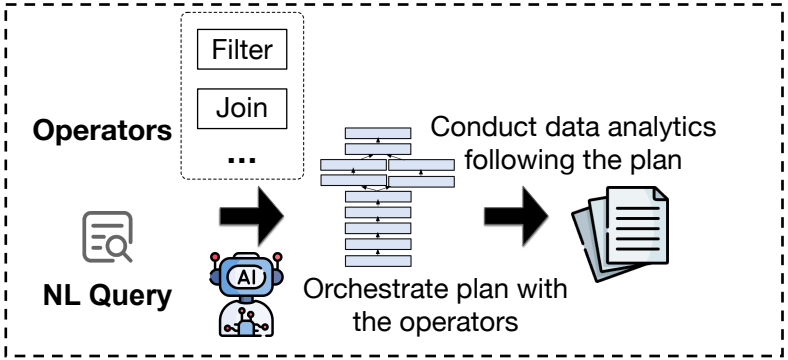
☐ *Structured Information Extraction*



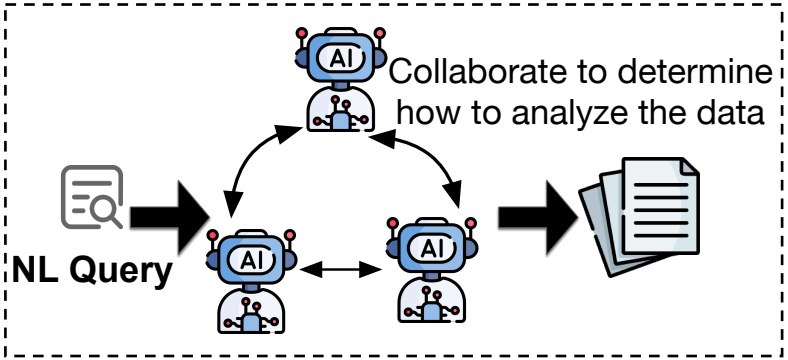
☐ *Manually Write Code*



☐ *NL2Pipeline*

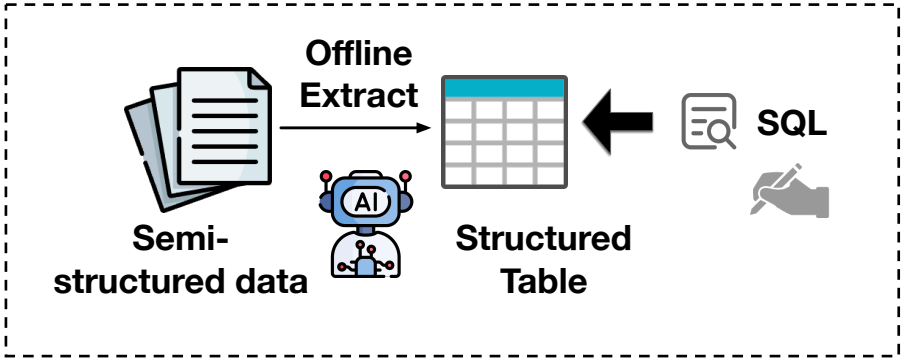


☐ *Data Agent*



Category 1: Structured Information Extraction

❑ **Key idea:** Extract structured tables from semi-structured data, then analyze by SQL



❑ **Challenges:**

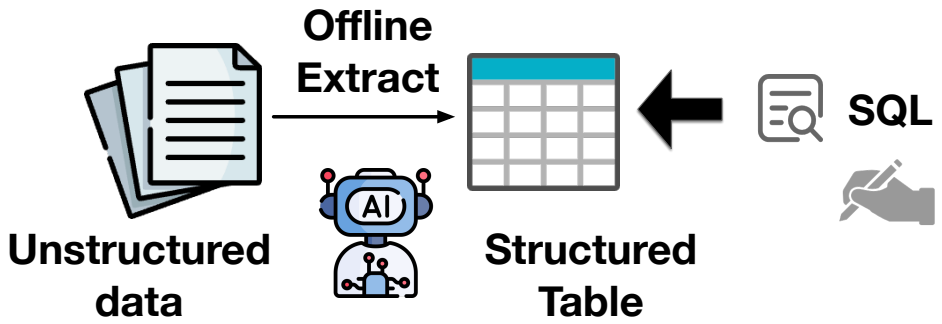
- How to determine the key schema automatically?
- How to improve the accuracy of information extraction?
- How to reduce the cost for structured information extraction?

Summary of Structured Information Extraction Methods

❑ **Asking LLMs to extract from each document is costly**

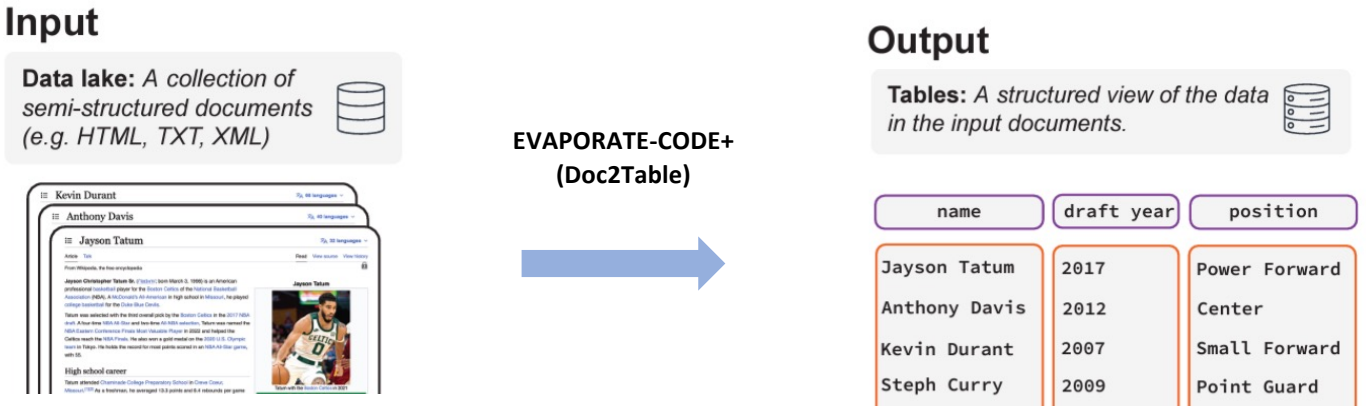
❑ **Common patterns in semi-structured data can be utilized to reduce the high LLM cost, potential solutions include:**

- **Generate code** to extract structured info. from fragments of templatized text
- Leverage **common hierarchical structures of headers** in templatized docs
- Leverage **common visual patterns** of templatized documents



Code Generation for Table Data Extraction from Semi-Structured Data

- ❑ **Hard to extract structured tables from documents**
- ❑ **Core Idea**
  - ❑ Feed sampled documents to the LLM, and prompt it to generate useful information that can form a structured table (e.g., writing code to extract the values of important attributes)
  - ❑ Unstructured data can thus be analyzed by analyzing structured tables through SQLs



Arora S, Yang B, Eyuboglu S, et al. Language models enable simple systems for generating structured views of heterogeneous data lakes[J]. VLDB, 2023.

Code Generation for Table Data Extraction from Semi-Structured Data

❑Prompt-based Table Data Extraction

❑ Schema Synthesis

- ❑ With a sampling subset of documents, it prompts LLMs to extract attributes based on their occurrence frequencies
- ❑ Rerank the extracted attributes by adjusting their frequency weights with LLMs

❑ Code Synthesis

- ❑ A heavy job to extract attribute values from every document → Prompt LLM to write code to extract the attribute values more efficiently
- ❑ **Limitation:** require documents follow certain structures (semi-structured)

Function Prompt

Here is a file sample:
<title>U.S. GDP Rose 2.9% in the Fourth Quarter </title>
<meta itemProp="datePublished"
content="2023-01-26T10:30:00Z"/>
…

Question: Write a python function called "get_date_published_field" to extract the "datePublished" field from the text. Include any imports.

```
from bs4 import BeautifulSoup
def get_date_published_field(text: str):
  soup = BeautifulSoup(
    text, parser="html.parser"
  )
  date_published_field = soup.find(
    'meta', itemprop="datePublished"
  )
  return date_published_field['content']
```

Arora S, Yang B, Eyuboglu S, et al. Language models enable simple systems for generating structured views of heterogeneous data lakes[J]. VLDB, 2023.

Table Data Extraction Based on Hierarchical Structures of Headers

❑ Key Insight:

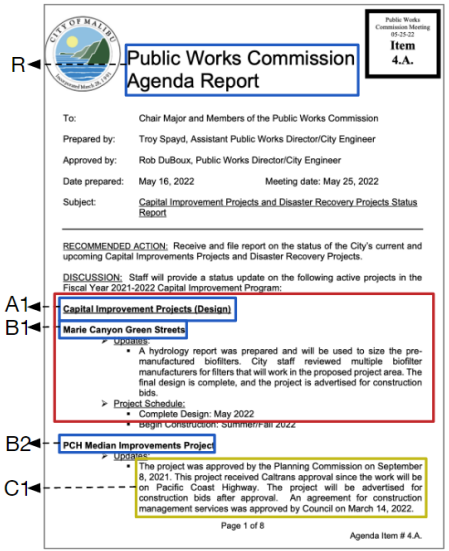
- Many documents are organized in the same way while with different content, e.g., reports,
- Such **templatized documents** follow **consistent hierarchical structures of headers**

❑ To identify such common structures:

- **Sample** a subset of documents
- Identify common structures by **matching the header structures** extracted by LLMs of the documents

❑ Document structure can be represented by a tree

- **Nodes** correspond to **header phrases** and sections in the document.
- **Edges** represent **semantic hierarchy** (e.g., Section > Subsection > Paragraph)
- This tree structure can be used for matching across documents

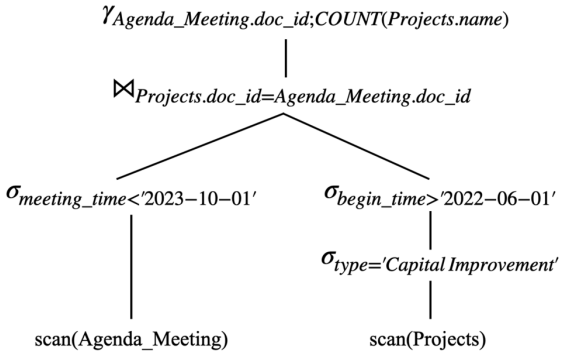


a) Civic Project Agenda Report

Table Data Extraction Based on Hierarchical Structures of Headers

- **Populating Tables (Structure Tree) from Documents**
  - Uses LLMs to identify **common structures** in a **sample document**
  - Uses **rule-based identification** for other documents based on the identified template (**Assume all documents follow the same template**)
- **Support SQL query (**attribute corresponds to certain text span and node**)**
  - Each node in the structure tree has a summary sketch (small text and metadata)
  - Efficiently locate the text span needed in the query

```
SELECT Agenda_Meeting.doc_id, COUNT(Projects.name)
FROM Projects, Agenda_Meeting
WHERE Projects.type = 'Capital Improvement'
AND Projects.begin_time > '2022-06-01'
AND Agenda_Meeting.meeting_time < '2023 October'
AND Projects.doc_id = Agenda_Meeting.doc_id
GROUP BY Agenda_Meeting.doc_id
```



- **Limitation:** Rely on the assumption of all documents strictly follow the same template

Table Data Extraction Based on Visual Patterns

- **Semi-structured data contain common visual patterns that store values of certain attributes**
  - **Field Prediction:** Identify which text phrases within **sampled documents** are template "Fields" (e.g., headers, keys) versus "Values" or "Metadata"
    - Extract phrases by **OCR** and check the text content **at the same location** across different documents by LLM
  - **Template Assembly:** Combine partial fields and identify their nested relationships by LLM
  - **Template-guided Data Extraction:** Process other documents based on the identified template

- **Limitation:** Rely on the assumption of documents **strictly follow** the same template (Values of the same attribute occur at the same position)

Report Criteria: Complaints Occurred Between: 1/1/2023 AND 11/20/2023

Complaints Detail Rpt #A-3

Champaign Police Department

Complaints By Date

Date	Number	Investigator	Date Assigned	Racial	Category / Type	Location Of Occurrence	Disposition	Completed	Recorded On Camera
5/15/2023	05-01	Johnson, Mary	5/16/2023	Yes	FORMAL Citizen	Downtown Park	SUSTAINED	6/1/2023	N/A Yes

Complainant: [redacted]	DOB:	Gender: FEMAL	Address: [redacted] Terr, Springfield IL 62701	H Phone: [redacted]

	Type Of Complaint	Description	Complaint Disposition
Complaint #: 1	R-4A.2 Conduct: Excessive Force	Discourteous Conduct	EXONERATED

	Name	ID No.	Rank	Division	Officer Disposition	Action Taken	Body Cam
Officer #: 1	Smith, Robert	763	LIEUTENANT	Field Operations	SUSTAINED	COUNSELING	No

Record 1

Date	Number	Investigator	Date Assigned	Racial	Category / Type	Location Of Occurrence	Disposition	Completed	Recorded On Camera
5/20/2023	05-02	Lane, Sarah	8/12/2023	No	INFORMAL Citizen	Not Stated	SUSTAINED	9/1/2023	N/A No

Complainant: [redacted]	DOB:	Gender: FEMAL	Address: [redacted], Champaign IL 61821	H Phone: [redacted]

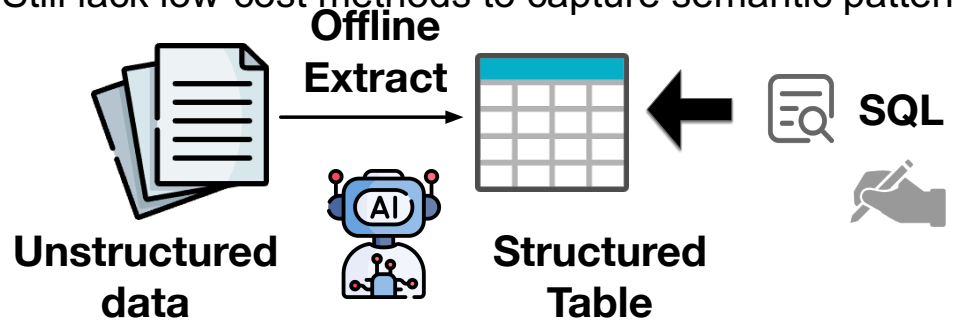
	Type Of Complaint	Description	Complaint Disposition
Complaint #: 1	R-3B.1 Courtesy:Profanity	Rude Conduct	NOT SUSTAINED
Complaint #: 2	R-3B.4 COURTESY: COMMENT	Discourteous Conduct	SUSTAINED
Complaint #: 3	R-5D Use of physical force	Wrong Action by Employee	EXONERATED

	Name	ID No.	Rank	Division	Officer Disposition	Action Taken	Body Cam
Officer #: 1	Carter, Michael	842	Senior Officer	Field Operations	SUSTAINED	NONE	No

Record 2

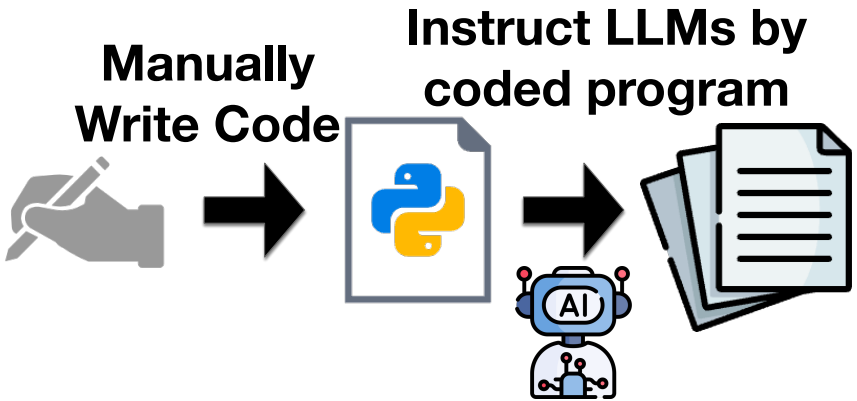
Takeaways of Structured Information Extraction Methods

- **Common patterns in semi-structured data can be utilized to avoid LLM calls**
  - **Keyword** or data following certain **regular expressions** can be extracted by simple code
  - **Structures of headers** can segment documents into spans with different semantic meanings
  - Common **visual patterns** that contain **key-value info** can be identified by a sample of data
- **Problems:**
  - **Low Generality:** Requiring data to follow different degrees of templates, i.e., semi-structured
  - **Low Accuracy:** The extracted tables are lossy representations of original data
  - **High Cost:** Still lack low-cost methods to capture semantic patterns in unstructured data



Category 2: Manually Write Code

- **Key idea:** Manually orchestrate execution process and conduct semantic operations following prompts in the code



- **Challenges:**
  - How to optimize the efficiency of the manually orchestrated plan?
  - How to reduce the LLM cost of the manually orchestrated plan?

Summary of Manually Write Code Methods

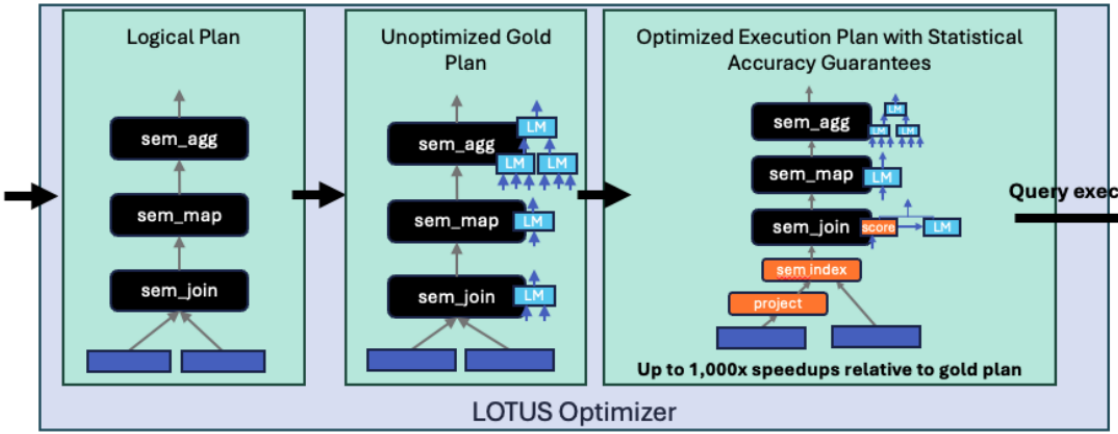
- **Manually orchestrated plans, though relatively accurate, face efficiency & cost issues**
- **Cost/Efficiency Optimization Methods**
  - **Bypass LLM:** Replace expensive LLM invocations with **cheap approximate methods**
  - **Model Cascade:** Use LLMs with **smaller #parameters** instead of **large #parameters**
  - **Approximate Processing:** Estimate aggregation queries by **executing on samples**
  - **Cost-based Optimization:** **Estimate execution cost** to optimize plans
  - **Query Rewrite:** **Reduce the amount of data** to be processed by LLMs

Semantic Operators for Tables of Unstructured and Structured Data

- Many real-world tasks require **semantic reasoning** over large datasets, such as summarizing research papers, extracting biomedical insights
- Semantic processing is **beyond the capability of relational operators**
- **Propose a set of pandas-like semantic operators:** support multi-row, natural language-specified operations over tables

```
def get_paper_digest(papers_df, projects_df):
  return papers_df\
    .sem_join(projects_df, "the paper {abstract:left} is
      highly relevant to my {research_areas:right}")\
    .sem_map("What is the key insight of the {abstract}
     and how does it relate to my {research_areas}",
     name="insights")\
    .sem_agg("Write a digest summarizing the research {
     insights}")
```

Semantic Operator Query



Semantic Operators for Tables of Unstructured and Structured Data

- **Definition:** Semantic operators are declarative, natural language-parameterized transformations over data
- Users can write pandas-like code to design their data analytics process

Operator	Description	Definition	Gold Algorithm
$sem\_filter(l\text{:}X \to Bool)$	Returns the tuples that pass the langex predicate.	$\{t_i \in T \mid l_M(t_i) = 1\}$	Compute $M(t_i, l) \forall t_i \in T$
$sem\_join(t\text{:}T,\ \ l\text{:}(X,Y) \to Bool)$	Joins a table against a second table t by keeping all tuple pairs that pass the langex predicate.	$\{(t_i, t_j) \mid l_M(t_i, t_j) = 1, t_i \in T_1, t_j \in T_2\}$	Compute $M(t_i, t_j, l) \forall t_i \in T_1, t_j \in T_2$
$sem\_agg(l\text{:}T[X] \to X)$	Aggregates input tuples according to the langex reducer function.	$l_M(t_1, ..., t_n) \forall t_1, .., t_n \in T$	Perform a reduce algorithm, recursively computing $a_{i+1,j} = M(a_{i,f(j)}, ..., a_{i,f(j)+n'}, l)$, $a_{0,j} = M(t_{f(j)}, ..., t_{f(j)+n'}, l)$
$sem\_topk(l\text{:}T[X] \to Seq[X],\ \ k\text{:}int)$	Returns an ordered list of the k best tuples according to the langex ranking criteria.	$\langle t_1, ..., t_k \rangle$ st $\forall (t_i, t_j), i < j \implies l_M(t_i, t_j) = \langle t_i, t_j \rangle$	Perform top-k sorting algorithm using pairwise comparisons, $M(t_i, t_j, l)$
$sem\_group\_by(l\text{:}X \to Y,\ \ C\text{:}int)$	Groups the tuples into C categories based on the langex grouping criteria.	$\underset{\{\mu_1,...\mu_C\},\mu_i \in V^{\mathbb{N}}}{\arg\max} \sum_{t_i \in T} \max_{j \in 1...C} l_M(t_i, \mu_j)$	Obtain centers $\mu_1, ...\mu_C$ with a clustering algorithm, and perform pointwise assignments $M(t_i, \mu_1, ..., \mu_C) \forall t_i \in T$
$sem\_map(l\text{:}X \to Y)$	Performs the projection specified by the langex.	$\{l_M(t_i), \forall t_i \in T\}$	Compute $M(t_i, l) \forall t_i \in T$

Semantic Operators: A Declarative Model for Rich, AI-based Data Processing. arXiv 2025

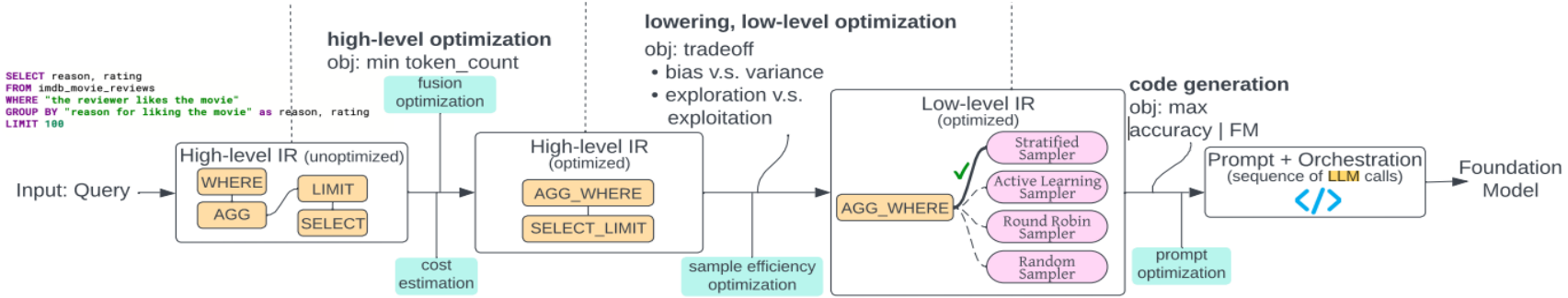
Replace LLMs with Cheaper Approximations for Acceleration

- **Main idea:** Not all cases must be processed by LLMs to get correct result
- Use a **fast-but-imperfect** approximate model to handle easy cases, reserving the **slow-but-accurate** model only for hard decisions
- **Execute on data samples to determine whether to use approximations**
- **Examples:**
  - **Filter:** Use embedding-based classifier or distilled LLMs to filter out obvious matches/mismatches
  - **Join**: Use embedding-based similarity to filter tuple pairs
- **Limitation:**
  - Optimization degree is low; cannot optimize at the level of plan structure
  - Inappropriate adoption of approximation methods results in low accuracy

Semantic Operators: A Declarative Model for Rich, AI-based Data Processing. arXiv 2025

Approximate Processing for Accelerating Aggregation Queries

- ❑ UQE enables user to query **tables containing unstructured columns** by SQL with semantic predicates
- ❑ Support semantic predicates by **prompting LLMs** for processing unstructured columns
- ❑ Propose stratified sampling for accelerating aggregation queries
  - ❑ Accelerate by **reducing the amount of data processed by LLMs**
  - ❑ Embed all rows and cluster them into K groups
  - ❑ Perform stratified sampling within clusters to select a small number of rows
  - ❑ Use weighted averaging of sampled results to unbiasedly estimate aggregation queries



Online Active Learning of Lightweight Model for Non-Aggregation Queries

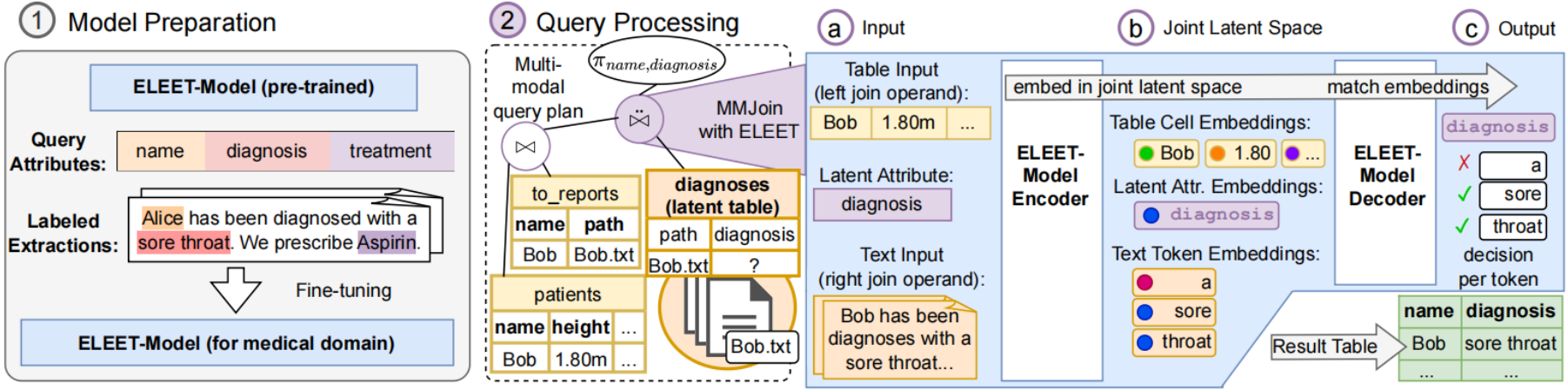
- **Online Active Learning for Non-Aggregation Queries to reduce LLM cost**
  - Embed all rows and initialize a lightweight model (**randomly initialized**)
  - At each step, sample rows with highest predicted relevance (predicted by the lightweight model, **ensure sample effectiveness for exploitation**) plus small noise (**ensure diversity of sampled data, for exploration**)
  - Call LLMs to label the sampled data and update the lightweight model
  - Repeat above process, and finally process remaining data using the lightweight model

```
SELECT agent_name, "reason to cancel"
FROM airline_customer_service_log
WHERE "the customer asked to cancel
       the flight"
LIMIT 100
```

- **Limitation:** Hard to collect enough data online for accurate model training, e.g., label skewness for extreme selectivity

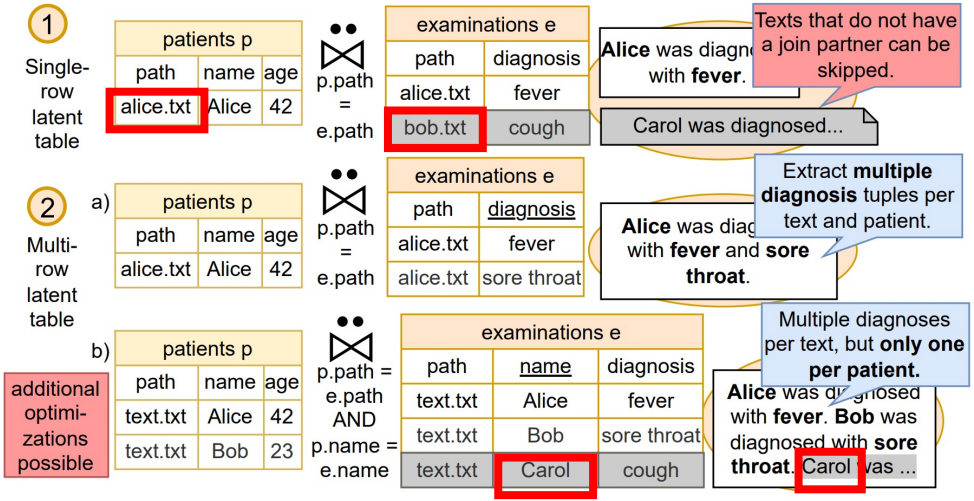
Pretrain Lightweight Language Models for Querying Tables and Text

- **Scenario**: Query over **both structured tables and unstructured text**
- **Relational operators are insufficient to handle unstructured text**
- **Method**:
  - **Propose multi-modal operators that take documents as input, and output tables**
    - Since the outputs are tables, new operators can be included in the same plan with relational operators
  - **Using LLMs to implement these operators is costly**



Pretrain Lightweight Language Models for Querying Tables and Text

- **Rather than extracting structured data in advance, ELEET conducts online information extraction with the SLMs**
  - **Key idea:** Information in tables can help locate structured information in text
  - SLMs are more efficient than LLMs, ensuring efficient online extraction
  - **Examples:**



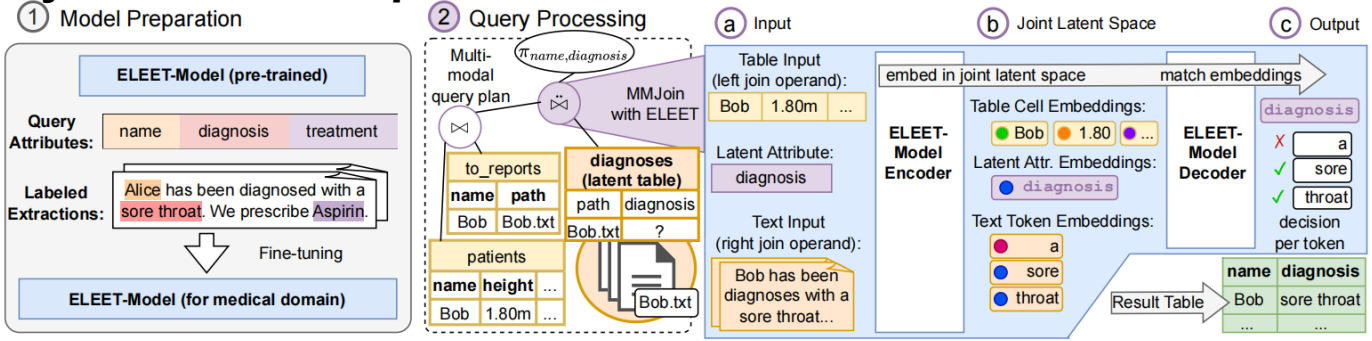
❑Structured table operations avoid the processing of some documents (Avoid processing **bob.txt and carol.txt**)

❑Help extract multiple tuples from a text (**multiple diagnosis for Alice**)

❑If the text contains multiple instances (**Alice, Bob, Carol…**), structured data (**name=Carol**) can help identify the target instance

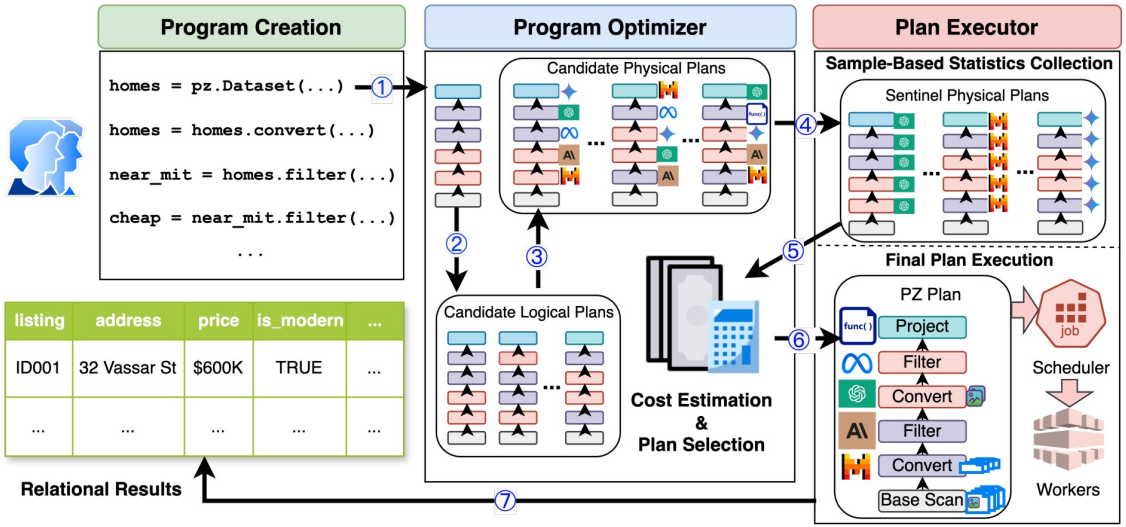
Limitation of Specialized Small Language Models

- **Cannot support complex semantic analytics**
  - SLMs have weaker semantic understanding ability than LLMs
  - Only supports operations supported by traditional databases (queries text like tables)
- **Lack world knowledge**
  - SLMs do not have world knowledge like LLMs
  - Cannot support multi-step logical reasoning with world knowledge
- **Rely on the assumption that attributes in text are known**



Cost-based Plan Optimization for Improving Performance

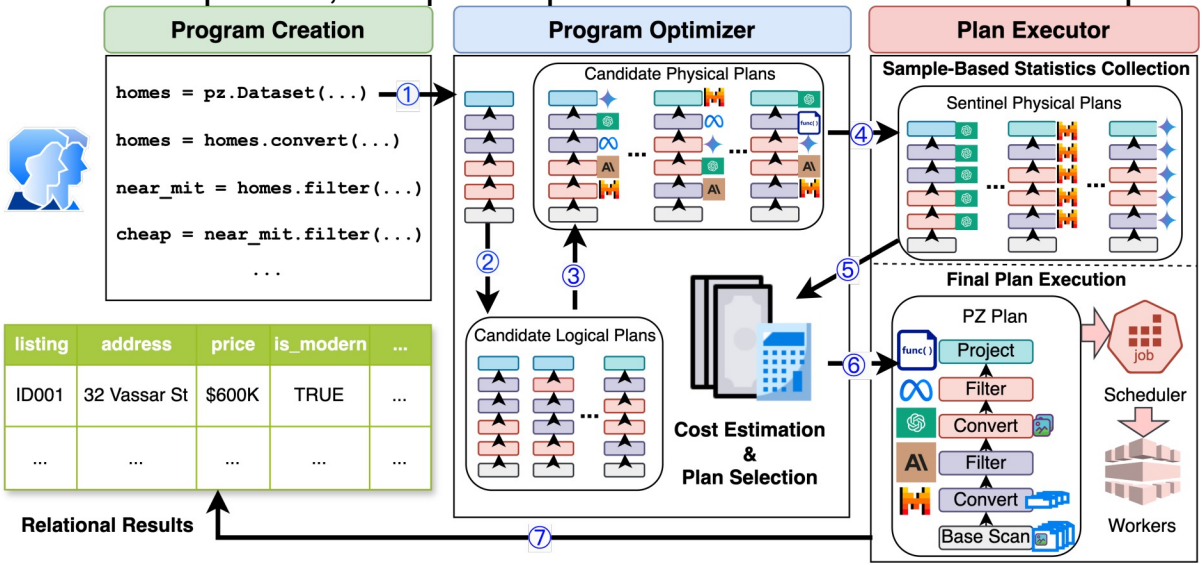
- **PALIMPZEST allows users to pose AI-powered analytics queries over collections of unstructured data using declarative APIs**
  - Users manually set **target runtime, LLM cost, and result quality**
  - Transforms the program into various equivalent logical plans
  - Selects the plan with **lowest estimated cost** under runtime and quality constraint



- **Challenge:** Cost estimation for execution over unstructured data is difficult

Cost-based Plan Optimization for Improving Performance

- For plan selection, needs to estimate the performance of each plan
- In the worst case, requires enumerating an exponentially number of plans
- Assumption: operators are independent
  - Estimate each operator, compose operators estimations to estimate plan performance



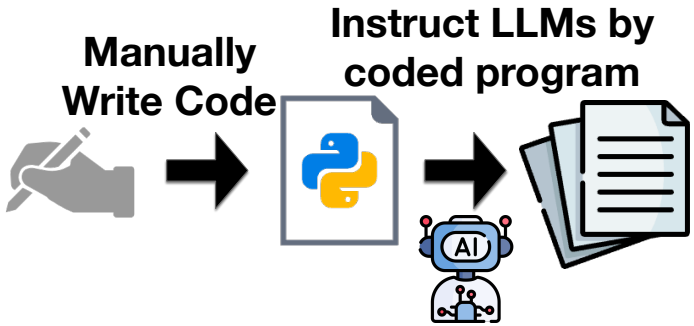
Cost-based Plan Optimization for Improving Performance

- **Method:**
  - **Executes a set of plans on a small set of sampled data**
  - **Obtain per-operator estimates:**
    - distribution of runtimes, per-record cost and quality of each operator
  - **Estimate performance of each plan by composing its per-operator estimates**
    - Sums the runtime
    - Sums the cost
    - Takes the product of their qualities
- **Limitation:** Estimation by executing over sampled data is time-consuming and inaccurate, which limits optimization effectiveness

Takeaways of Manually Write Code Methods

❑ **Summary of different optimization methods:**

- Using proxy methods may **influence accuracy** of the results
- Approximate processing is **not universal**, only support aggregation queries
- Cost-based optimization directly relies on the accuracy of **cost estimation**
  - Require cardinality estimation for semantic predicates. Uniform sampling is **inaccurate**

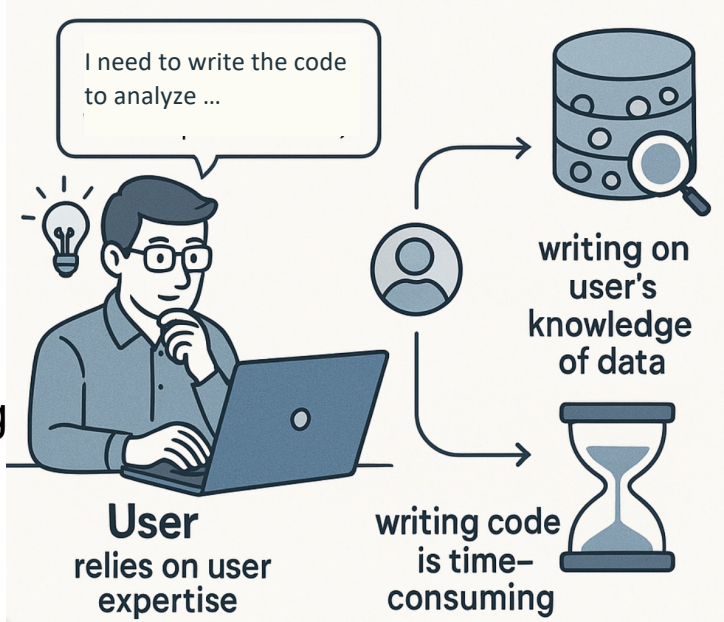


In addition to LLM cost, human cost should also be considered

Limitations of Manually Write Code Methods

■ Users query by writing code

- Rely on user expertise
- Rely on user's knowledge of data
- Coding and debugging is time-consuming

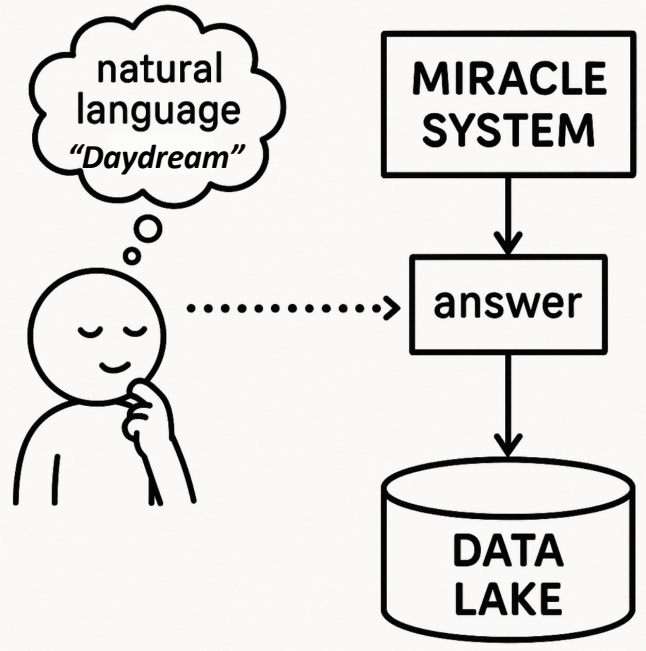


Even though the LLM cost can be optimized…

Human cost is too high! Can we make analytics more accessible?

Category 3: NL2Pipeline

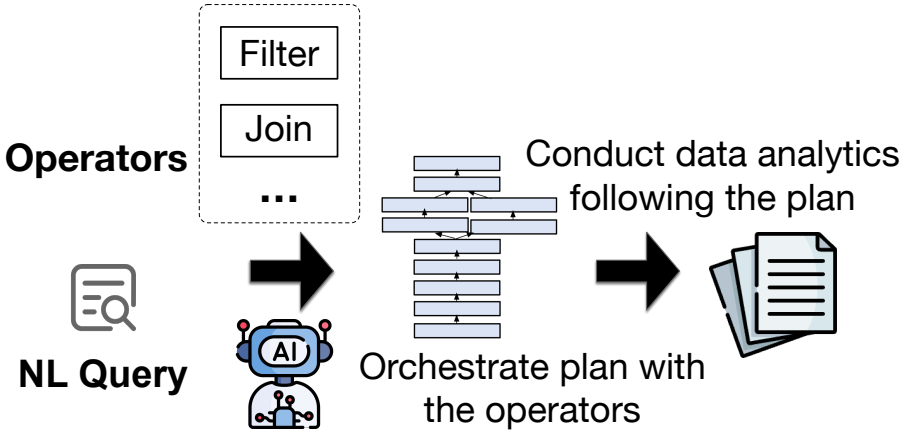
- **Natural language is a easy way to express analytics queries**
  - ✓ Easy to access for users
  - ✓ Low human effort
  - ✓ Difficulties are left to the analytics system



How to answer natural language analytics queries automatically?

Category 3: NL2Pipeline

- **Key idea:** Predefine the semantic operators and transform the natural language query into plans composed of the operators for execution



- **Challenges:**
  - How to automatically generate plan with correct logic?
  - How to optimize the efficiency of the generated plan?

Summary of NL2Pipeline Methods

❑**Candidate plan generation solutions for NL2Pipeline:**

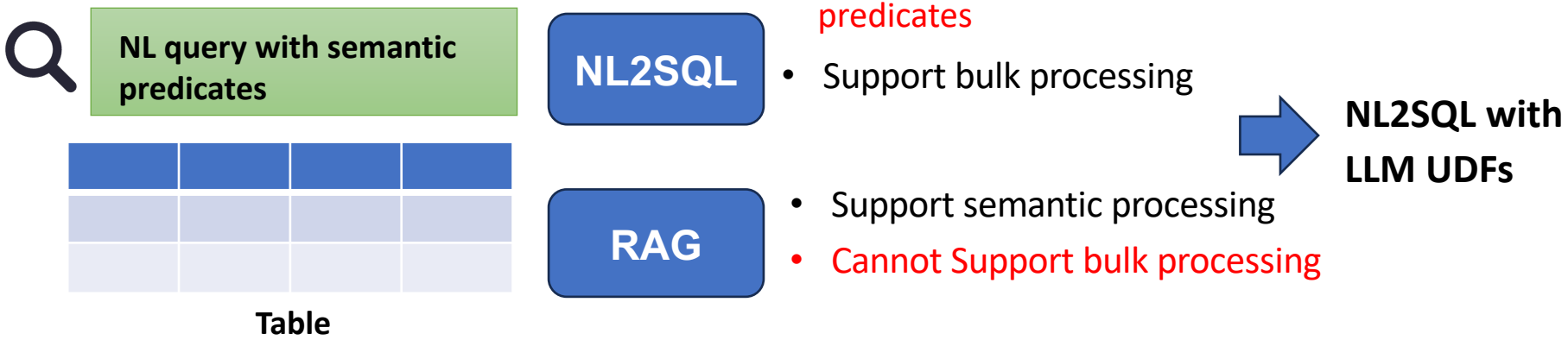
① Use **static** predefined execution process

② **Instruct LLMs** to determine the plan by providing descriptions of the available operator

③ **Progressively match** appropriate operators for the query

Using Predefined Static Execution Process for Data Analytics

- **TAG:** Focus on natural language questions that can be expressed in relational algebra over tables
- Support semantic predicates by UDFs that invoke LLMs
- **Main idea:** Transform the natural language query into **SQLs with LLM UDFs**



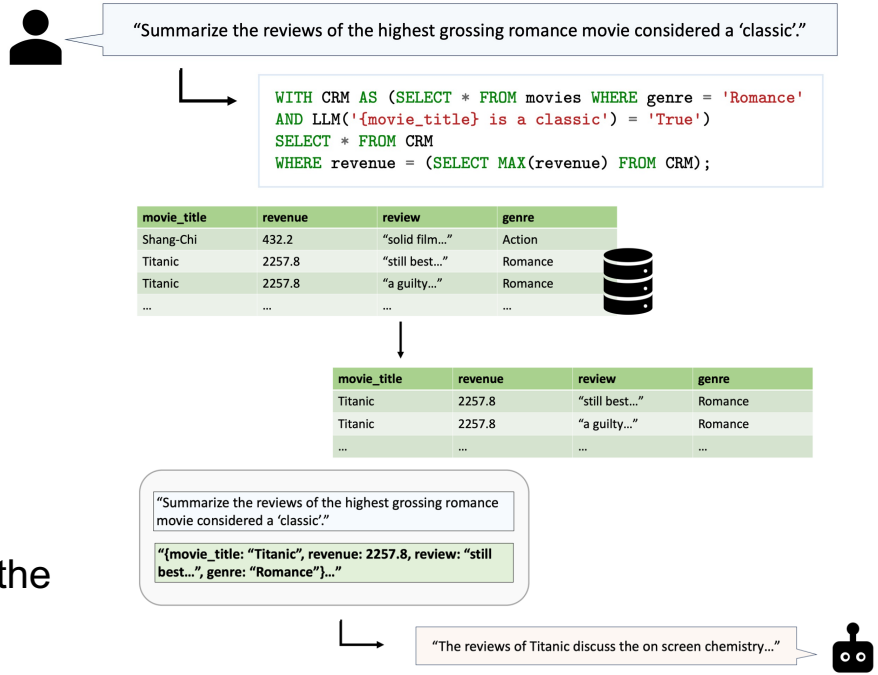
Using Predefined Static Execution Process for Data Analytics

☐ Predefined Static Execution Process in TAG:

1. **Query Synthesis:** Converts the user query into a SQL and express semantic predicates as LLM-based UDFs

2. **Query Execution:** Executes the SQL query within a database system

3. **Answer Generation:** Uses an LLM to generate the final NL answer based on the user query and retrieved table data

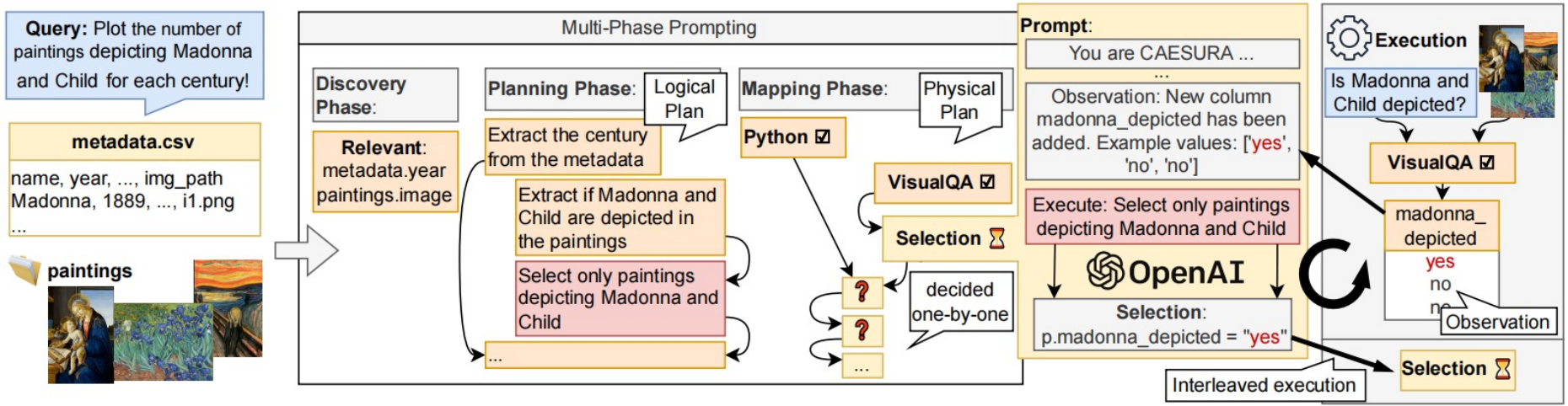


☐ **Limitations:** Only support queries that can be represented by relational algebra

Do not support multi-step logical reasoning and execution is costly

Instruct LLMs to Generate Plans of Multi-Model Large Models

- **Problem**: Answer natural language queries over multi-modal data including tables, text, figures
- **Method**: Transforms natural language queries into executable multimodal query plans by prompting LLMs
  - The **prompting is manually designed** with multi-phase to improve plan quality
  - The descriptions of data, available operators and query is included in the designed prompt



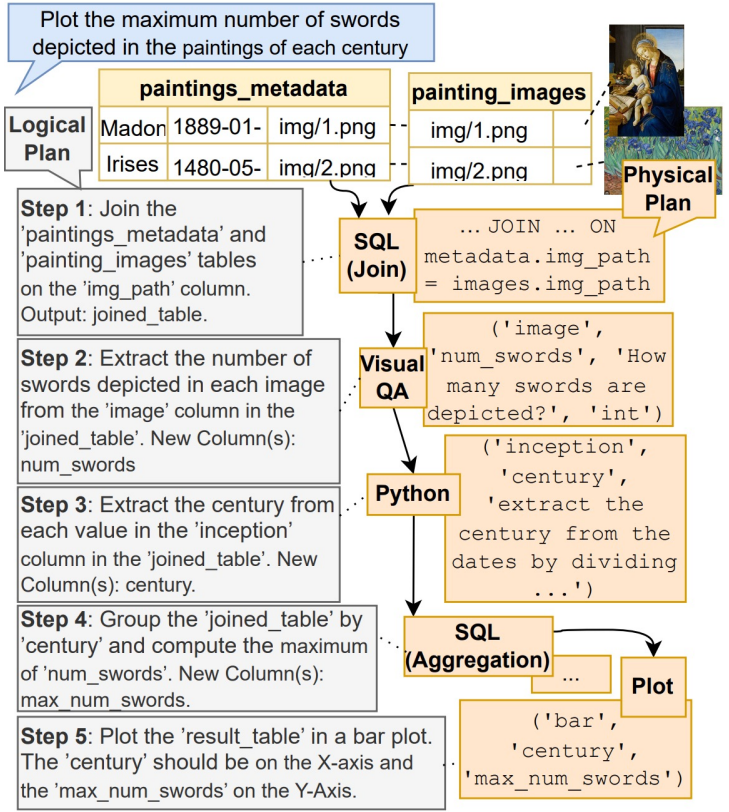
Instruct LLMs to Generate Plans of Multi-Model Large Models

❑Multi-phase Prompting

- ❑ **Planning**: Prompt LLMs to write a step-by-step logical plan in natural language
- ❑ **Mapping**: Convert each logical step into an executable operator (SQL, Python, Visual QA, etc.)

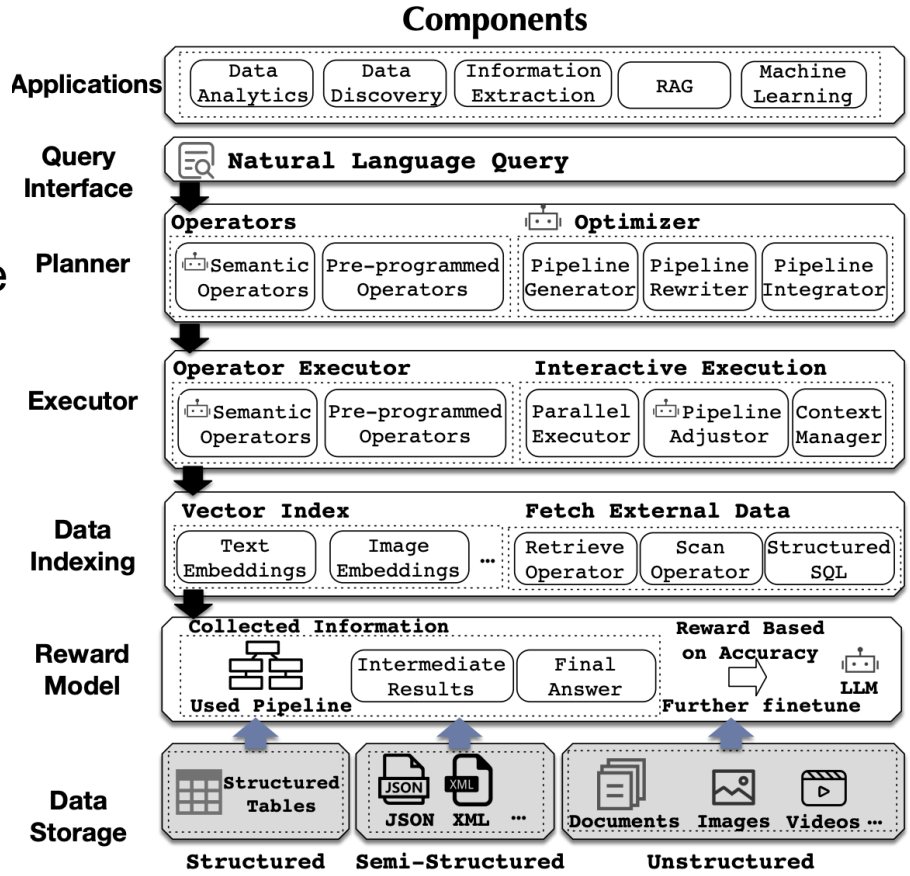
❑Limitations:

- ➢ The plans generated by directly prompting the LLMs suffer from **low accuracy**
- ➢ The generated plans are sequential with **low efficiency**



Instruct LLMs to Generate Plans of Semantic Operators

- **Problem**: Answer natural language queries over data lakes including structured, semi-structured and unstructured data
- **Key idea**: human-crafted pipelines are essentially well-constructed assemblies of **standard semantic operators**
  - Identify key operators for building effective LLM pipelines
  - Provide operator descriptions for orchestrating pipelines by LLMs



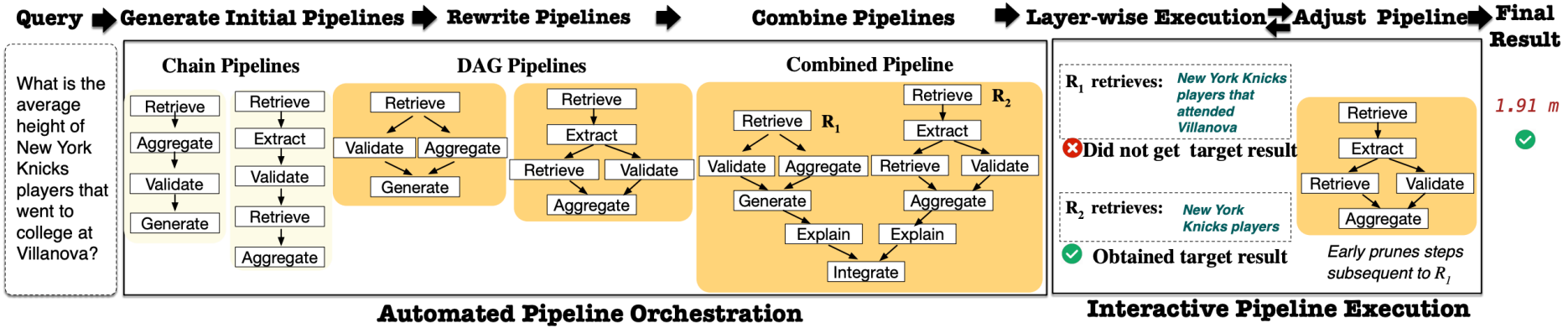
Instruct LLMs to Generate Plans of Semantic Operators

❑ Method:

- ➢ **Instruct LLMs** to generate multiple chain-format pipelines by prompts
- ➢ **Optimize the pipelines** into DAG structure by analyzing the operator dependencies
- ➢ **Combine** different pipelines together
- ➢ **Layer-wise pipeline execution** to obtain the final result

❑ Benefit: Reduce plan generation complexity as each operator can correctly solve a subtask

❑ Limitation: Rely on LLMs to generate plan by prompts, which may be beyond LLM capabilities



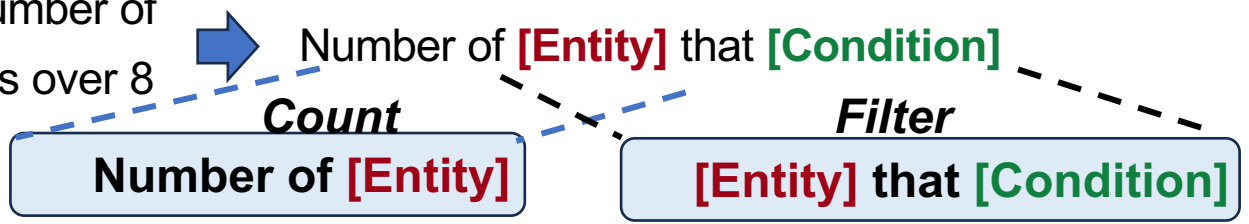
Wang J, Li G. Aop: Automated and interactive llm pipeline orchestration for answering complex queries. CIDR, 2025.

Progressively Match Appropriate Operators for the Query

- Unify proposes a set of operators for unstructured data analytics
- **Observation:** Each operator corresponds to certain NL expressions
  - **Examples:**
    - **Filter:**
      - Questions that are related to football
      - Films that have ratings over 8

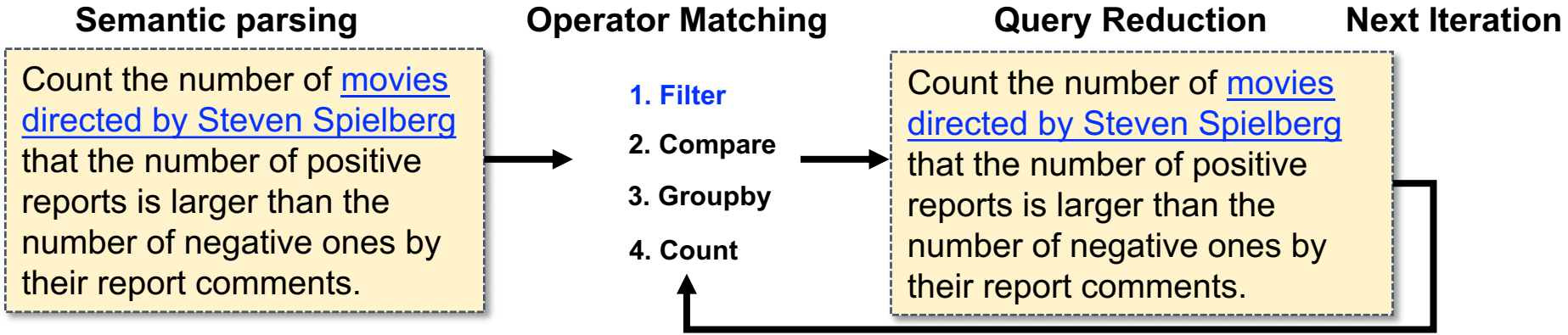
      → **[Entity] that [Condition]**
    - **Count:**
      - Number of articles

      → **Number of [Entity]**
- **Key idea:** Prepare operator expressions for online matching
  - **Example Query:** Number of films that have ratings over 8



Progressively Match Appropriate Operators for the Query

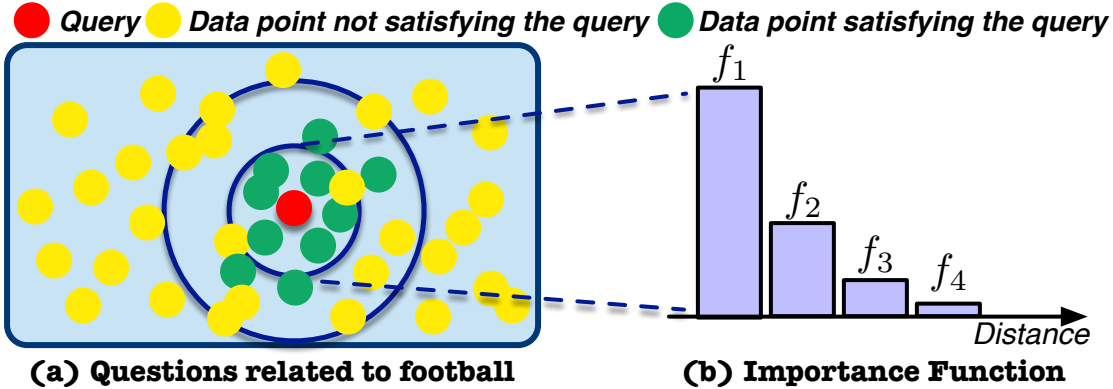
- **Overview**: progressively identifying appropriate pre-defined logical operators and reducing the query with the operators.
  - ① **Semantic Parsing**: extract the logical representations from the query
  - ② **Operator Matching**: identify the matched logical operators
  - ③ **Query Reduction**: reduce with the logical operators to generate a plan
  - ④ **Error Handling**: backtrack to the previous reduction



Jiayi Wang, et al. Unify: An Unstructured Data Analytics System. (ICDE 2025)

Cost-based Plan Optimization with More Accurate Cardinality Estimation

- **Observation:** data points satisfying the query often have high semantic relevance with the query
- **Key Ideas**:
  - Estimation by importance sampling
  - Focus more on data points closer to the query vector

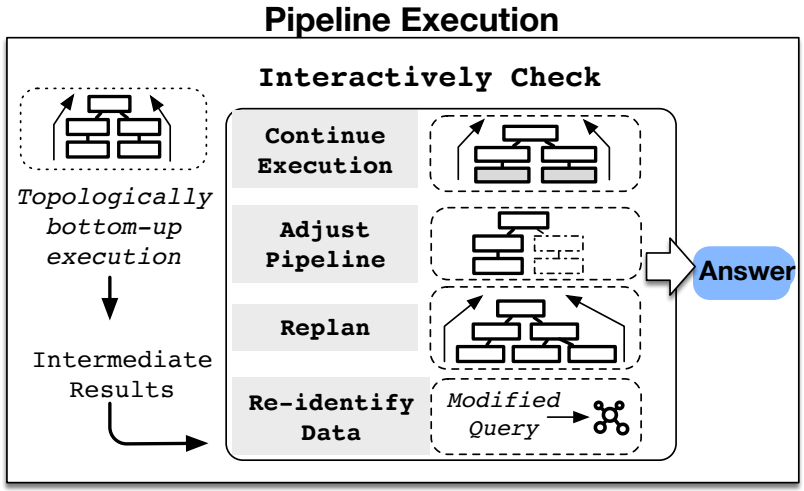


Jiayi Wang, et al. Unify: An Unstructured Data Analytics System. (ICDE 2025)

Optimize Execution Efficiency of Generated Plans

Problem: How to optimize the execution efficiency of the plan?

- **Plan Adjustment During Execution:** adjusts the plan dynamically when operator execution fails or can be **replaced by other low-cost operators**
- **Parallel Execution** for **low latency**

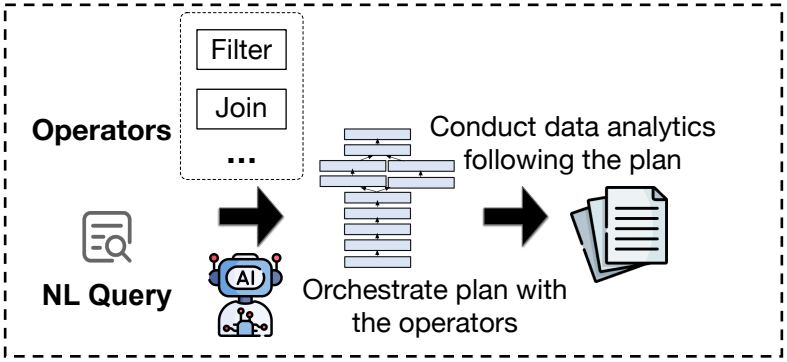


Jiayi Wang, et al. iDataLake: An LLM-Powered Analytics System on Data Lakes. (IEEE Data Engineering Bulletin 2025)

Takeaways of NL2Pipeline Methods

❑ **Summary of different pipeline generation methods:**

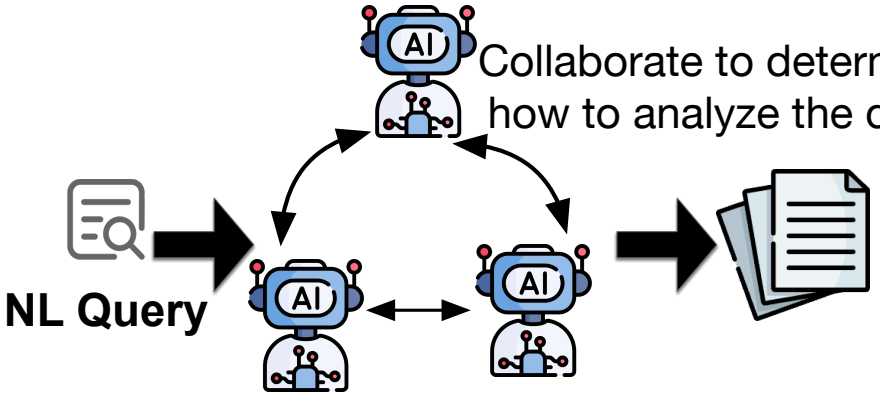
- Static predefined execution process **cannot handle complex queries**
- Directly instructing LLMs to generate pipeline achieves **limited accuracy**, since
- Progressively matching appropriate operators is **limited by inflexibility of operaotrs**, strict requirement of intput/output relationship of operators



Operators are still not flexible enough and restricts the flexibility of NL

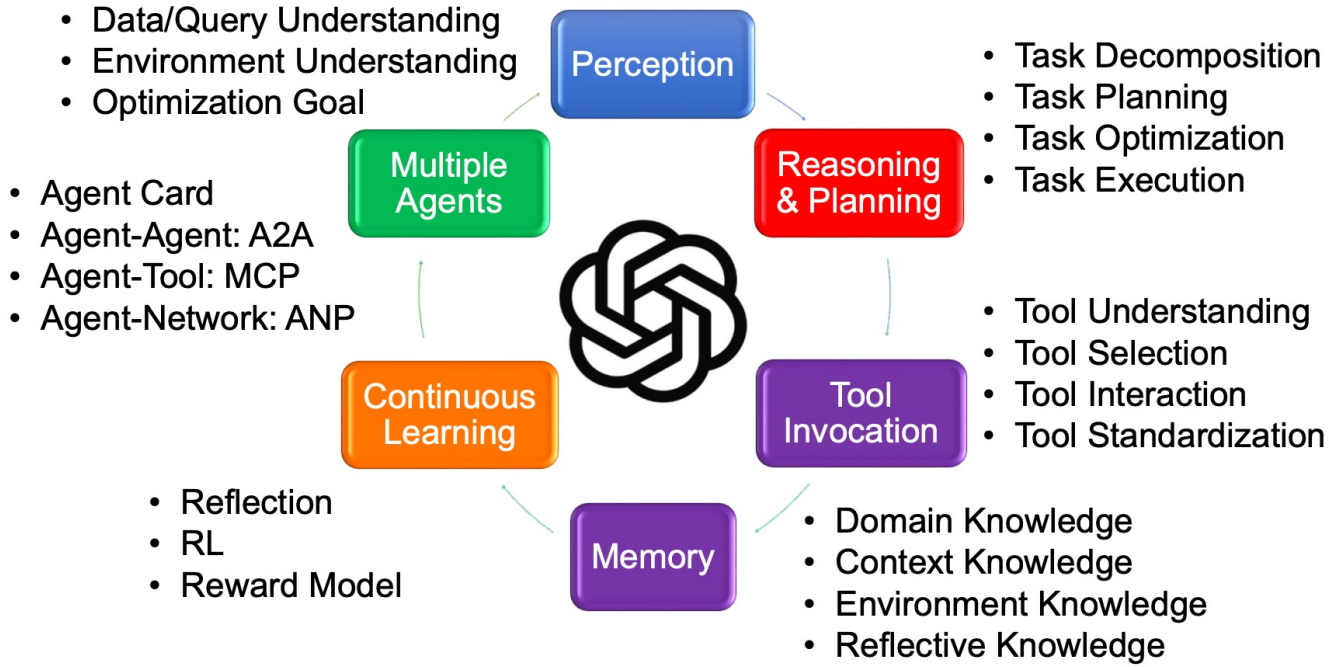
Category 4: Data Agent

- **Data Agent:** designed to autonomously carry out data-related tasks with capabilities for knowledge comprehension, automatic planning, and self-reflection of LLMs



- **Challenges:**
  - How can data agents **understand** queries, data, other agents, and tools?
  - How can data agents **orchestrate** effective and efficient pipelines to bridge the gaps between user requirements and underlying heterogeneous data?
  - How to **schedule and coordinate** agents/tools to improve effectiveness?

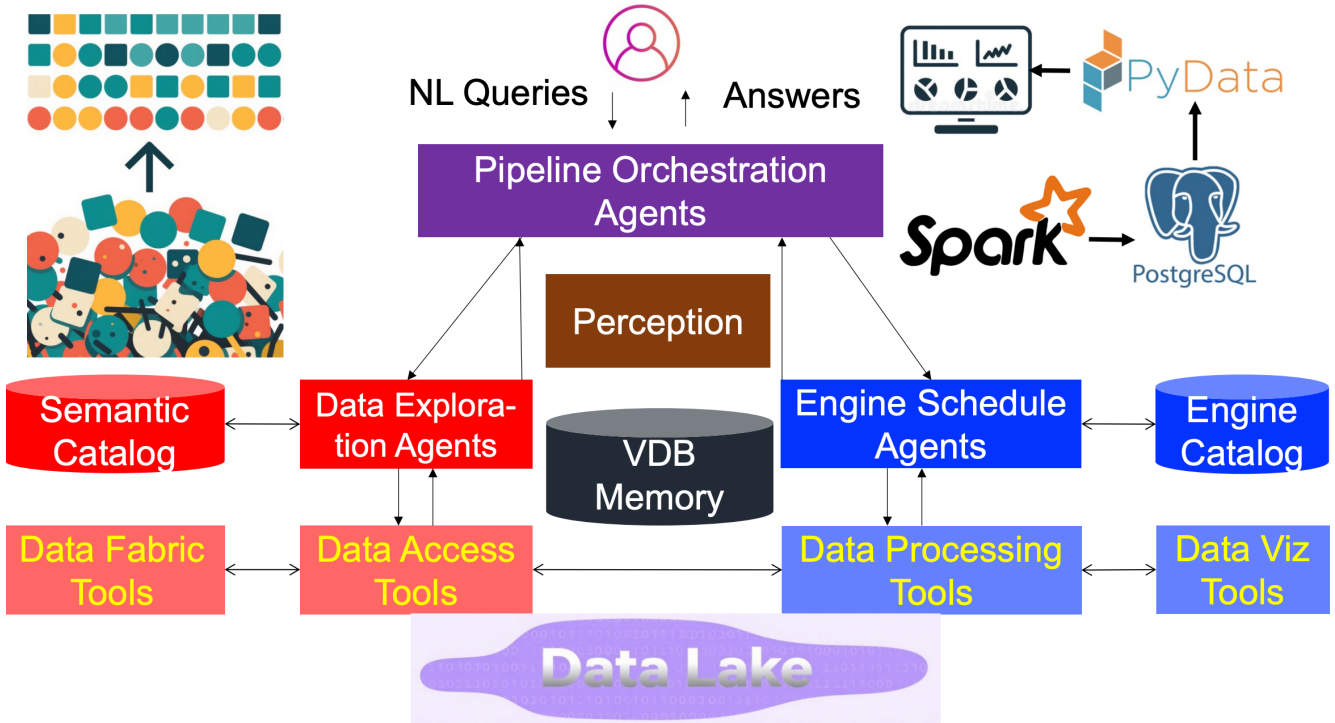
Key Factors of Data Agent



- **The Data Agent is designed to autonomously carry out data-related tasks with capabilities for knowledge comprehension, automatic planning, and self-reflection.**

Zhaoyan Sun, et al. Data Agent: A Holistic Architecture for Orchestrating Data+AI Ecosystems. (IEEE Data Engineering Bulletin 2025)

A Framework Design of Data Agent



- **Need to solve challenges in multiple important components:**
  - Unified semantic catalog, data fabric over heterogeneous data, agent-agent interaction…

Zhaoyan Sun, et al. Data Agent: A Holistic Architecture for Orchestrating Data+AI Ecosystems. (IEEE Data Engineering Bulletin 2025)

Summarization of Unstructured Data/Data Lake Analytics Methods

Method Type	Challenges	Advantages	Drawback
Structured Information Extraction	• Determine schema • Improve extraction accuracy • Reduce extraction cost	**Fast analytics:** Only involve structured data	**Low generalizability**: semi-structured **Low accuracy**: information loss **High cost**: extract large-volume data
Manually Write Code	• Plan efficiency • Reduce LLM cost	**High accuracy**: Human-craft plans	**High human cost**: Human-craft **Time-consuming**: Coding takes time
NL2Pipeline	• Automatically generate plans with correct logic • Plan efficiency	**Ease to use**: No human; NL interface	**No Theoretical guarantee**: NL is open-ended and no strict syntax like SQLs
Data Agent	• Understand data and queries • Orchestrate plan with agents • Coordinate agents	**Ease to use:** No human **High Flexibility**: No need to maintain operator set **High Generalizability**: Easy to adapt to other tasks	**High LLM cost:** a large number of LLM invocations **Hard to design**: Effective agentic workflow with multiple components is hard to design

Data4LLM

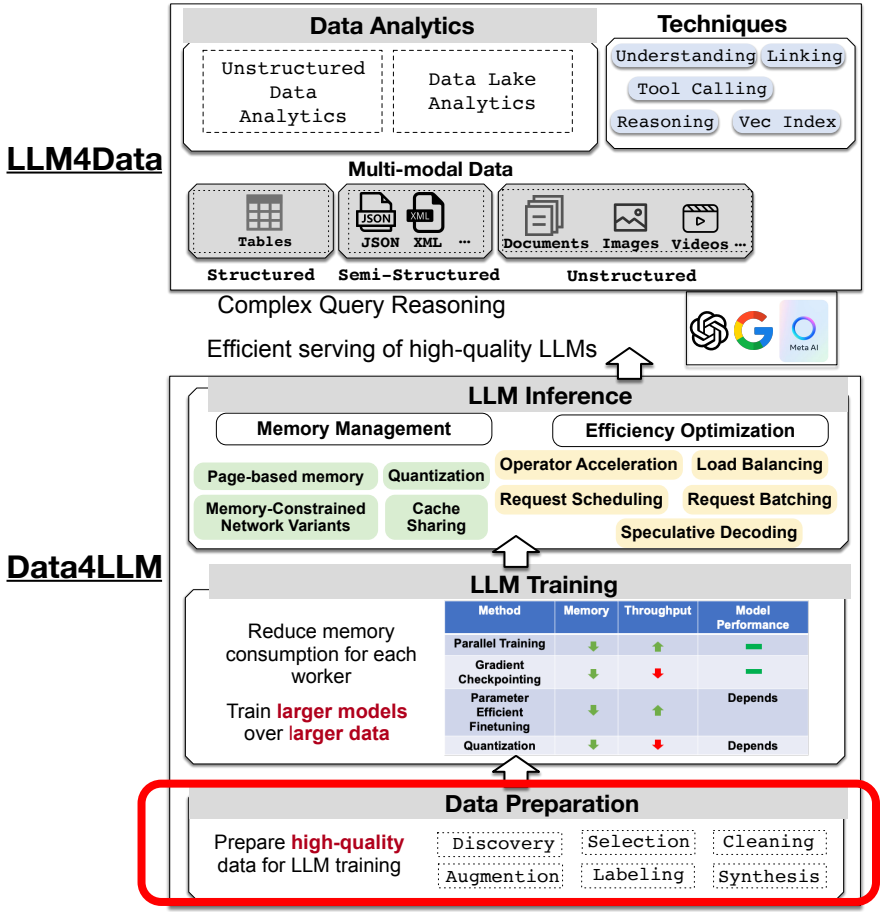
❑LLM4Data Techniques

- LLM Prompting
- RAG & Vector DB
- Data Agents
  - Unstructured Data Analytics
  - SQL + Semantics
  - Data Lake Analytics

❑ Data4LLM Techniques

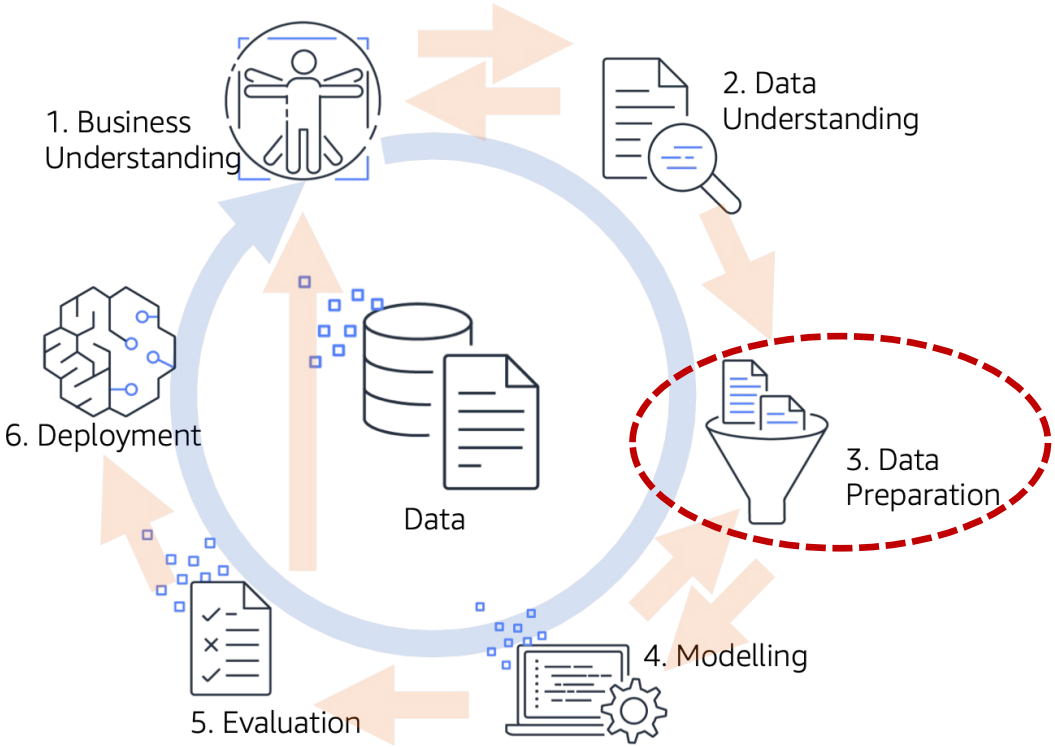
- Data Preparation
- LLM Inference
- LLM Training

❑ Open Challenges



Data Preparation in machine learning life cycle

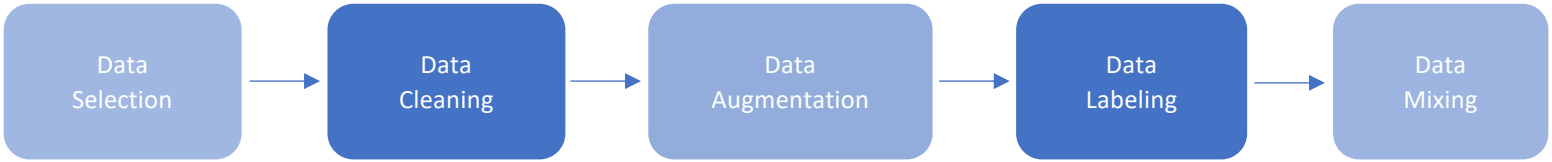
- **Data Preparation**: The prerequisite to building high-performance model
  - Turn big dirty data into a subset of good data
  - Select, clean, augment, label, mix, and even synthesize data



Data preparation for machine learning using Amazon Timestream

Data Preparation in machine learning life cycle

- Data Preparation: Turn **big dirty data** into **a subset of good data**

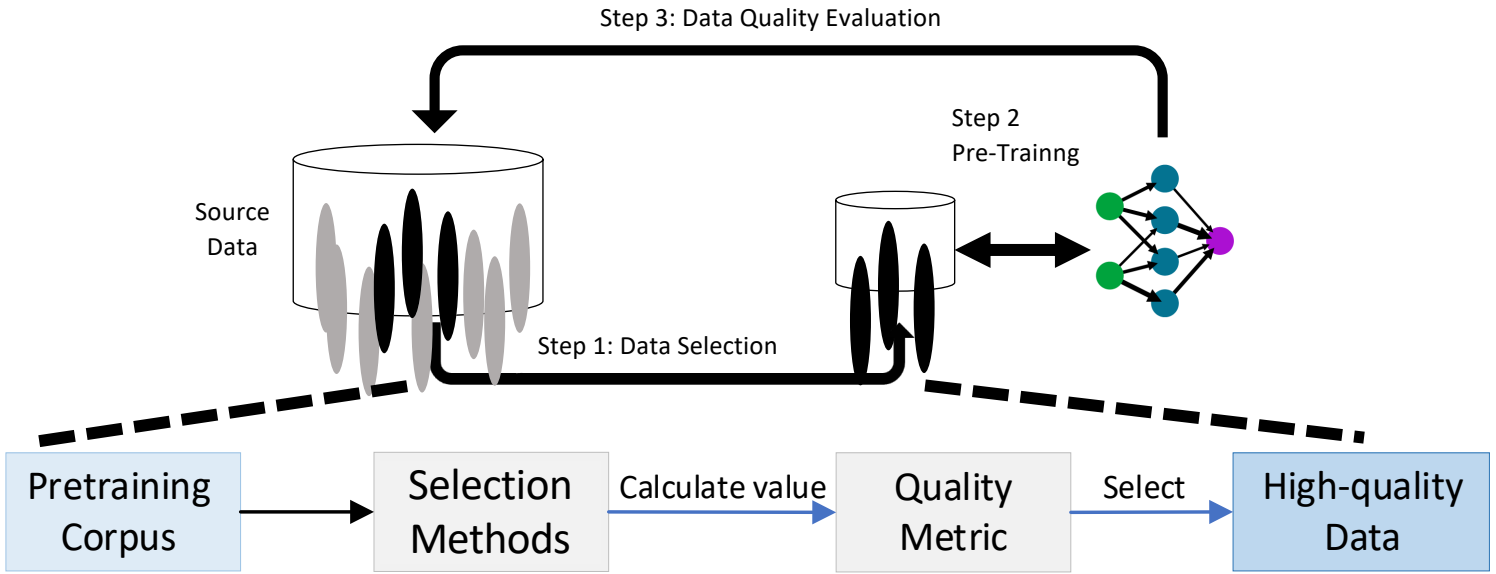


Data Preparation Pipeline

- **Challenges**
  - Rely on experts
  - Time-consuming
  - Hard to discover the optimal solution
    - E.g., numerous candidate pipelines

Data Selection for LLM

- **Data Preparation: Turn big dirty data into a subset of good data**
  - Data Selection: Obtain reduced representation in volume but produce similar or even better training results



Data Selection For LLM

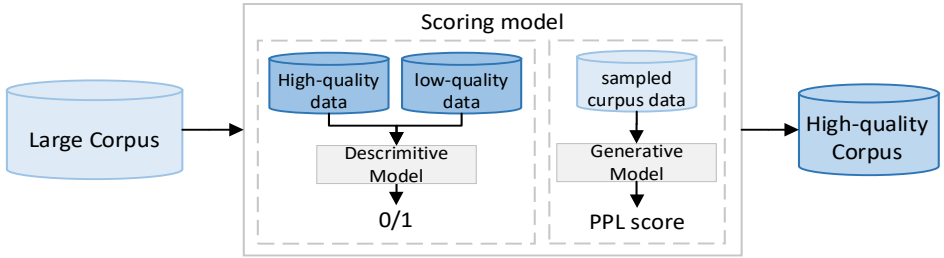
Challenge: How to select high-quality pretraining datasets?

- Content-based Selection: Select high-quality data (e.g., data edited by humans; data from trustable sources like peer-reviewed articles)
- Classification-based Selection: Identify data points that are likely from the same (or similar) distribution as a known “high-quality” corpus of data points
  - **Step 1: Feature Hashing**
    - Consider text words "the","quick","brown","fox". Using a hashing function, these might be mapped to indices [5,17,3,12] in a feature vector of size 20.
  - **Step 2: Train Classifier with Curated / Other Pages**
    - Class 1 (Curated Content): High-quality sources like Wikipedia, books, and selected websites.
    - Class 2 (Other Webpages): Typical webpages found on the internet.
  - **Step 3: Score with the Well-Trained Classifier**
    - Assigns a quality score to webpages by how similar their content is to the Curated class.
  - **Step 4: Sample using Pareto Distribution**
    - Balances the inclusion of lower-quality pages to prevent bias:

Data Selection For LLM

Challenge: How to select high-quality pretraining datasets?

- Content-based Selection: Select high-quality data (e.g., data edited by humans; data from trustable sources like peer-reviewed articles)
- Perplexity-based Selection: Train an LLM and evaluate on the data to achieve higher selection performance
  - Sentence example:
    - "*I love machine learning*"
  - Calculate conditional probability
    - P(i)=0.2
    - P(love|i)=0.1
    - P(machine|i,love)=0.05
    - P(learning|i,love,machine)=0.01
    - M=4



A model with probability distribution ***P*** predicting a sequence of N words $w_1, w_2, \ldots, w_N$

$$PP(W) = 2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 P(w_i|w_1,\ldots,w_{i-1})}$$

closer to the true data distribution

Brown T, et al. Language models are few-shot learners[J]. NeurIPS, 2020, 33: 1877-1901.

Data Selection For LLM

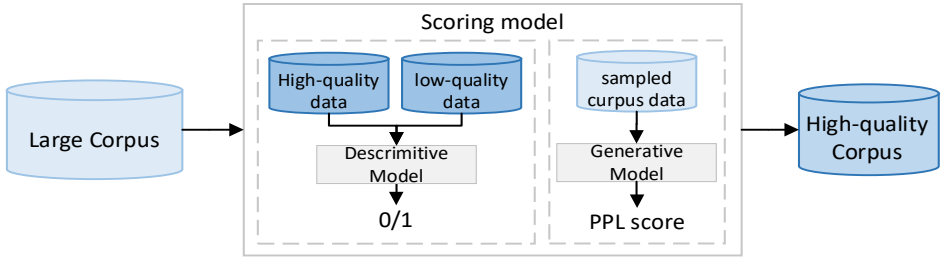
Challenge: How to select high-quality pretraining datasets?

- Content-based Selection: Select high-quality data (e.g., data edited by humans; data from trustable sources like peer-reviewed articles)
- Perplexity-based Selection:
  - Calculate the average value of logarithmic probabilities

$$\log P(\text{i}) = \log 0.2$$
$$\log P(\text{love}|\text{i}) = \log 0.1$$
$$\log P(\text{machine}|\text{i, love}) = \log 0.05$$
$$\log P(\text{learning}|\text{i, love, machine}) = \log 0.01$$
$$\frac{1}{4}(log0.2 + log0.1 + log0.05 + log0.01) \approx -2.8782$$

  - Calculate perplexity

$$Perplexity(P) = \exp(-(-2.8782)) \approx 17.77$$



A model with probability distribution ***P*** predicting a sequence of N words $w_1, w_2, \ldots, w_N$

$$PP(W) = 2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 P(w_i|w_1,\ldots,w_{i-1})}$$

quality sample

Brown T, et al. Language models are few-shot learners[J]. NeurIPS, 2020, 33: 1877-1901.

Data Selection For LLM

Challenge: How to select high-quality pretraining datasets?

- Content-based Selection: Select high-quality data (e.g., data edited by humans; data from trustable sources like peer-reviewed articles)
- Model-based Selection: Use Model to rate multiple documents along various dimensions of perceived quality → *Capture human intuitions about data quality*
  - **Quality Criteria:**
    - ***Writing style:*** *With polished or beautiful words*
    - ***Expertise:*** *The difficulty level of the corpus*
    - ***Facts & Trivia:*** *With high density of long-tail factual knowledge*
    - ***Educational value:*** *Includes clear explanations, step-by-step reasoning, or questions and answers*

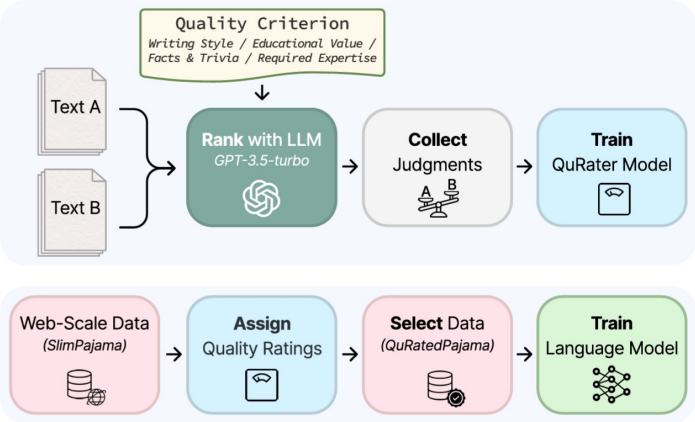
Data Selection For LLM

Challenge: How to select high-quality pretraining datasets?

- Content-based Selection: Select high-quality data (e.g., data edited by humans; data from trustable sources like peer-reviewed articles)
- Model-based Selection: Use Model to rate multiple documents along various dimensions of perceived quality
  - **1.** Sample text pairs (A, B) from a vast collection of documents
  - **2.** With the criteria and a pair (A, B), LLM (e.g., GPT3.5) gives a confidence of *B is better than A, i.e.,* $p_{B \succ A} \in [0, 1]$
  - **3.** Generate a dataset of judgement

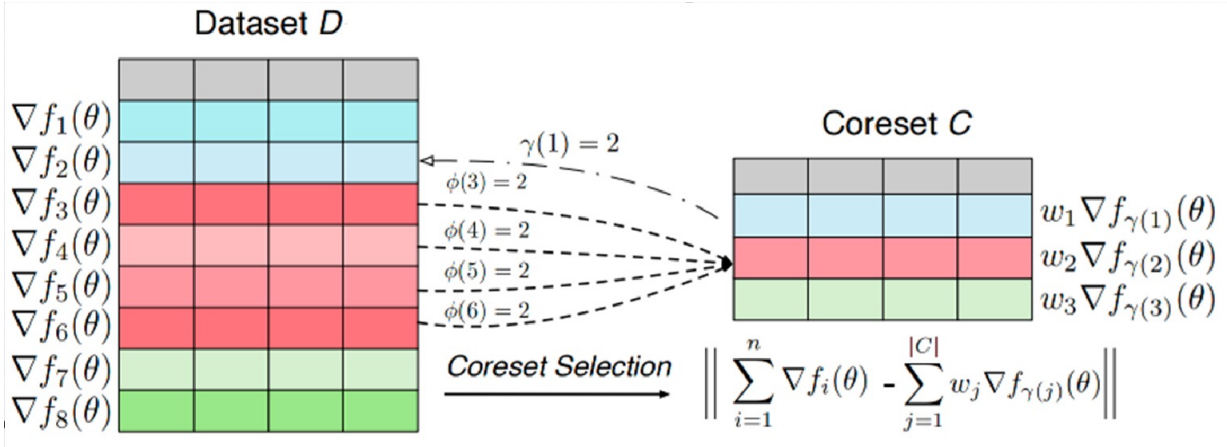
$$\mathcal{J} = \{(t_i, t_j, p_{i \succ j})\}$$

  - **4.** Fine-tune a 1.3B Sheared-Llama
    - Predict quality ratings under the four criteria



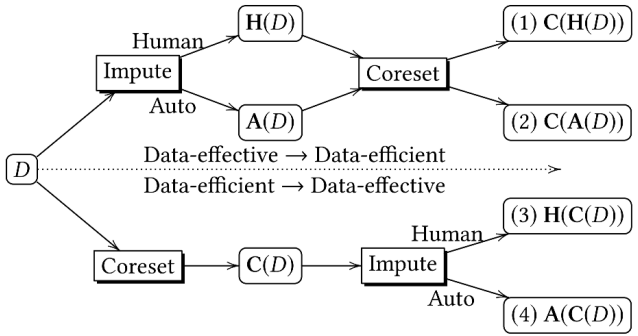
Gradient-based Data Selection

- **Algorithm: stochastic gradient decent**
- Data: Coreset
  - Given a large train set *D*, Coreset *C(D)* is a core subset of *D,* which is selected to represent *D* such that M(*C(D)*)≈M(*D*), denoting that *C(D)* has the same performance theoretically with *D.*



Gradient-based Data Selection

- **Intuitive baselines (sequential)**



- *First data-effective (impute) then data-efficient (coreset):*

  (1) Impute-Human: $\mathbf{H}(D) \rightarrow$ Coreset: $\mathbf{C}(\mathbf{H}(D))$
  (2) Impute-Auto: $\mathbf{A}(D) \rightarrow$ Coreset: $\mathbf{C}(\mathbf{A}(D))$

- *First data-efficient (coreset) then data-effective (impute):*

  (3) Coreset: $\mathbf{C}(D) \rightarrow$ Impute-Human: $\mathbf{H}(\mathbf{C}(D))$
  (4) Coreset: $\mathbf{C}(D) \rightarrow$ Auto-Human: $\mathbf{A}(\mathbf{C}(D))$

Solution	Accuracy	Human Cost	Machine Cost
(1) $\mathbf{C}(\mathbf{H}(D))$	High	High	Low
(2) $\mathbf{C}(\mathbf{A}(D))$	Low	None	Low
(3) $\mathbf{H}(\mathbf{C}(D))$	Low	Low	Low
(4) $\mathbf{A}(\mathbf{C}(D))$	Low	None	Low

Challenge

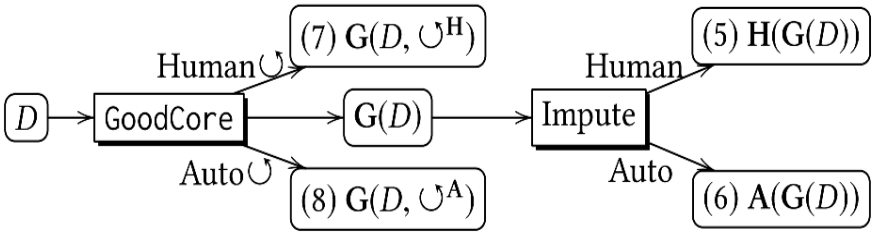
- Computing a good coreset from dirty data is to accurately estimate the ground truth of each missing value, which has multiple possible repairs.
- The combinations of all possible repairs constitute a huge search space.

Gradient-based Data Selection

Key idea of GoodCore

- Model the combinations of possible repairs as *possible worlds* of the original dirty data *D*
- Selecting *an expected optimal coreset* that can *represent the possible worlds of D* via gradient approximation without training in advance

D → G(D): Very Time-consuming



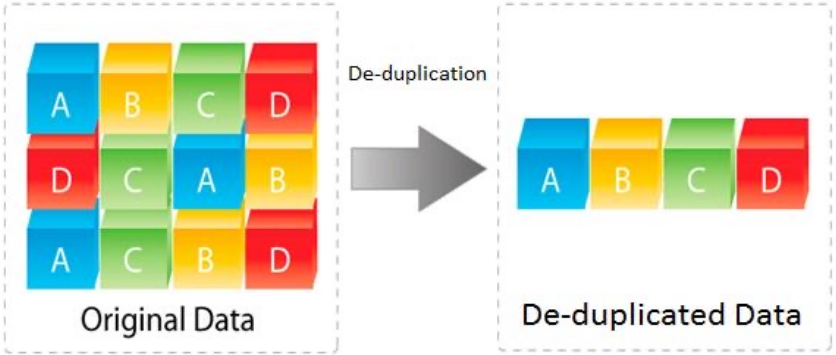
Solution	Accuracy	Human Cost	Machine Cost
(1) C(H(*D*))	High	High	Low
(2) C(A(*D*))	Low	None	Low
(3) H(C(*D*))	Low	Low	Low
(4) A(C(*D*))	Low	None	Low
Our goal	High	None or Low	Low
(5) H(G(*D*))	High	Low	High
(6) A(G(*D*))	Medium	None	High
(7) G(*D*, ↻H)	High	Low	Low
(8) G(*D*, ↻A)	Medium	None	Low

Data Cleaning For LLM

Data Preparation: Turn big dirty data into a subset of good data

- Data Cleaning: Remove duplicate records; Remove (noise) outliers; Resolve inconsistencies; Fill in missing values (generally not conducted in LLM)
- Data Deduplication: Training on identical documents slows down training and may harm the performance → Identify same/similar documents and retain one

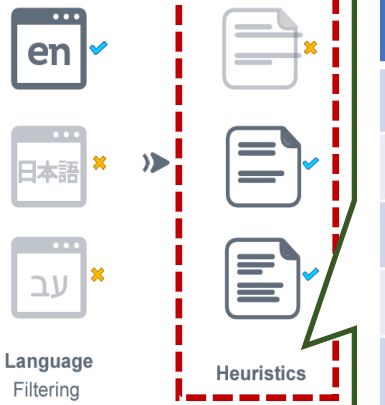
- **Exact Matching**: Leverage MD5 hashing to ensure documents are identical.
- **Near Matching**: Use min-hash/sim-hash to locate overlapped text, measured by jaccard similarity scores
- **Semantical Matching**: Clustering documents with pretrained embeddings



Data Cleaning For LLM

Challenge: How to select high-quality pretraining datasets?

- Rule-based Cleaning: Remove undesirable data with Heuristic Rules



Ensuring data is coherent, contextually rich, free of bias

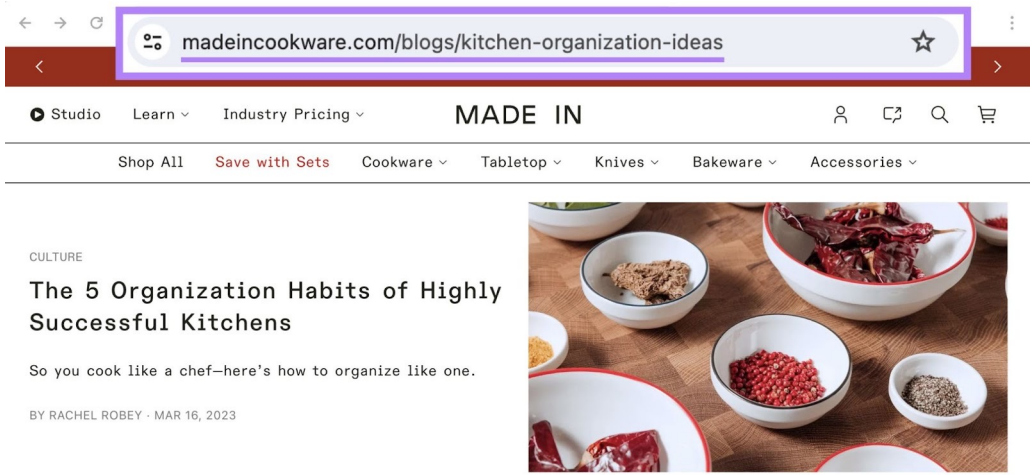
Goals	Heuristic Rules
Ensure Text Quality	**Word Count:** 50 - 100,000 words
Proper Word Length	**Mean Length:** 3 - 10 characters
Manage Symbol Use	**Symbol Ratio:** <0.1 for # and ...
Limit List Formatting	**List Control:** <90% bullets start, <30% ellipsis end
Require Alphabetic Words	**Alphabet Presence:** 80% of words
Filter Non-Coherent English	**Stop Words:** Must have at least two common words

Rae J W, Borgeaud S, Cai T, et al. Scaling language models: Methods, analysis & insights from training gopher[J]. arXiv preprint arXiv:2112.11446, 2021.

Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

- Data Deduplication: Remove duplicates to enhance training performance
- Exact Matching Techniques:
  - 1. URL Deduplication: Remove data that shares the same URL
    - Individual web pages may appear multiple times



Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

- Data Deduplication: Remove duplicates to enhance training performance
- Exact Matching Techniques:
  - 2. Hash Functions: Guarantee to find all exact matches

    (1) Initialize a Set for Hashes

    A set ~ The hashes of encountered text entries.

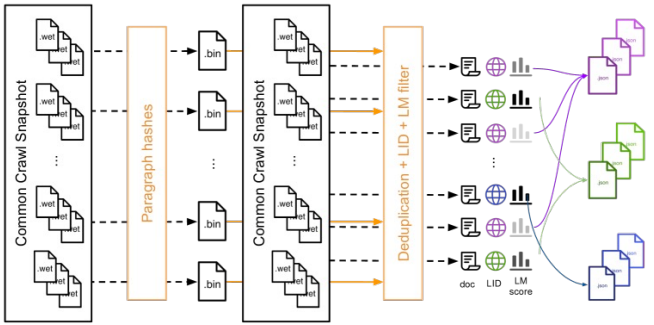
    (2) Hash Each Text Entry

    For each text entry, compute a simple hash (e.g., the sum of ASCII values of its characters).

    (3) Check for Duplicates

    If the hash of the current entry is already in the set, it is a duplicate and will be ignored.

    If the hash is not in the set, add the hash to the set and keep the entry.



Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

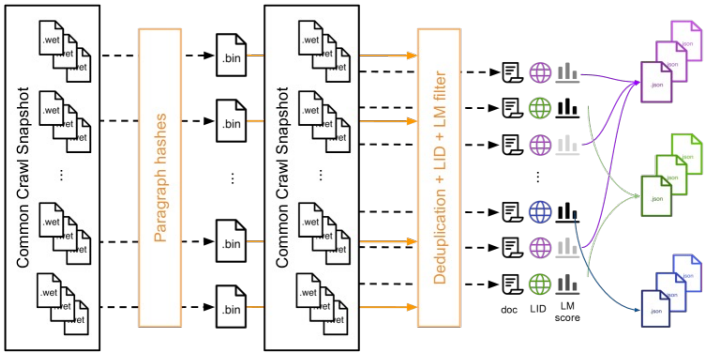
- Data Deduplication: Remove duplicates to enhance training performance
- Exact Matching Techniques:
  - 2. Hash Functions: Guarantee to find all exact matches

    (1) Initialize a Set for Hashes

    A set ~ The hashes of encountered text entries.

    (2) Hash Each Text Entry

    For each text entry, compute a simple hash (e.g., the sum of ASCII values of its characters).



    (3) Check for Duplicates

    If the hash of the current entry is already in the set, it is a duplicate and will be ignored.

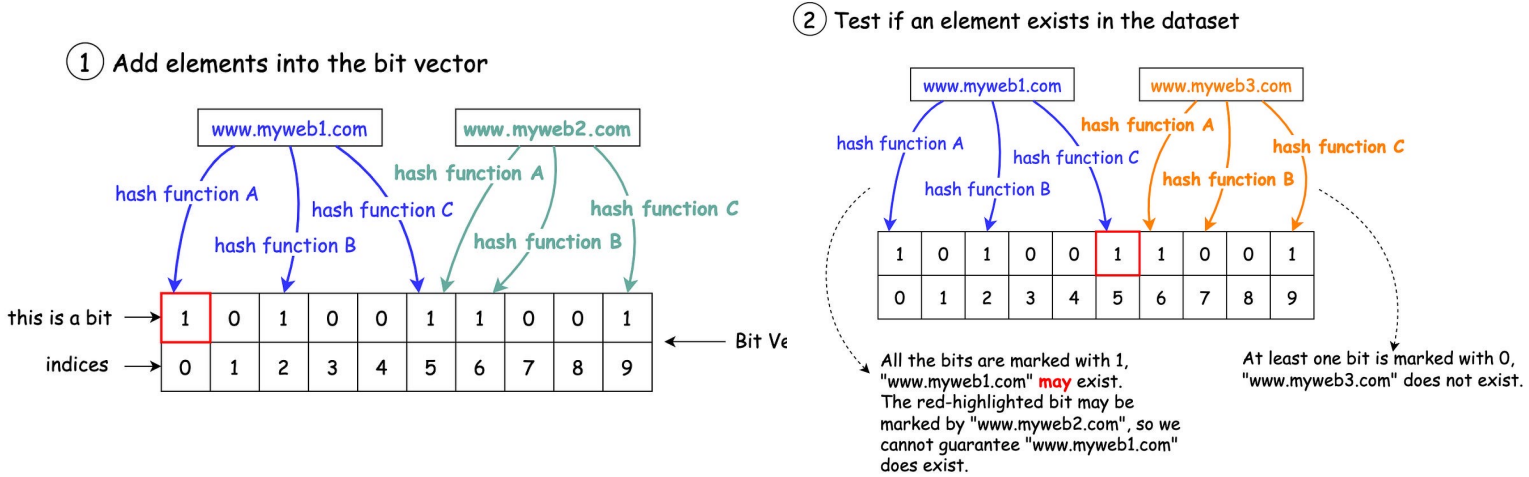
    If the hash is not in the set, add the hash to the set and keep the entry.

Efficient and Fast*, but *may find false positives due to hash collisions and remove non-matching documents

Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

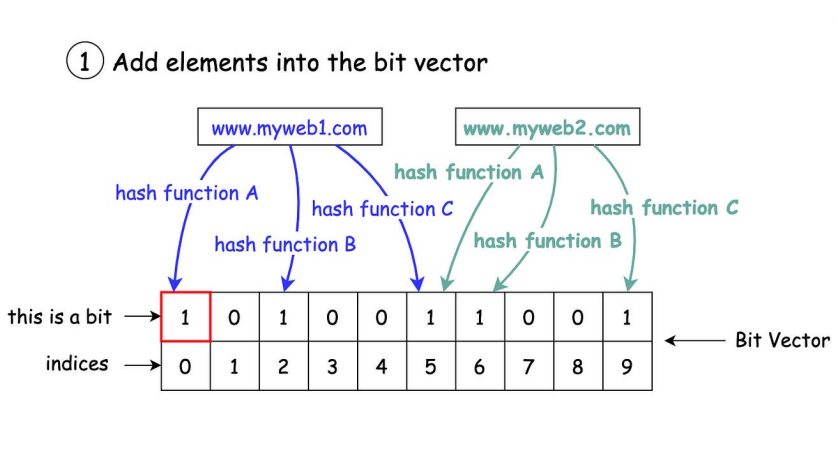
- Data Deduplication: Remove duplicates to enhance training or sometimes improve accuracy
- Exact Matching Techniques:
  - Bloom Filters: Space-efficient method using bit arrays for document comparison.

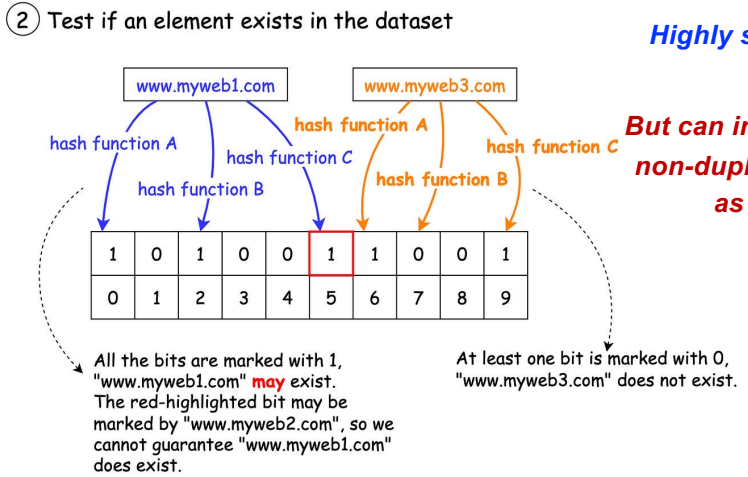


Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

- Data Deduplication: Remove duplicates to enhance training or sometimes improve accuracy
- Exact Matching Techniques:
  - Bloom Filters: Space-efficient method using bit arrays for document comparison.





Highly space-efficiency

But can incorrectly identify non-duplicate documents as duplicates

Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

- Data Deduplication: Remove duplicates to enhance training or sometimes improve accuracy
- Approximate Matching Techniques:
  - 1. String Metric Method
  - **S1**: Use MinHash to approximate the Jaccard Index:

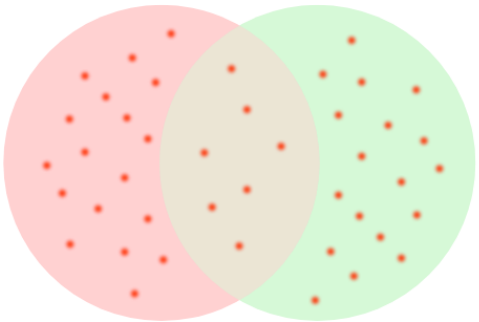
$$\text{Jaccard}(d_i, d_j) = |d_i \cap d_j| / |d_i \cup d_j|$$

    - d_i: The n-grams of document I
    - High Jaccard Index indicates high text similarity

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

total elements in intersection

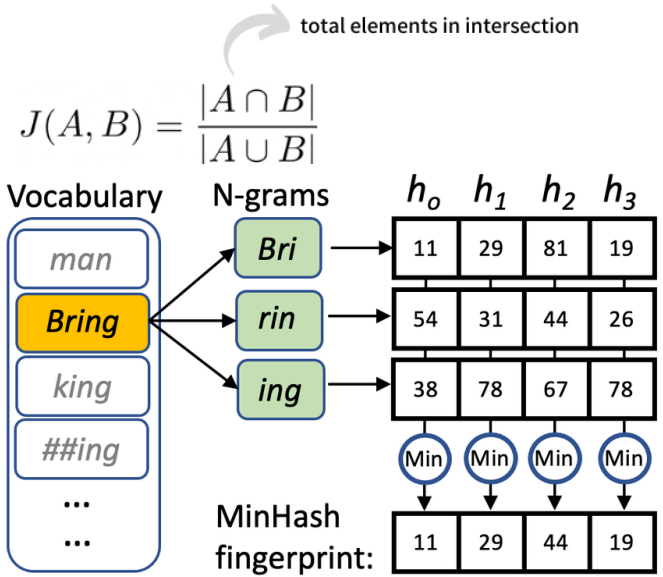
total elements in union i.e. Universal Set



Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

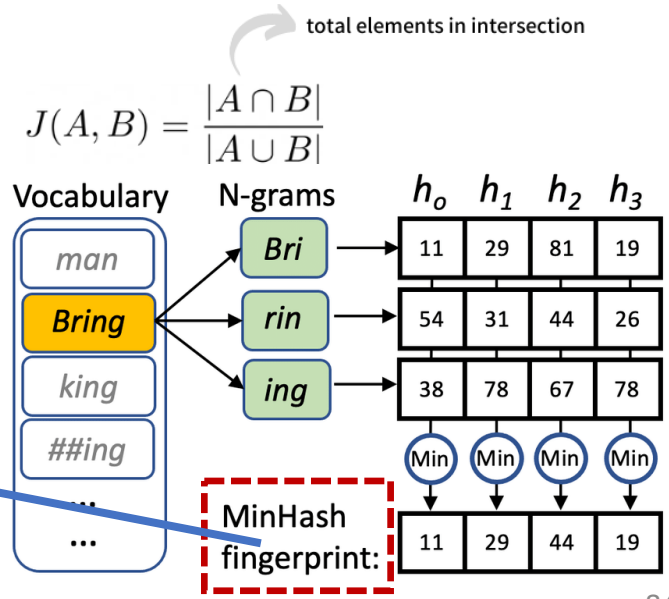
- Data Deduplication: Remove duplicates to enhance training or sometimes improve accuracy
- Approximate Matching Techniques:
  - 1. String Metric Method
  - **S1**: Use MinHash to approximate the Jaccard Index:
  - **MinHash:** Construct document signatures by sorting each n-gram via a hash function; Then keep only the k smallest hashed n-grams.



Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

- Data Deduplication: Remove duplicates to enhance training or sometimes improve accuracy
- Approximate Matching Techniques:
  - 1. String Metric Method
  - **S1**: Use MinHash to approximate the Jaccard Index:
  - **MinHash:** Construct document signatures by sorting each n-gram via a hash function; Then keep only the k smallest hashed n-grams.



- *These **MinHash fingerprints** are then partitioned into r bucket (with b hashes per bucket).*
- *In each bucket, the b hashes are augmented into one value.*
  - *If two documents have the same value in at least one bucket, they'll be marked as a **potential match**.*

Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

- Data Deduplication: Remove duplicates to enhance training or sometimes improve accuracy
- Approximate Matching Techniques:
  - 1. String Metric Method
  - **S1**: Use MinHash to approximate the Jaccard Index: $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$ (total elements in intersection)
  - **MinHash:** Construct document signatures by sorting each n-gram via a hash function; Then keep only the k smallest hashed n-grams.
  - **S2:** For each "potentially similar" pair, compute edit similarity

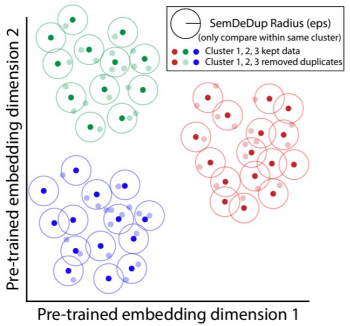
$$\text{EditSim}(x_i, x_j) = 1 - \frac{\text{EditDistance}(x_i, x_j)}{\max(|x_i|, |x_j|)}$$

    - E.g., two documents are similar if EditSim is greater than 0.8

Data Cleaning For LLM

Motivation: Pretraining prefers to remove duplicates, ensuring greater coverage with less redundancy

- Data Deduplication: Remove duplicates to enhance training or sometimes improve accuracy
- Approximate Matching Techniques:
  - 2. Model-based Method: Use pretrained models for semantic deduplication
  - **S1:** Leverage embedding spaces created by pre-trained LLM, providing a semantically meaningful distance metric for identifying duplicates
  - **S2:** Each data point is embedded using the LLM
  - **S3:** The embedded data points are clustered using k-means
  - **S4:** Within each cluster, pairwise cosine similarities between data points are calculated.
  - **S5:** For identified duplicates within a cluster, only the point with the lowest cosine similarity to the cluster centroid is kept, and the others are removed.

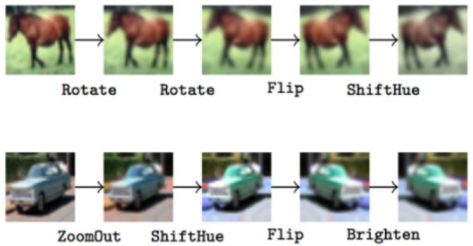


Data Augmentation For LLM

Data Preparation: Turn big dirty data into a subset of good data

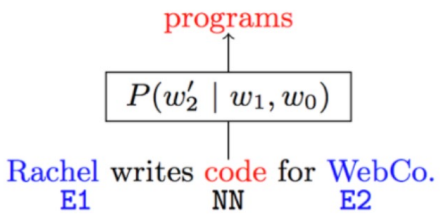
- Data Augmentation: Find auxiliary data which most resembles the distribution of desired data distribution (e.g., medicine or law).

Images



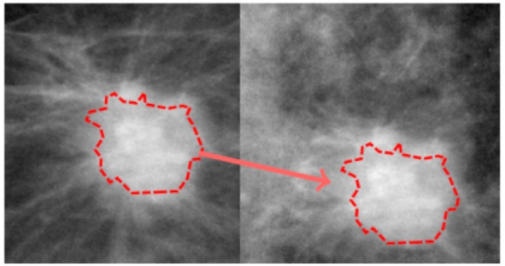
- Rotations
- Scaling / Zooms
- Brightness
- Color Shifts
- Etc…

Text



- Synonymy
- Positional Swaps
- Etc…

Medical



Domain-specific transformations.
Ex:
1. *Segment tumor mass*
2. *Move*
3. *Resample background tissue*
4. *Blend*

Data Augmentation For LLM

Challenge: How to select high-quality pretraining datasets?

- Data Augmentation: The goal is to find the **auxiliary data** which most resembles the distribution of in-domain data.
- Domain-Specific Selection: Let ***I*** be in-domain dataset, ***N*** be general purpose dataset, N_I be a subset of *N* that is in-domain that we wish to discover. The probability of "a data point *x(i)* drawn randomly from *N* being in N_I" is:

Moore-Lewis selection

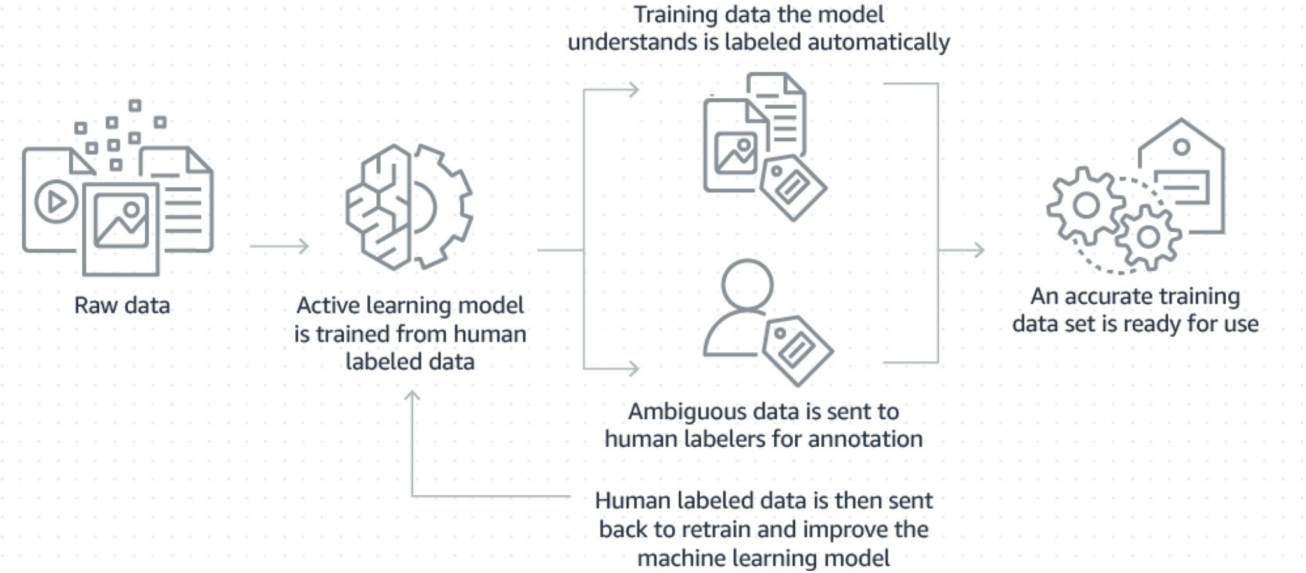
$$P(N_I|x^{(i)}, N) = \frac{P(x^{(i)}|I)P(N_I|N)}{P(x^{(i)}|N)}, \quad \frac{P(x^{(i)}|I)}{P(x^{(i)}|N)} \propto \frac{P(x^{(i)}|I)P(N_I|N)}{P(x^{(i)}|N)}$$

- Train models to estimate for $P(x^{(i)}|I)$ and $P(x^{(i)}|N)$ on *I* and a sample of *N*
- $\frac{P(x^{(i)}|I)}{P(x^{(i)}|N)}$ is approximated by $\log(P(x^{(i)}|I)) - \log(P(x^{(i)}|N))$, i.e., the cross-entropy loss from models trained on *I* and *N*.

Data Labeling For LLM

Data Preparation: Turn big dirty data into a subset of good data

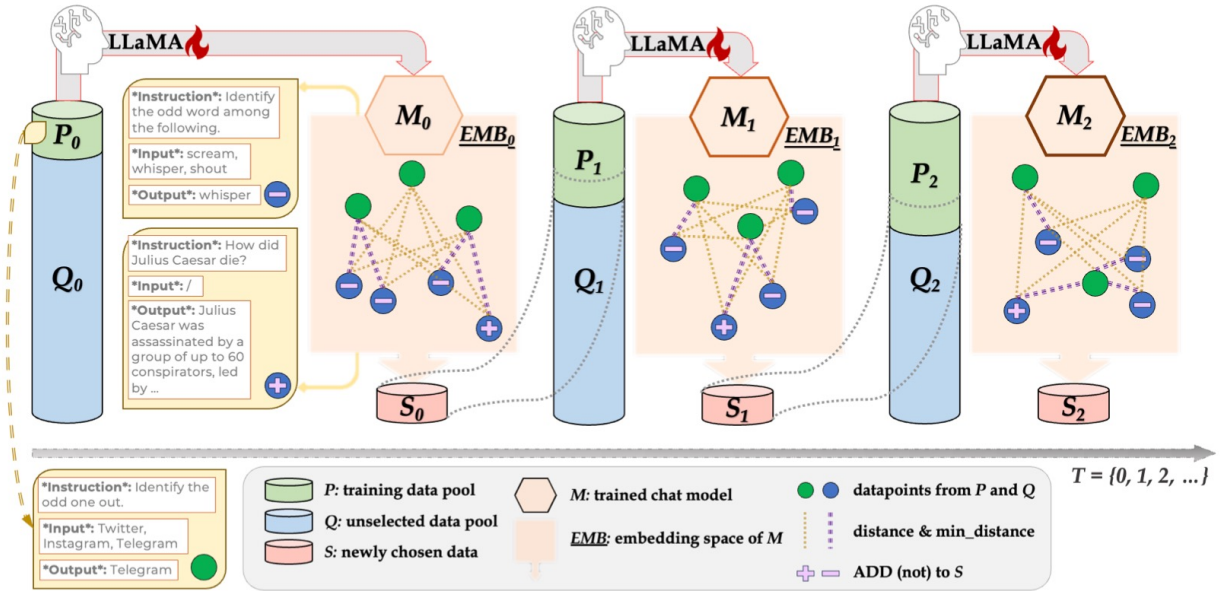
- Data Labeling: Annotate or tag raw data (images, texts, videos, etc) with meaningful information to provide context for LLM to learn
- A properly labeled dataset is **"Ground Truth"** in model training and assessing



Distance-based Data Labeling

Motivation: No intervention from humans or more advanced LLMs

- Use the model itself to iteratively guide data selection



Target: *Enhance the instruction-following capabilities*

Core idea: *Select new data points most distinct from existing ones*

Wu, Shengguang, et al. "Self-evolved diverse data sampling for efficient instruction tuning." arXiv preprint arXiv:2311.08182 (2023).

Data Mixing For LLM

Data Preparation: Turn big dirty data into a subset of good data

- Data Mixing: Data mixing optimizes the **weighting of different data domains** in training corpora to enhance model training efficiency and performance.

Component	Raw Size	Weight	Epochs	Effective Size	Mean Document Size
Pile-CC	227.12 GiB	18.11%	1.0	227.12 GiB	4.33 KiB
PubMed Central	90.27 GiB	14.40%	2.0	180.55 GiB	30.55 KiB
Books3†	100.96 GiB	12.07%	1.5	151.44 GiB	538.36 KiB
OpenWebText2	62.77 GiB	10.01%	2.0	125.54 GiB	3.85 KiB
ArXiv	56.21 GiB	8.96%	2.0	112.42 GiB	46.61 KiB
Github	95.16 GiB	7.59%	1.0	95.16 GiB	5.25 KiB
YoutubeSubtitles	3.73 GiB	0.60%	2.0	7.47 GiB	22.55 KiB
PhilPapers	2.38 GiB	0.38%	2.0	4.76 GiB	73.37 KiB
NIH ExPorter	1.89 GiB	0.30%	2.0	3.79 GiB	2.11 KiB
Enron Emails†	0.88 GiB	0.14%	2.0	1.76 GiB	1.78 KiB
The Pile	**825.18 GiB**			**1254.20 GiB**	**5.91 KiB**

Table 1: Overview of datasets in the Pile before creating the held out sets. Raw Size is the size before any up- or down-sampling. Weight is the percentage of bytes in the final dataset occupied by each dataset. Epochs is the number of passes over each constituent dataset during a full epoch over the Pile. Effective Size is the approximate number of bytes in the Pile occupied by each dataset. Datasets marked with a † are used with minimal preprocessing from prior work.

Data Mixing For LLM

Challenge: How to select high-quality pretraining datasets?

- Data Mixing: Determine the optimal domain ratios to improve the training efficiency and model performance
- Empirical-Determined Method
  - **Rule 1:** Prevent small sources (e.g., MultiUN) from oversampled;
  - **Rule 2:** Large proportion of code (e.g., 50%) does not harm to NL performance, and can benefit reasoning-based tasks;
  - **Rule 3:** Test different combinations over small-sized LLMs like 1B parameters.

Github	Microsoft	2008-4	-	All
mC4	Google Research	2021-6	251 GB	All
MNBVC	Liwu Community	2023-1	20811 GB	All
MTP	BAAI	2023-9	1.3 TB	All
MultiUN	German Research Center for Artificial Intelligence (DFKI) GmbH	2010-5	4353 MB	All
News-crawl	UKRI et al.	2019-1	110 GB	All

Nan Du, et al. GLaM: Efficient scaling of language models with mixture-of-experts . ICML, 2022.

Data Mixing For LLM

Challenge: How to select high-quality pretraining datasets?

- Data Mixing: Determine the optimal domain ratios to improve the training efficiency and model performance
- Model-Determined Method: Optimize the ratios assigned to different domains in training a model without relying on downstream tasks
  - Optimize domain ratios using a small proxy model

$$\min_{\theta} \max_{g \in \mathcal{G}} \mathbb{E}_{(x,y)\sim\mathcal{D}_g} \left[\ell\left(f_\theta(x), y\right) \right]$$

- θ: Model parameters
- g: Group/domain
- $\mathcal{D}_g$: Data distribution for group g
- ℓ: Loss function

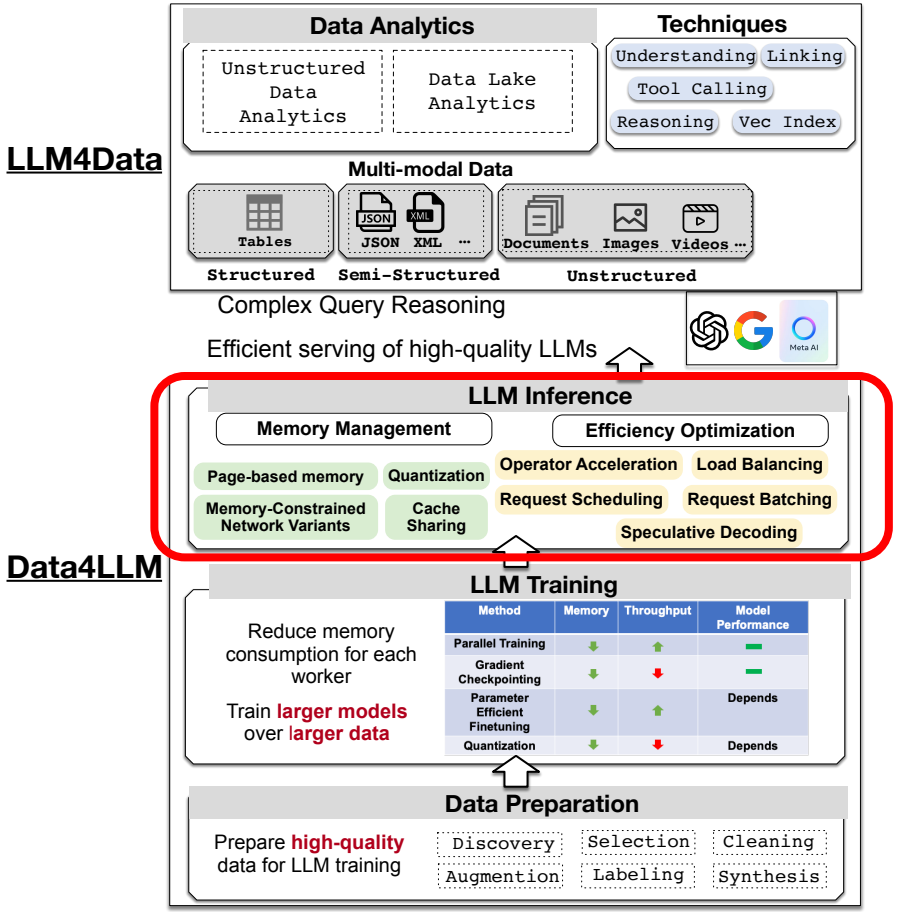
Minimize the maximum loss across all domains

  - Train a larger model using the optimized domain ratios

Sang Michael Xie, Hieu Pham , et al. Doremi: Optimizing data mixtures speeds up language model pretraining . NeurIPS, 2023.

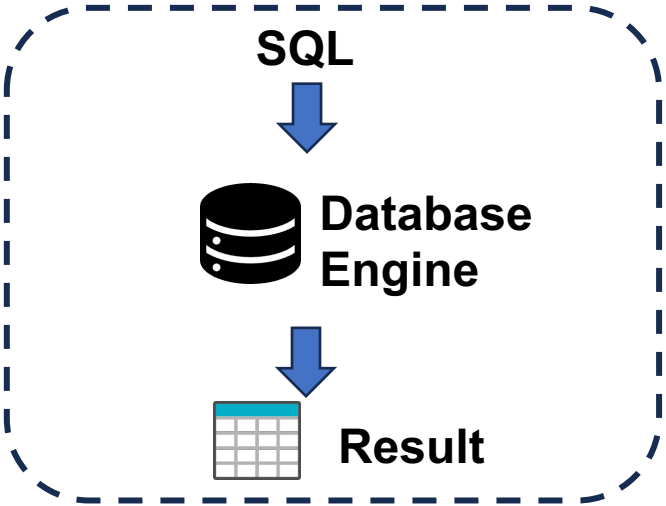
Data4LLM

- ❑ **Data Management tasks**
- ❑ **LLM4Data Techniques**
  - LLM Prompting
  - RAG & Vector DB
  - Data Agents
    - Unstructured Data Analytics
    - SQL + Semantics
    - Data Lake Analytics
- ❑ **Data4LLM Techniques**
  - Data Preparation
  - **LLM Inference**
  - LLM Training
- ❑ **Open Challenges**



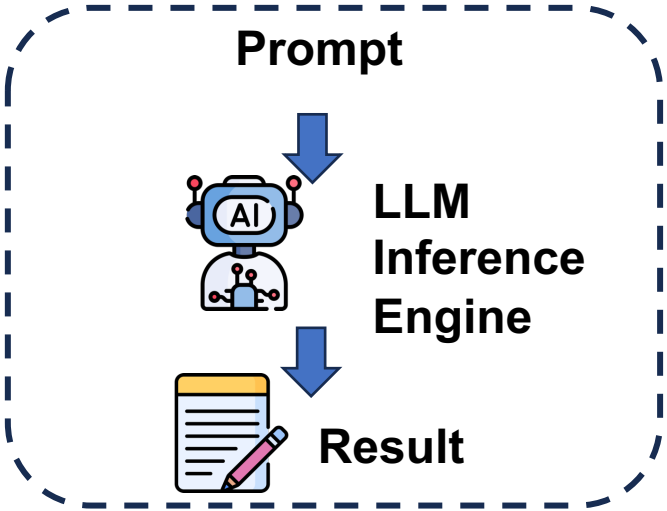
DB Query Processing vs LLM Inference

DB Query Processing



- Goal
  - Minimize **latency**
  - Maximize **throughput**

LLM Inference



- Goal
  - Minimize **latency**
  - Maximize **throughput**

LLM inference has the same goal as DB query processing

How to Reduce LLM Inference Latency and Improve Throughput?

Q1: How to reduce latency for a **single query** on **one GPU**?

- KV cache
- Quantization
- Memory-optimized model
- Speculation

Q2: How to optimize throughput for **multiple queries** on **one GPU**?

- Page-based memory allocation
- Cache persistence and sharing
- KV cache eviction/offloading
- Request batching
- Request scheduling

Q3: How to optimize throughput for **multiple queries** on **multiple GPUs**?

- Load balancing
- Disaggregated prefilling and decoding

How to Reduce LLM Inference Latency and Improve Throughput?

Q1: How to reduce latency for a **single query** on **one GPU**?

- KV cache
- Quantization
- Memory-optimized model
- Speculation

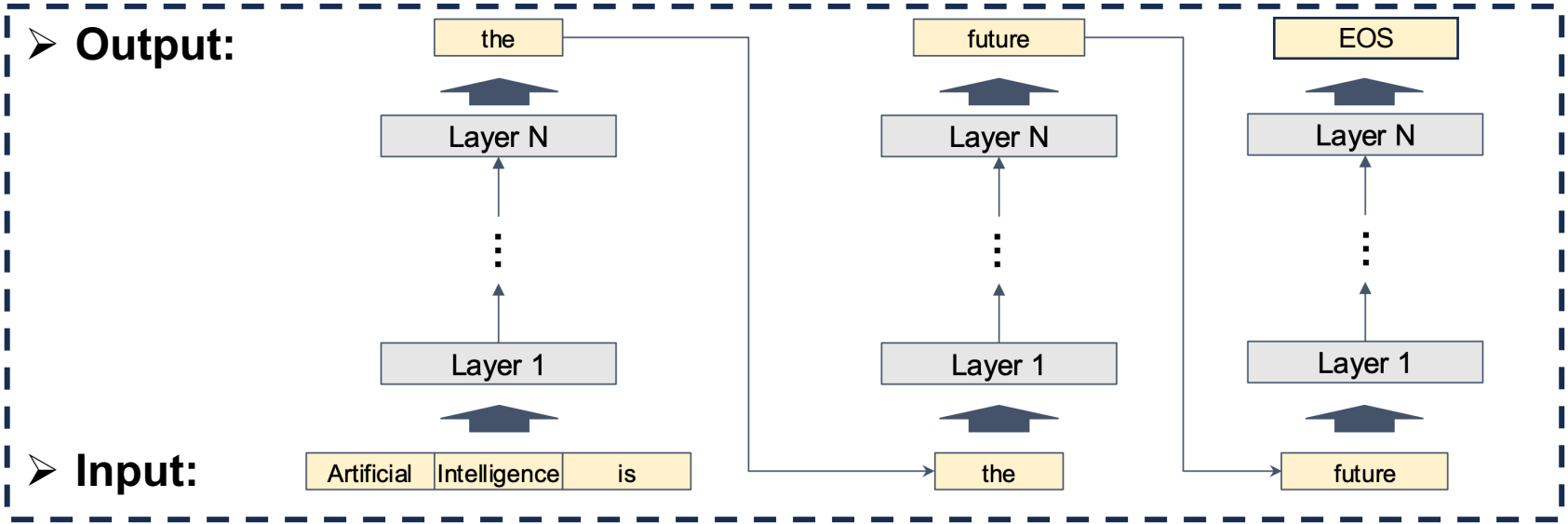
Q2: How to optimize throughput for **multiple queries** on **one GPU**?

- Page-based memory allocation
- Cache persistence and sharing
- KV cache eviction/offloading
- Request batching
- Request scheduling

Q3: How to optimize throughput for **multiple queries** on **multiple GPUs**?

- Load balancing
- Disaggregated prefilling and decoding

Background: LLM Inference Process



For each LLM request

- **Input:** a text string (prompt)
- **Output:** a text string with **non-deterministic** length

Predict next token until it

- Generates certain ending tokens
- Reaches its pre-defined maximum length

Background: LLM Inference Process

- A request consists of an initial input (called prompt or prefix)

$$x_1, \cdots, x_p$$

- The response is a completed sequence

$$x_1, \cdots, x_p, \cdots, x_n$$

- For each $i \geq p$, it requires one execution of the model over all previous tokens

$$x_{i+1} = LLM(x_1, \cdots, x_i)$$

The output sequence is formed one token at a time by feeding previous tokens

Ashish Vaswani et al. Attention Is All You Need. NeurIPS 2017

Background: LLM Request Processing Process Zoom-in

- **Attention Computation**

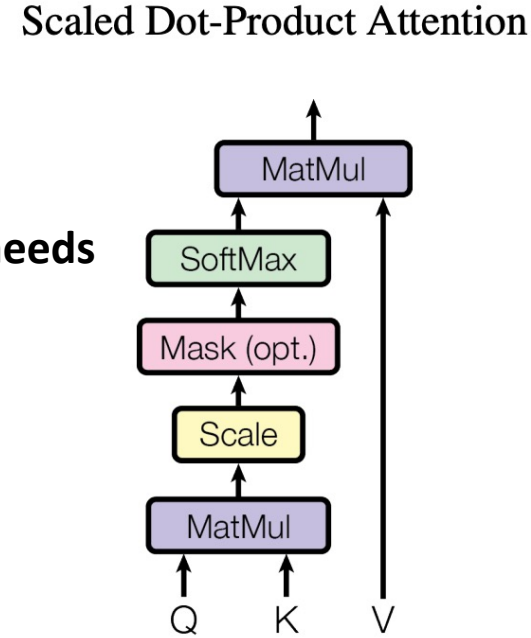
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

- **To compute** $x_{i+1} = LLM(x_1, \cdots, x_i)$ **, it needs**

$$K_j = X_j W^K$$

for all $1 \leq j \leq i$

$$V_j = X_j W^V$$

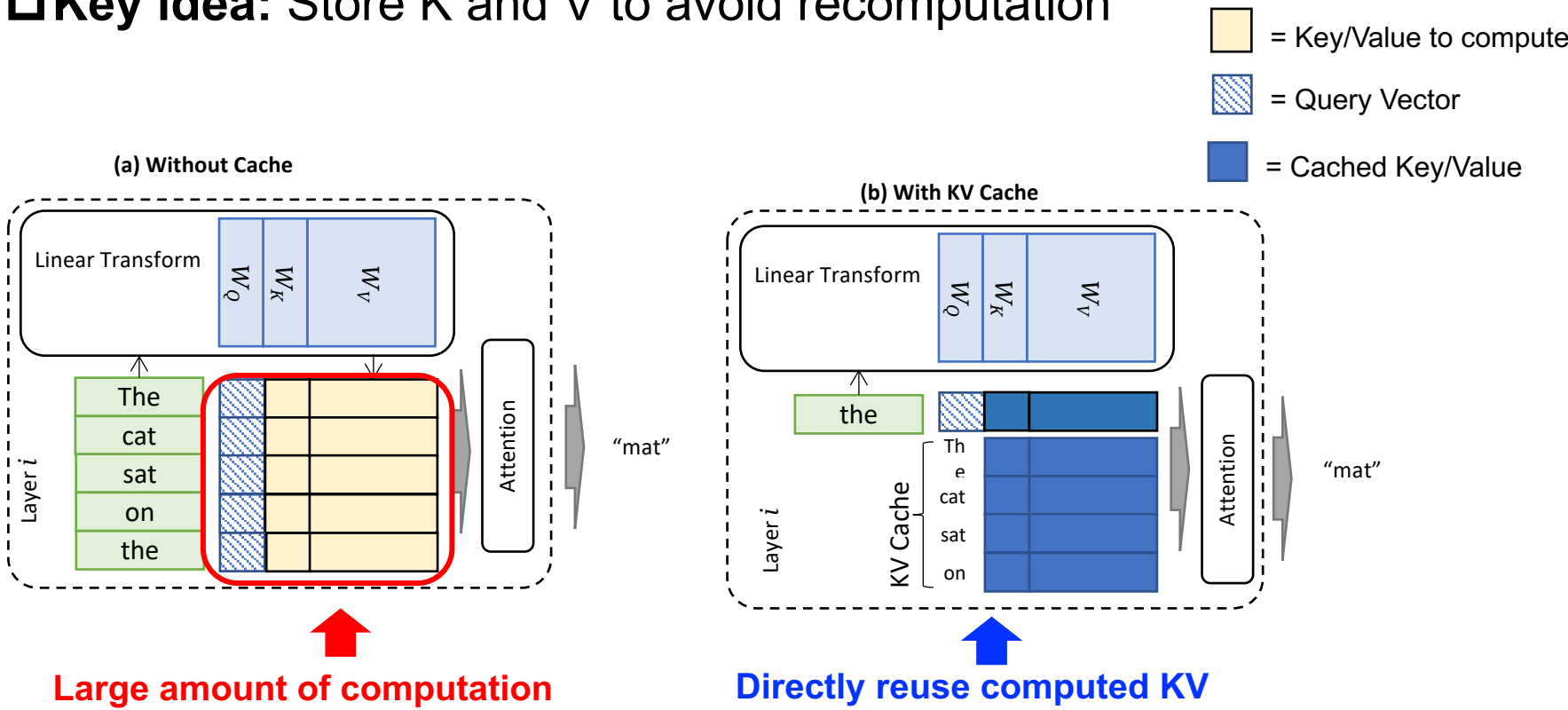


Expensive to recompute all K and V for generating each x_{i+1}

Ashish Vaswani et al. Attention Is All You Need. NeurIPS 2017

Use KV Cache to Avoid Recomputation

Key idea: Store K and V to avoid recomputation



Use KV Cache to Avoid Recomputation

- **Key idea:** Store K and V to avoid re-computation
  - **Pre-filling (Compute bound)**
    - Process all input tokens at once
    - Compute K and V for all input tokens in the prompt
  - **Decoding (Memory bound)**
    - Generate a single token based on previous tokens
    - Compute Q for current status
    - After generating the new token, add its K and V to KV cache

Limitation: Can result in large memory consumption if the sequence is very long

See solutions in later slides

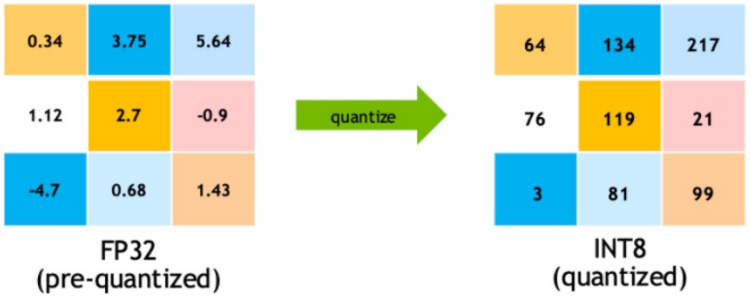
Zhou Z, Ning X, Hong K, et al. A survey on efficient inference for large language models[J]. arXiv preprint arXiv:2404.14294, 2024.

Quantization Techniques for Model Compression

☐**Key idea:** Lower the numerical precision to enable compact data formats

☐**Can reduce the physical byte sizes of:**

- Weight matrices
- Embedding vectors
- Intermediate activations
- Cache entries



☐**GPUs perform better when processing data with smaller bit widths:**

- E.g., on NVIDIA's A6000 GPU
  - 155 TOPS/s for **FP16**
  - 310 TOPS/s for **INT8**
- Speed up general matrix multiplication

Limitation: Quantization may influence model quality

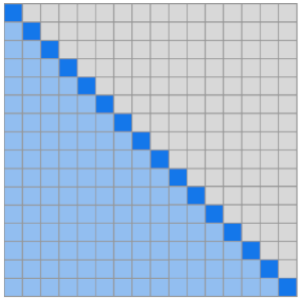
Yuan Z, Shang Y, Zhou Y, et al. Llm inference unveiled: Survey and roofline model insights[J]. arXiv preprint arXiv:2402.16363, 2024.

Optimized Model Structure – Sparse Attention

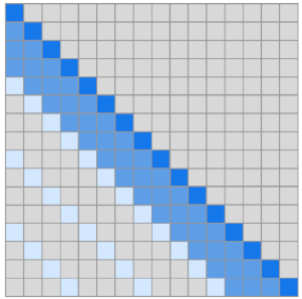
Key idea: Omit certain attention calculations

Method:

- Compute the attention status only for **certain tokens**
- Discover these significant keys through:
  - Static filtering (e.g., windowed, strided)
  - Query-dependent masks (e.g., learning-based)
  - K-nearest neighbor search indexes



Basic Attention



Sparse Attention

Limitation: Hurt inference accuracy

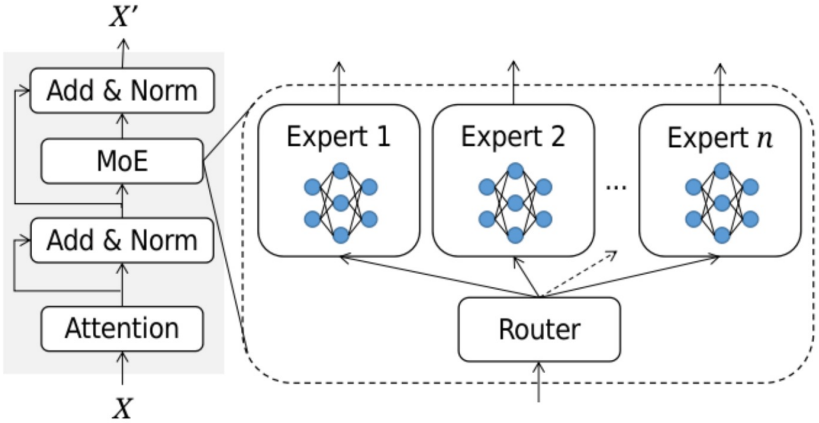
Child R, Gray S, Radford A, et al. Generating long sequences with sparse transformers[J]. arXiv preprint arXiv:1904.10509, 2019.

Optimized Model Structure – Mixture of Experts

Key idea: Allocate varying computation budgets to different tokens

Method:

- Replace network with **a set of smaller networks (experts)**
- During inference**, selectively activates specific experts** controlled by router
- Since each expert is much smaller than the original network, compute cost can be substantially reduced



Limitations:

- Routing Instability
- Load Imbalance

Shazeer N, Mirhoseini A, Maziarz K, et al. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer ICLR, 2017.

Speculative Decoding

- **Key idea:** use a smaller, faster model to generate draft tokens that are then verified **in parallel** by the LLM

- **Example:**
  - A landmark in Paris is the Eiffel **[Tower]**

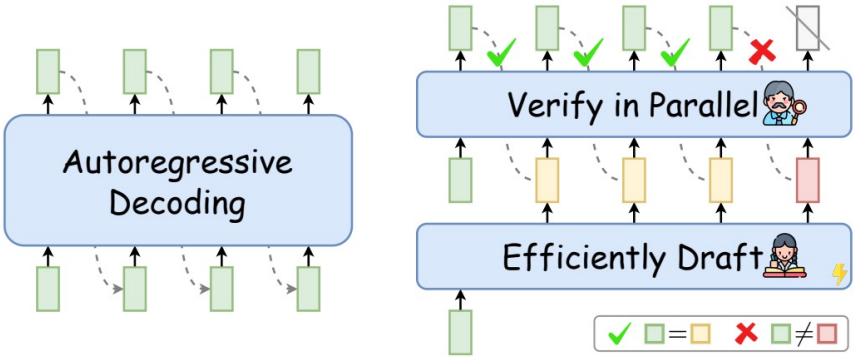
Can be accurately predicted by a small model

- **How to leverage cheap models to accelerate decoding?**

Leviathan Y, Kalman M, Matias Y. Fast inference from transformers via speculative decoding[C]//International Conference on Machine Learning. PMLR, 2023

Speculative Decoding

- **Key idea:** use a smaller, faster model to generate draft tokens that are then verified **in parallel** by the LLM
  - **Method:**
    1. Approximate the next b tokens using a small language model
    2. Verify drafts by LLM **in parallel**
    3. Accept verified tokens and Iteratively repeat above process until reaching end of sequence



Limitation: Incur redundant computation and low-quality draft model may not be accurate

Xia H, Yang Z, Dong Q, et al. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding[J]. arXiv, 2024.

Takeaways

Q1: How to reduce latency for a **single query** on **one GPU**?

KV Cache

- **Pros:** Avoid recomputation, thus more efficient
- **Cons:** Increased memory usage for multiple queries

Quantization

- **Pros:** Higher efficiency, less memory consumption
- **Cons:** Influence model quality

Memory-optimized model

- **Pros:** Higher efficiency, less memory consumption
- **Cons:** Influence model quality

Speculation

- **Pros:** May bring lower latency by parallel token generation
- **Cons:** Incur redundant computation

How to Reduce LLM Inference Latency and Improve Throughput?

Q1: How to reduce latency for a **single query** on **one GPU**?

- KV cache
- Quantization
- Memory-optimized model
- Speculation

Q2: How to optimize throughput for **multiple queries** on **one GPU**?

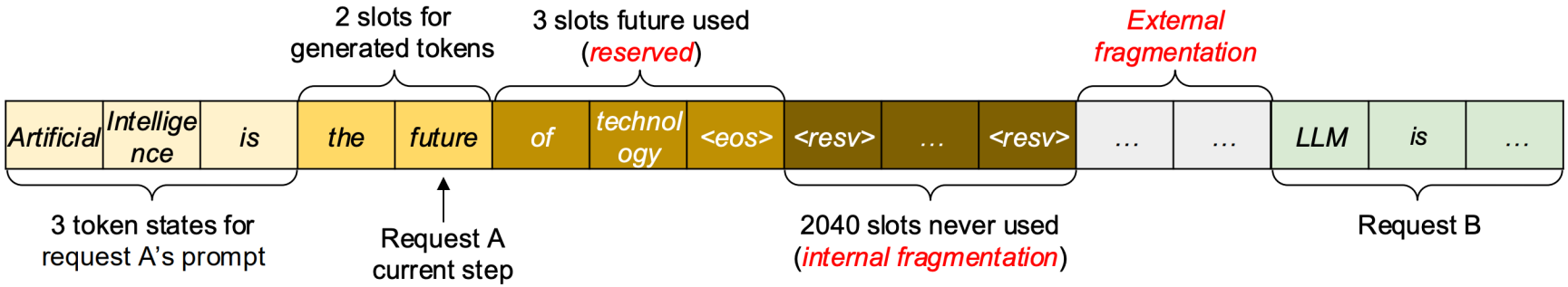
- Page-based memory allocation
- Cache persistence and sharing
- KV cache eviction/offloading
- Request batching
- Request scheduling

Q3: How to optimize throughput for **multiple queries** on **multiple GPUs**?

- Load balancing
- Disaggregated prefilling and decoding

Page-based Memory Allocation

Motivation

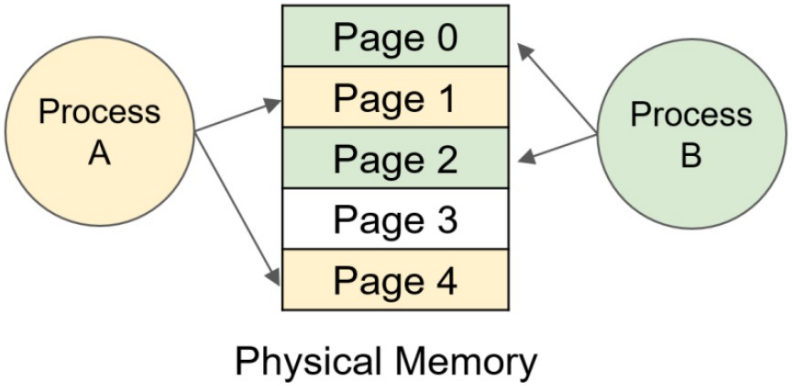


Wasted Memory:

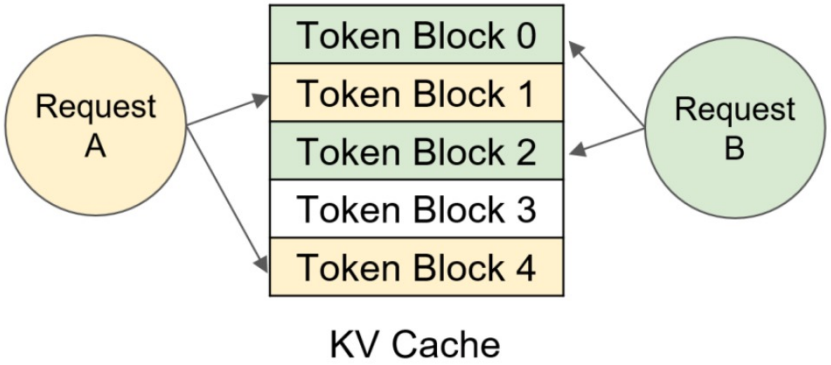
- **Reservation:** not being used now, but can actually be used by short requests
- **Internal fragmentation:** over-allocated due to the unknown output length
- **External fragmentation**: gap between memory regions allocated to different queries

Page-based KV Cache Memory Allocation

Key idea: Divide memory into blocks similar to virtual memory and paging in OS, and allocate in this granularity



Page-based memory management in OS



Page-based memory management in LLM serving

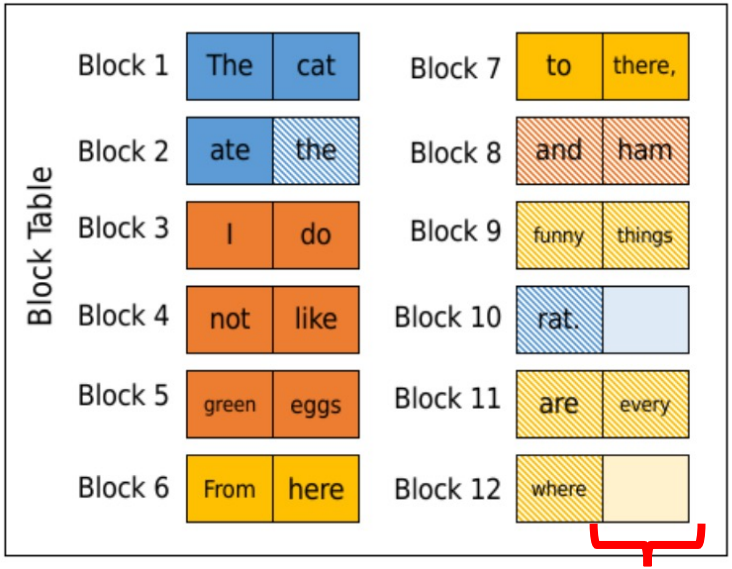
Page-based KV Cache Memory Allocation

- **Token Block:** Each token block is a fixed-size contiguous chunk of memory that can store token states from left to right

- **Ensures bounded internal fragmentation**
  - Only happens at the last block of a sequence
  - The wasted memory of a single query is bounded by block size

- **Eliminate external fragmentation**

Limitation: Requires rewriting attention kernels



KV Cache Eviction/Offloading for Multiple Queries

- **Key idea:** Make room by evicting non-critical cache
  - **Eviction:** Need recomputation to recover
  - **Offloading:** Can be tranferred back to GPU from other memory containers (e.g. CPU)

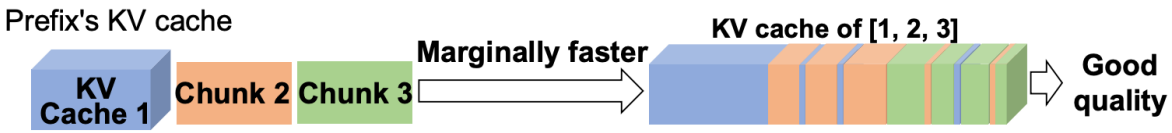
- **Strategies:**
  - Least recently used
  - Least frequently used
  - All-or-nothing (vLLM)

Limitation: May hurt latency for each single query due to the cost of cache recovery

Qin R, Li Z, He W, et al. Mooncake: A kvcache-centric disaggregated architecture for llm serving[J]. arXiv preprint arXiv:2407.00079, 2024.

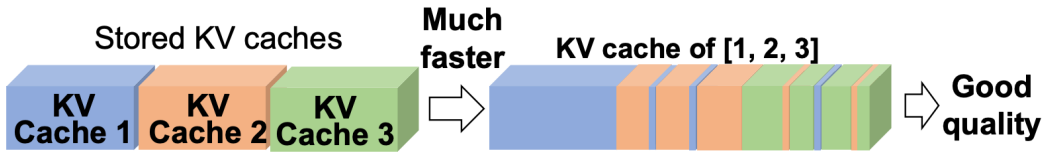
Cache Sharing for Improving Efficiency

- **Key idea:** Reuse computed results of previous requests
- **Prefix Sharing:**
  - Reuse persisted cache entries under exact-match prefixes
  - Can only reuse ***prefix's*** KV cache, since prefix matching requirement is strict



Too strong requirement

- **Selective Reconstruction:**
  - Reuse all KV cache but re-computing a small fraction of KV
  - Mitigate quality degradation by recomputing KV for a subset of impactful tokens

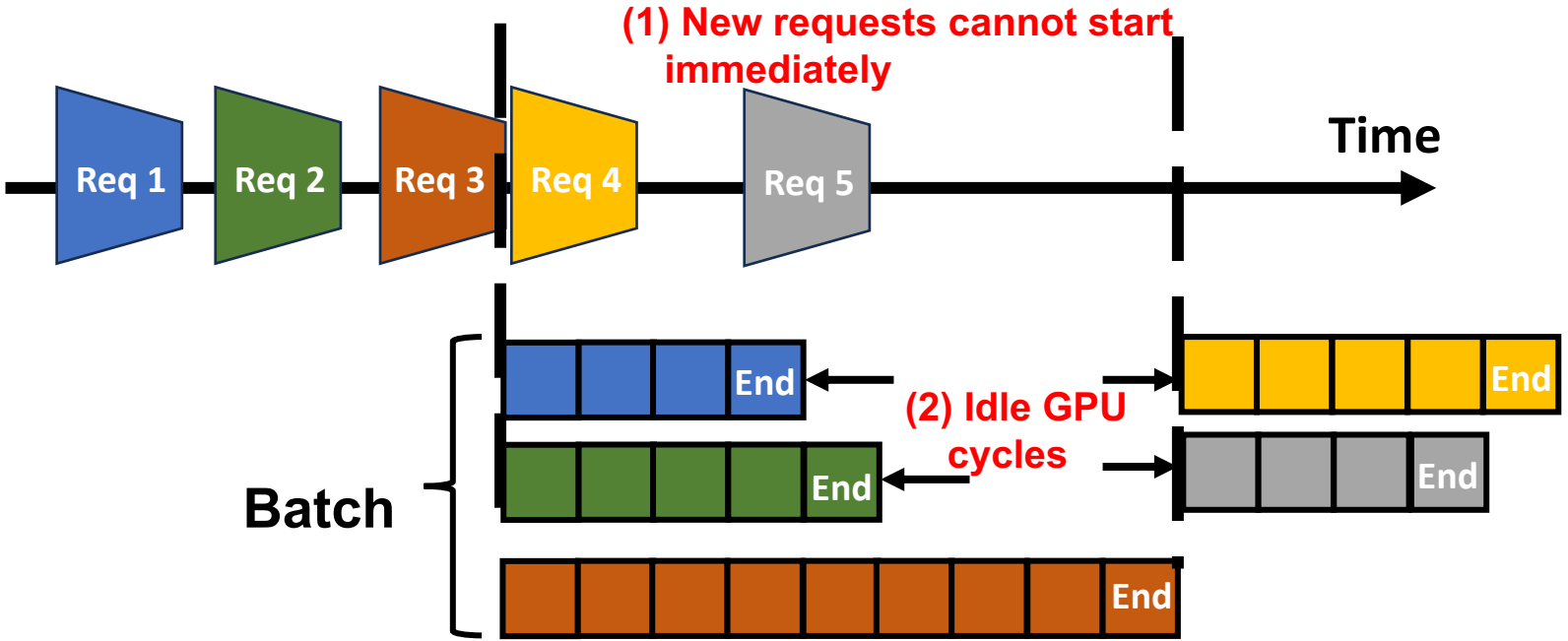


May hurt accuracy

Yao J, et al. CacheBlend: Fast large language model serving for RAG with cached knowledge fusion, Eurosys 2025

LLM Request Batching – Static Batching

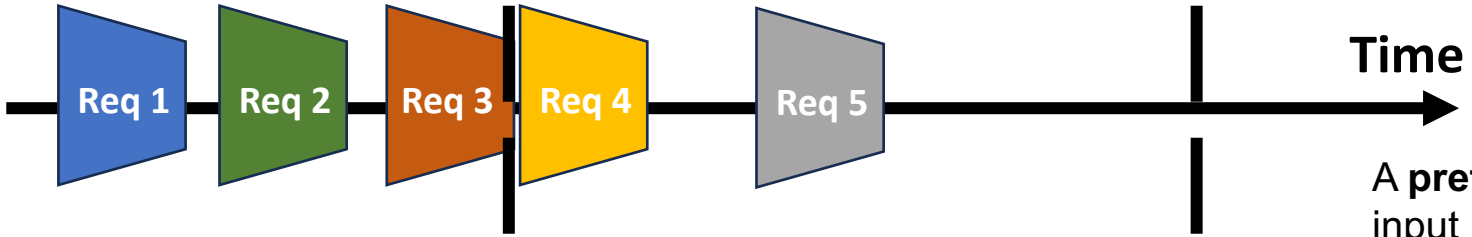
- **Key idea:** Batching requests together to improve GPU utilization
- Requests may complete at different iterations, which results in low throughput due to:

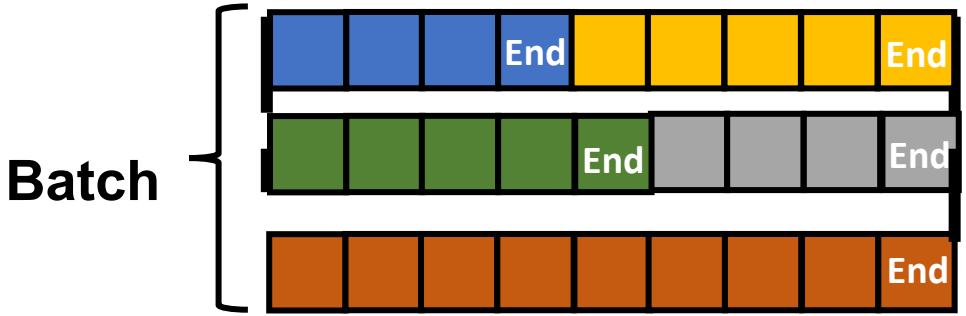


LLM Request Batching – Continuous Batching

❑**Key idea:** Different requests can be batched at the iteration level

❑**Benefits:**
- Higher GPU utilization, thus higher throughput
- New requests can start immediately





A **prefilling** step handles input prompt
Long computing time

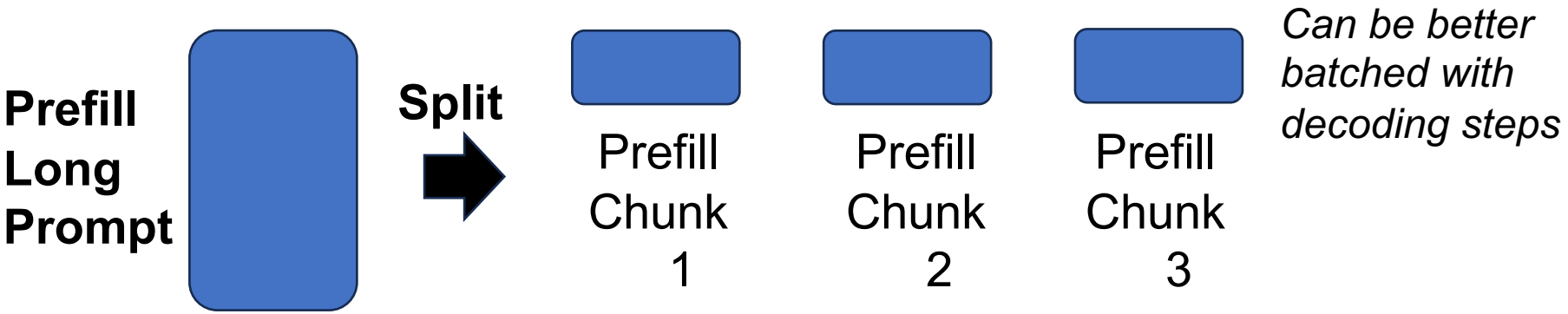
A **decoding** step generates a token
Much less time

Limitation: Batching a prefill step with a decode step can stall the decoding

Yu, G. I., Jeong, J. S., Kim, G. W., Kim, S., Chun, B. G. "Orca: A Distributed Serving System for Transformer-Based Generative Models" (OSDI 22)

LLM Request Batching – SplitFuse (Chunked Prefill)

- **Key idea:** Split prompt into chunks, and batch together chunked prefilling steps and decoding steps
- **Benefit:**
  - Remove stalls from new requests (for prefilling)



Limitation: The request latency of individual query can be harmed

Agrawal A, Kedia N, Panwar A, et al. Taming {Throughput-Latency} tradeoff in {LLM} inference with {Sarathi-Serve}, OSDI 24. 2024: 117-134.

LLM Request Scheduling

❑ Background:

- ➢ In some cases, the rate of requests exceeds the throughput of the system, even under batching
- ➢ New requests must wait in a queue before being processed
- ➢ The **order of executing requests** determines efficiency

LLM Request Priority – Shortest Job First

❑ Problem Statement

Given a set of requests, find an optimal ordering that minimizes the average latency

- ❑ **Basic Method:** First-Come First-Serve
- ❑ **Greedy Techniques:**
  - Ask the LLM, “How long will this prompt take?”
  - Train an Estimator
    - Using embeddings from last layer of LLM
    - Using small language model
  - Shortest prompts first
  - Max cache reuse

Limitation: Requires accurate predictions regarding the number of decoding rounds

Takeaways

Q2: How to optimize throughput for **multiple queries** on **one GPU**?

Page-based memory allocation

Pros: Reduce waste of memory

Cons: Require rewriting attention kernels

Cache persistence and sharing

Pros: Higher efficiency by reusing cache

Cons: Influence result quality

KV cache eviction and offloading

Pros: Less memory consumption

Cons: May hurt latency for individual query due to the cache recovery cost

Request batching

Pros: Higher utilization of GPUs, thus higher throughput

Cons: May hurt latency of individual query

Request Scheduling

Pros: Reduce average latency

Cons: Inappropriate scheduling results in low efficiency

How to Reduce LLM Inference Latency and Improve Throughput?

Q1: How to reduce latency for a **single query** on **one GPU?**

- KV cache
- Quantization
- Memory-optimized model
- Speculation

Q2: How to optimize throughput for **multiple queries** on **one GPU?**

- Page-based memory allocation
- Cache persistence and sharing
- KV cache eviction/offloading
- Request batching
- Request scheduling

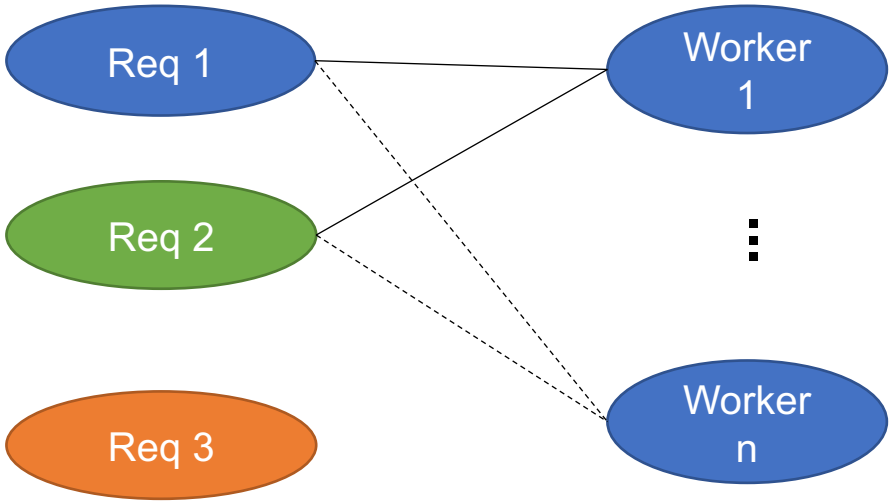
Q3: How to optimize throughput for **multiple queries** on **multiple GPUs?**

- Load balancing
- Disaggregated prefilling and decoding

LLM Request Load Balancing

❑ Problem Statement

- Given requests arriving online, assign them to workers (e.g. node or GPU) while maximizing throughput over the workload, subject to constraints (e.g. latency SLOs)



LLM Request Load Balancing Methods

❑ Technique 1: Greedy Matching

- **Max cache reuse**
  - To avoid long TTFT due to sow prefills
- **Least load**
  - To avoid unexpected TTFT, TBT
  - Memory usage, running reqs, etc.
- **Aggregate score**
  - Make a more precise estimate of TTFT and TBT
  - Cache construction cost, cache transfer, est. waiting time, etc.

$$load(s,r) = \max\left(\beta * (memory(r) - free\_mem(s)), \; queued\_tokens(s,r) / max\_tokens\_per\_batch\right)$$

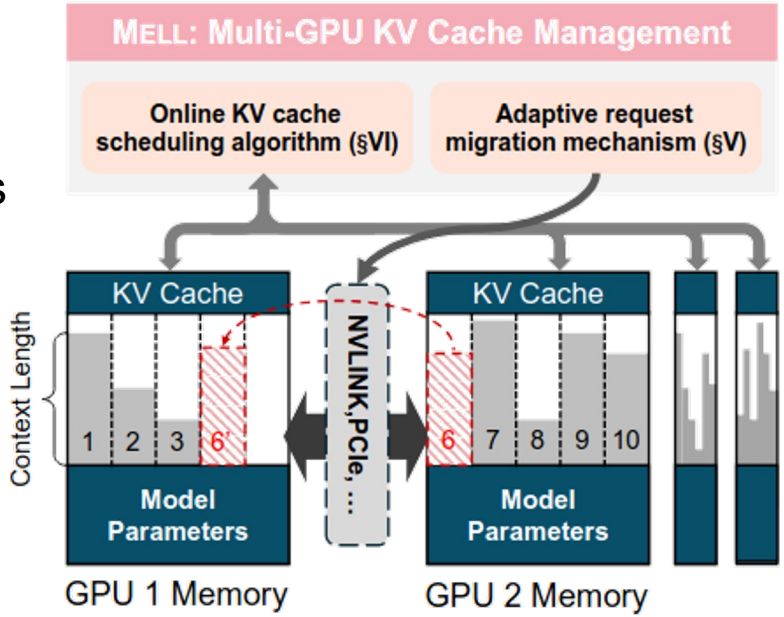
Fig: SAL's Load estimate equation

Limitation: Greedy strategy may result in ineffective load balancing

LLM Request Load Balancing Methods

❑ Technique 2: Rebalancing

- Periodically rebalance by moving KV cache to new worker
  - Avoid long TTFT due to slow prefills
- **Cache Migration**
  - To avoid memory thrashing (unexpected OOM due to long decode of past or current requests)
  - How to migrate?
    - Physically move the entries, OR
    - Recalculate from scratch (prefill)

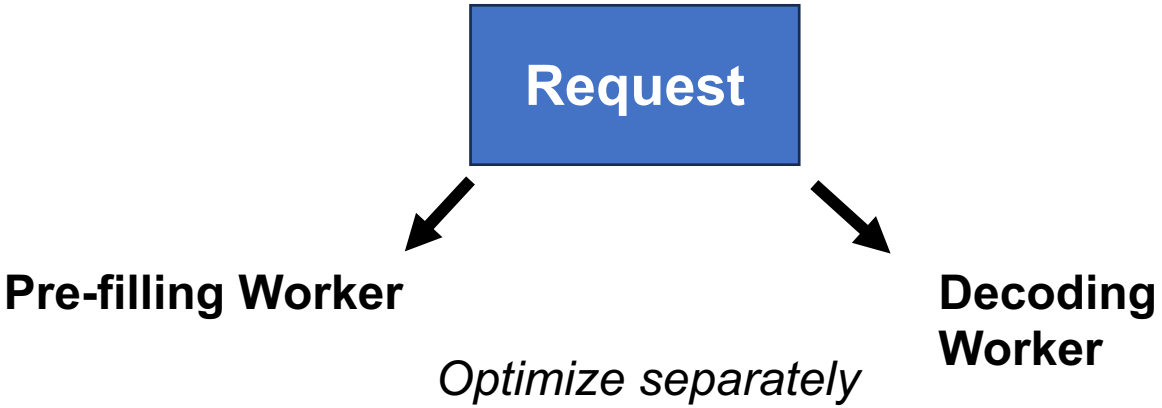


Limitation: Incur communication cost for cache migration

Qianli L, Zicong H, Fahao C, et al. Mell: Memory-Efficient Large Language Model Serving via Multi-GPU KV Cache Management[J]. arXiv preprint arXiv:2501.06709, 2025.

Disaggregated Prefilling and Decoding

- **Key idea:** Process prefilling and decoding independently based on their characteristics (*compute bound* vs *memory bound*)
- Remove the interference between these two steps



Limitation: May not utilize cache locality and incur communication overhead that should be considered

Zhou Z, Ning X, Hong K, et al. A survey on efficient inference for large language models[J]. arXiv preprint arXiv:2404.14294, 2024.

Takeaways

Q3: How to optimize throughput for **multiple queries** on **multiple GPUs**?

Load balancing

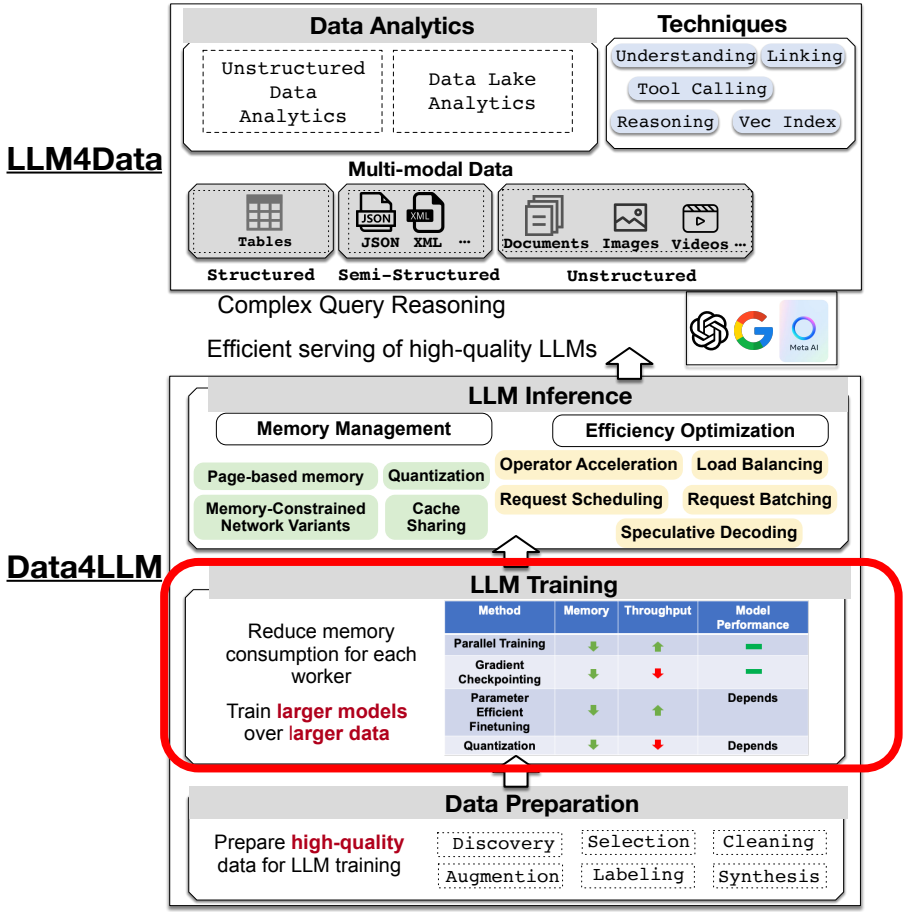
- **Pros:** Better utilization of computing resources, thus higher throughput
- **Cons:** Rely on effective scheduler that is hard to design

Disaggregated prefilling and decoding

- **Pros:** Improve hardware utilization based on features of these two stages
- **Cons:** High communication cost

Data4LLM

- ❑ **Data Management tasks**
- ❑ **LLM4Data Techniques**
  - LLM Prompting
  - RAG & Vector DB
  - Data Agents
    - Unstructured Data Analytics
    - SQL + Semantics
    - Data Lake Analytics
- ❑ **Data4LLM Techniques**
  - Data Preparation
  - LLM Inference
  - **LLM Training**
- ❑ **Open Challenges**



Overview of LLM Training

- **The costly training is dealing with:**
  - **Large model sizes** (**10B+**)
  - **Large dataset sizes** (more than **1T** tokens for pretraining, more than **1M** for supervised fine-tuning)
  - **Optimizer states** (e.g., momentum, variance) also doubles the space
- **Distributed training strategies are required**

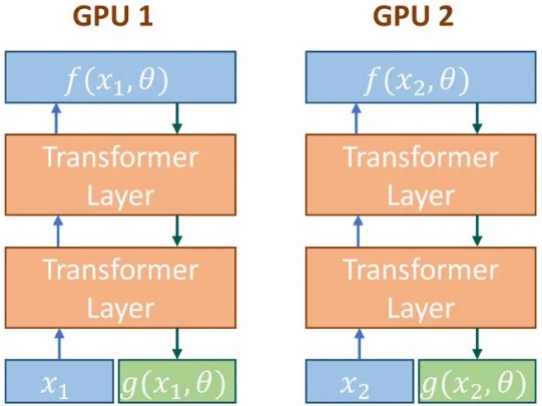
Crucial to reduce the unnecessary redundancy in the training process!

Li W, Chen X, Shu H, et al. ExCP: Extreme LLM checkpoint compression via weight-momentum joint shrinking. arXiv 2024.

Parallel Training Strategies

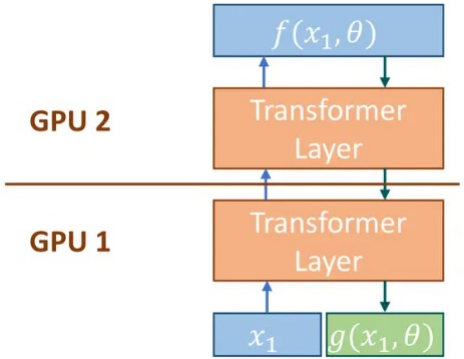
- **Key Problem:** need smart distributed training strategies, where each GPU worker only deals with **a fraction of training state and data**

Data Parallel



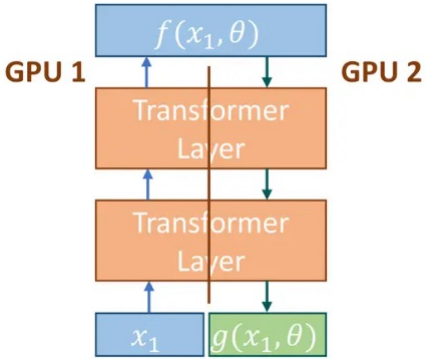
Each worker gets a **subset of mini-batch data,** computes the gradients on the data, average gradients across workers

Model Parallel



Split network by layers and place different model layers on different workers

Tensor Parallel



Split network tensors and place different parts on different workers

Different parallelism strategies can be combined for better throughput gains

Open Challenges

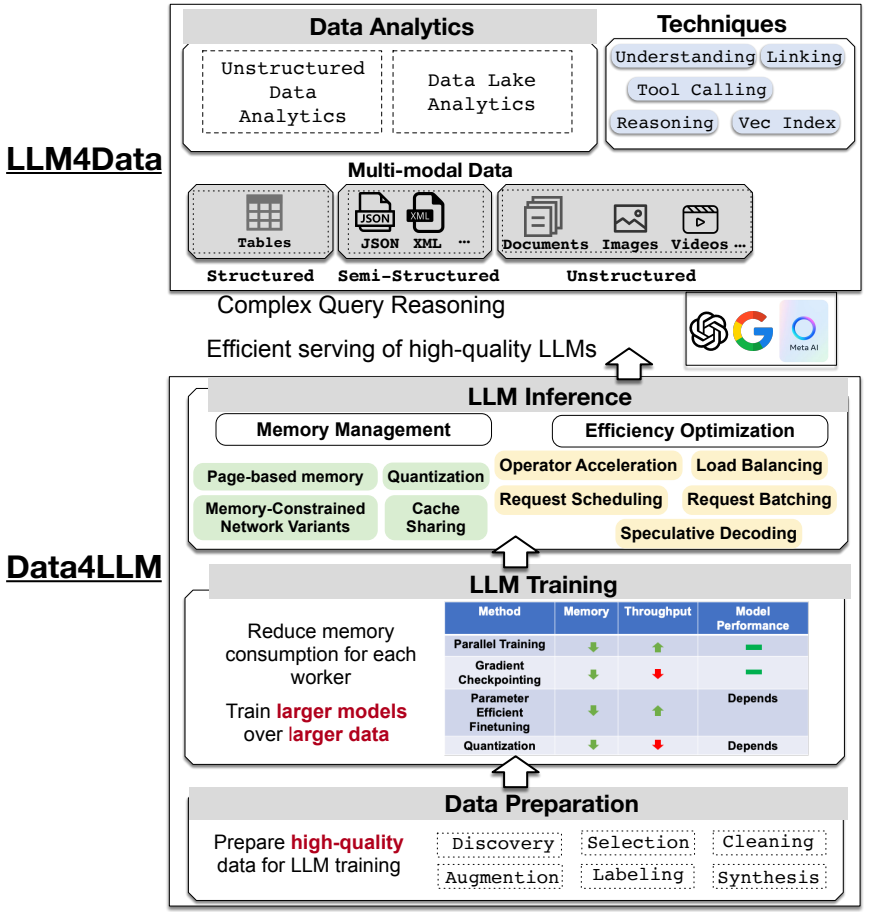
❑LLM4Data Techniques

- LLM Prompting
- RAG & Vector DB
- Data Agents
  - Unstructured Data Analytics
  - SQL + Semantics
  - Data Lake Analytics

❑ Data4LLM Techniques

- Data Preparation
- LLM Inference
- LLM Training

❑ Open Challenges

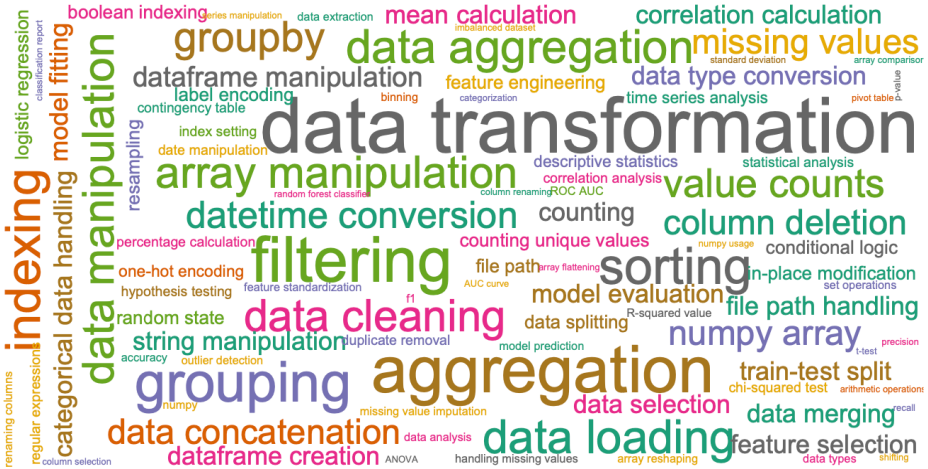


Open Challenges

- ❑ LLM4Data
  - ✓ Data Agent
  - ✓ Foundation Model for Data
- ❑ Data4LLM
  - ✓ Data Fabric
  - ✓ Data Flywheel
- ❑ Data + LLM
  - ✓ Data + LLM Codesign

① LLM4Data: Data Agent

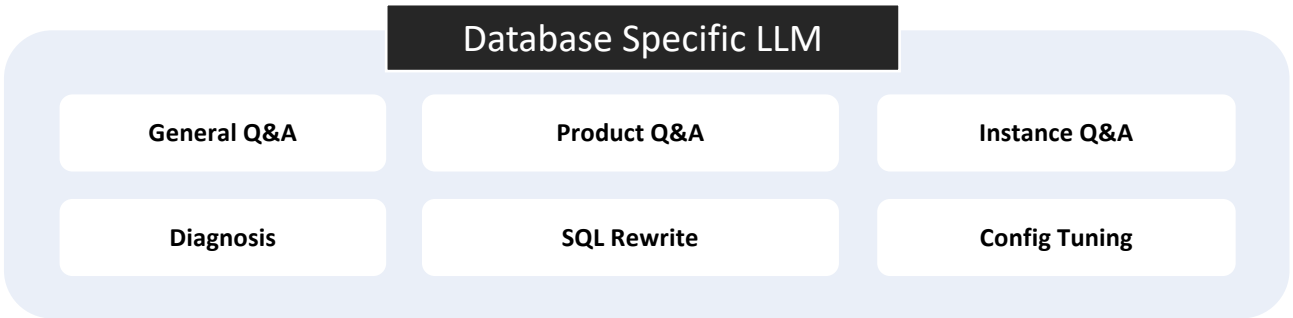
- ❑ **Data Analytics Agent**
  - ✓ **Unstructured Data Agent**
  - ✓ **Semantic Structured Data Agent**
  - ✓ **Data Lake Agent**
  - ✓ **Multi-Modal Data Agent**
- ❑ **Data Science Agent**
- ❑ **DBA Agent**
- ❑ **Database Development Agent**



② LLM4Data: Foundation Models for Data

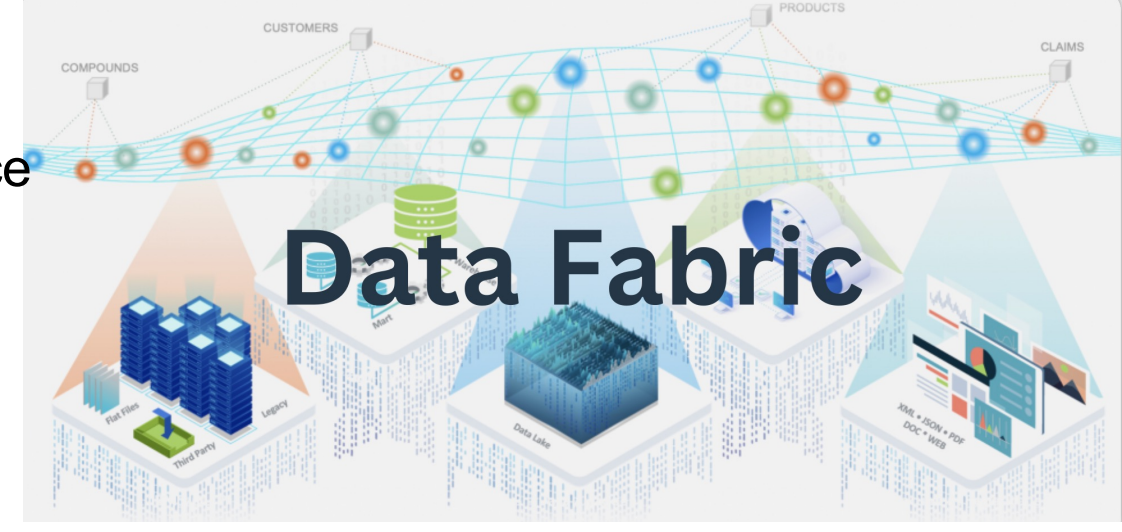
◻ Case-by-Case LLM Finetuning → Database-Specific LLM Construction

- **Pretrain:** Collect sufficient database-domain tokens (e.g., in millions) as pre-training corpora from sources like database textbook and query analysis
- **Finetune:** Instruction Understanding in SQL / Text → Basic Q&A (DB / Product / Instance) → Task-Solving in DB Domains → Alignment to Database Experts
- **Evaluation:** Evaluate the accuracy and robustness of the database model with carefully-crafted validation dataset, measuring metrics, and end-to-end testbed.



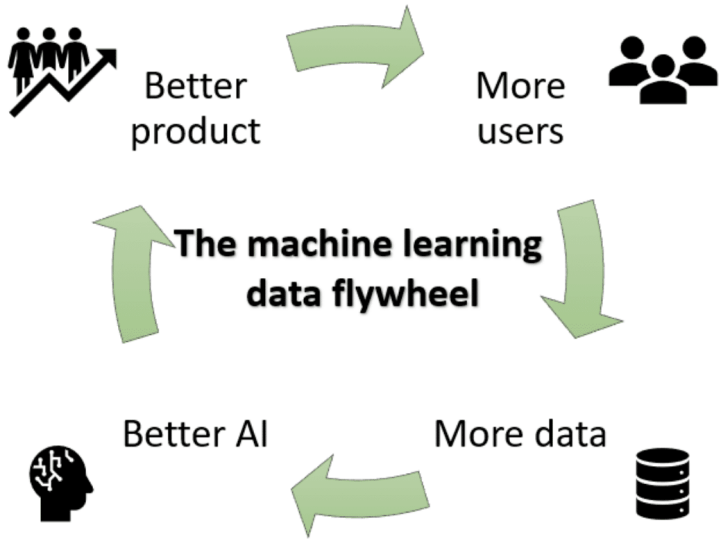
③ Data4LLM: Data Fabric

- Unified Data Access: Provides a single, consistent interface for accessing data, facilitates real-time data access and sharing across the organization.
- Semantic Catalog and Semantic Data Organization
- Active Meta Data Management and Update
- Data pipelines
- Data Lineage and Provenance
- Support for Diverse Tools
- Self-Service Analytics



④ Data4LLM: Data Flywheel

- ❑ Feedback Loop
- ❑ Data Augmentation
- ❑ Feature Augment
- ❑ Data Reflection
- ❑ Feedback Optimization
- ❑ Continuous Improvement



⑤ Data + LLM: Co-design

- ❑ Data + AI Model
- ❑ Iterative Loop
- ❑ Data + AI Ops
- ❑ Data + AI Infrastructure
- ❑ Data Designer

Thanks!

Slides: *https://dbgroup.cs.tsinghua.edu.cn/ligl/activities.html*

Data+AI Paper List: *https://github.com/code4DB/LLM4DB*

System*:* *https://github.com/TsinghuaDatabaseGroup/Unify*