北京大学
PEKING UNIVERSITY

# Retrieval-Augmented Generation for Large Language Models A Survey

Yunfan Gao[a], Yun Xiong[b], Xinyu Gao[b], Kangxiang Jia[b], Jinliu Pan[b], Yuxi Bi[c], Yi Dai[a], Jiawei Sun[a], Meng Wang[c], and Haofen Wang [a,c]

[a]Shanghai Research Institute for Intelligent Autonomous Systems, Tongji University
[b]Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
[c]College of Design and Innovation, Tongji University

year={2024.03.27}
Cites={139}

1 Naive RAG

2 Advanced RAG

3 Modular RAG

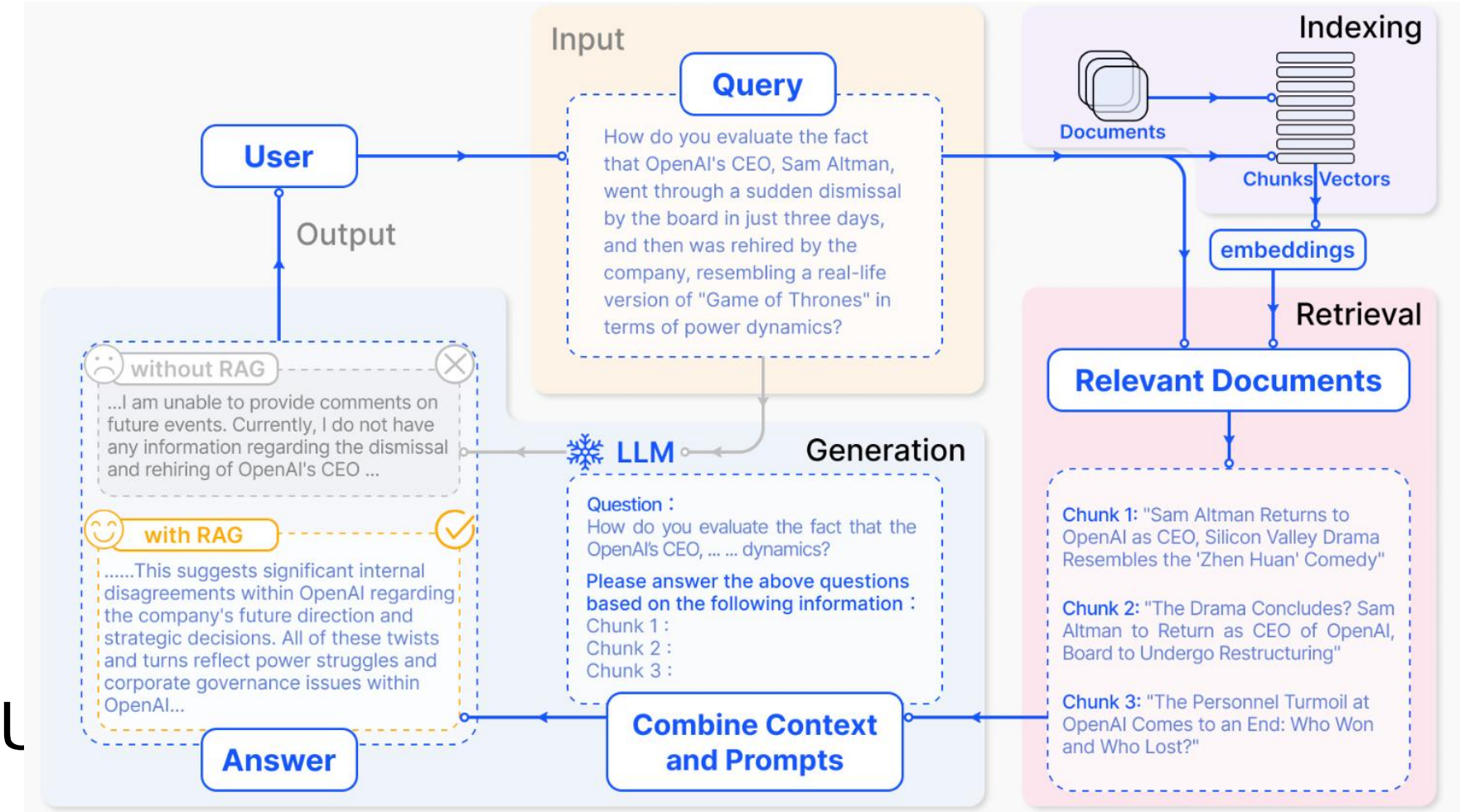
4 RETRIEVAL

5 GENERATION

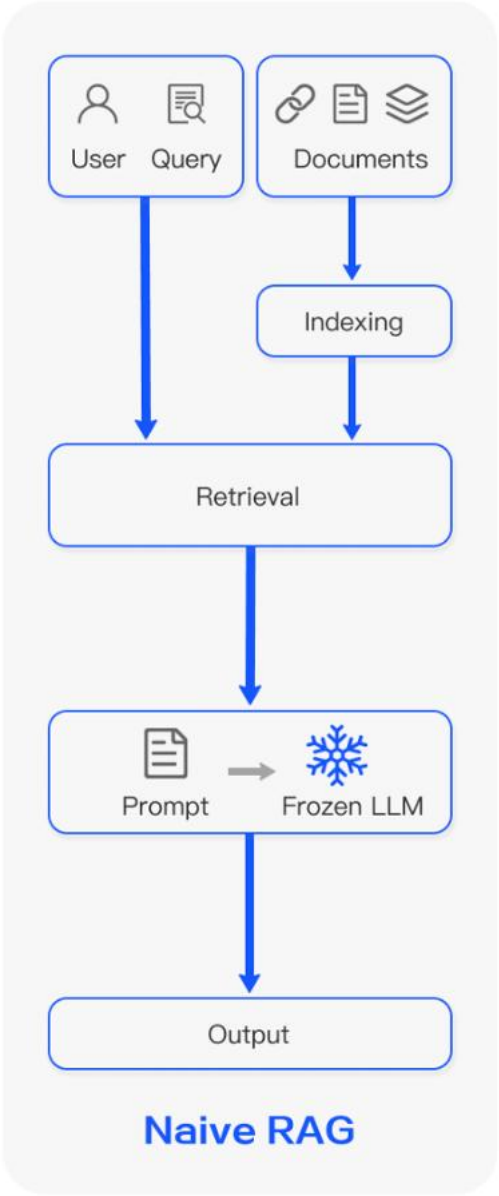
6 AUGMENTATION

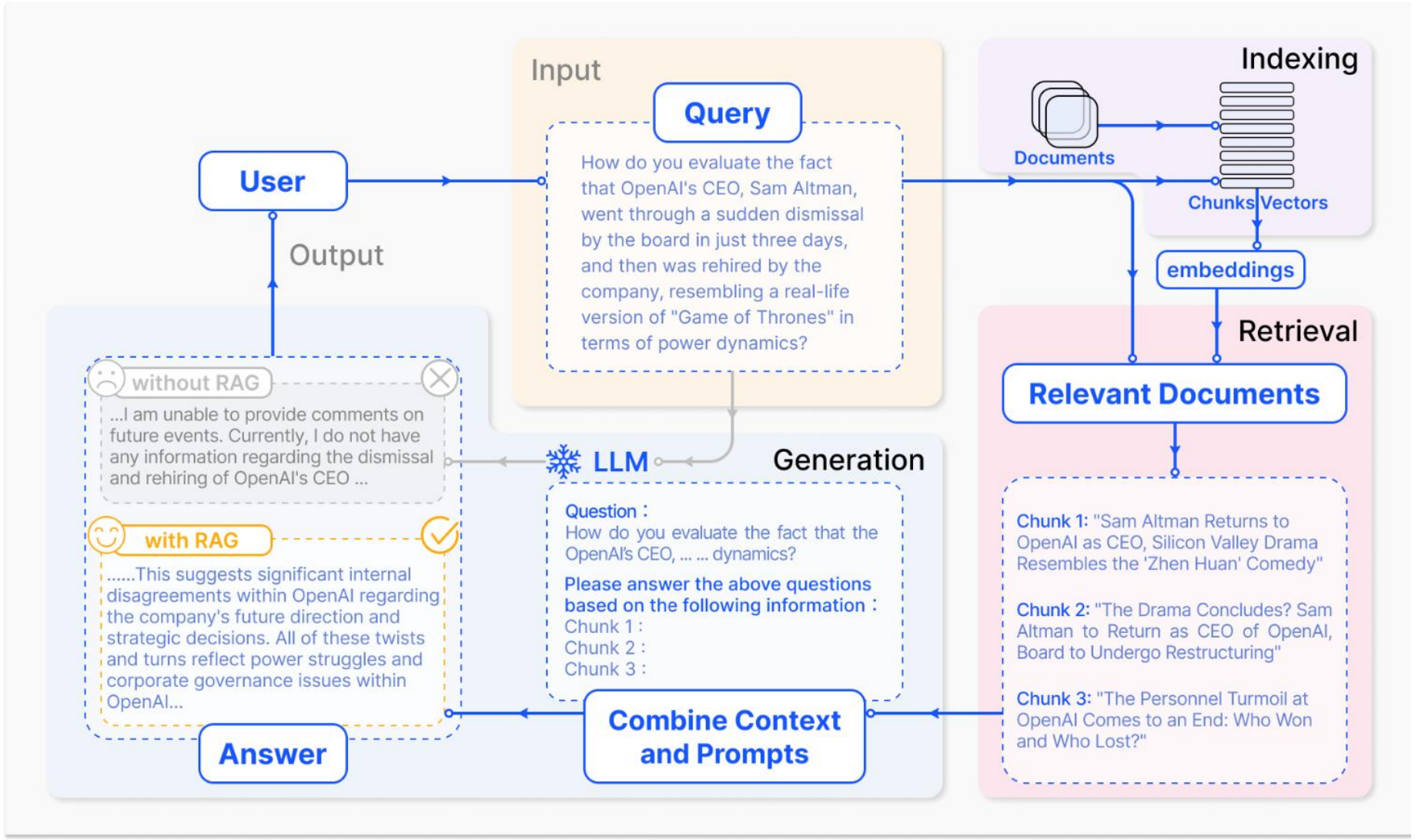
7 TASK AND EVALU

8 DISCUSSION AND FUTURE PROSPECTS

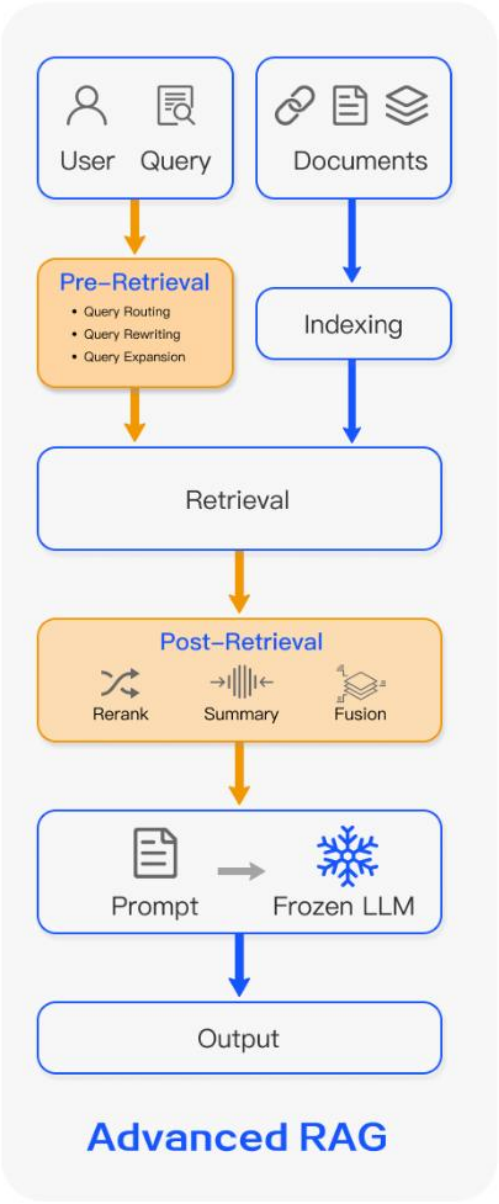


# Naïve RAG


User
Query
Documents
Indexing
Retrieval
Prompt
Frozen LLM
Output
Naive RAG


Input
Query
How do you evaluate the fact that OpenAI's CEO, Sam Altman, went through a sudden dismissal by the board in just three days, and then was rehired by the company, resembling a real-life version of "Game of Thrones" in terms of power dynamics?
User
Output
Indexing
Documents
Chunks Vectors
embeddings
Retrieval
Relevant Documents
Chunk 1: "Sam Altman Returns to OpenAI as CEO, Silicon Valley Drama Resembles the 'Zhen Huan' Comedy"
Chunk 2: "The Drama Concludes? Sam Altman to Return as CEO of OpenAI, Board to Undergo Restructuring"
Chunk 3: "The Personnel Turmoil at OpenAI Comes to an End: Who Won and Who Lost?"
without RAG
...I am unable to provide comments on future events. Currently, I do not have any information regarding the dismissal and rehiring of OpenAI's CEO ...
with RAG
.....This suggests significant internal disagreements within OpenAI regarding the company's future direction and strategic decisions. All of these twists and turns reflect power struggles and corporate governance issues within OpenAI...
LLM
Generation
Question：
How do you evaluate the fact that the OpenAI's CEO, ... ... dynamics?
Please answer the above questions based on the following information：
Chunk 1：
Chunk 2：
Chunk 3：
Answer
Combine Context and Prompts

# *Advanced RAG*




Overcome the limitations of Naive RAG.

## Pre-retrieval

(optimizing the indexing structure and the original query, enhance the quality of the content being indexed)

- enhancing data granularity
- optimizing index structures
- adding metadata
- alignment optimization
- mixed retrieval

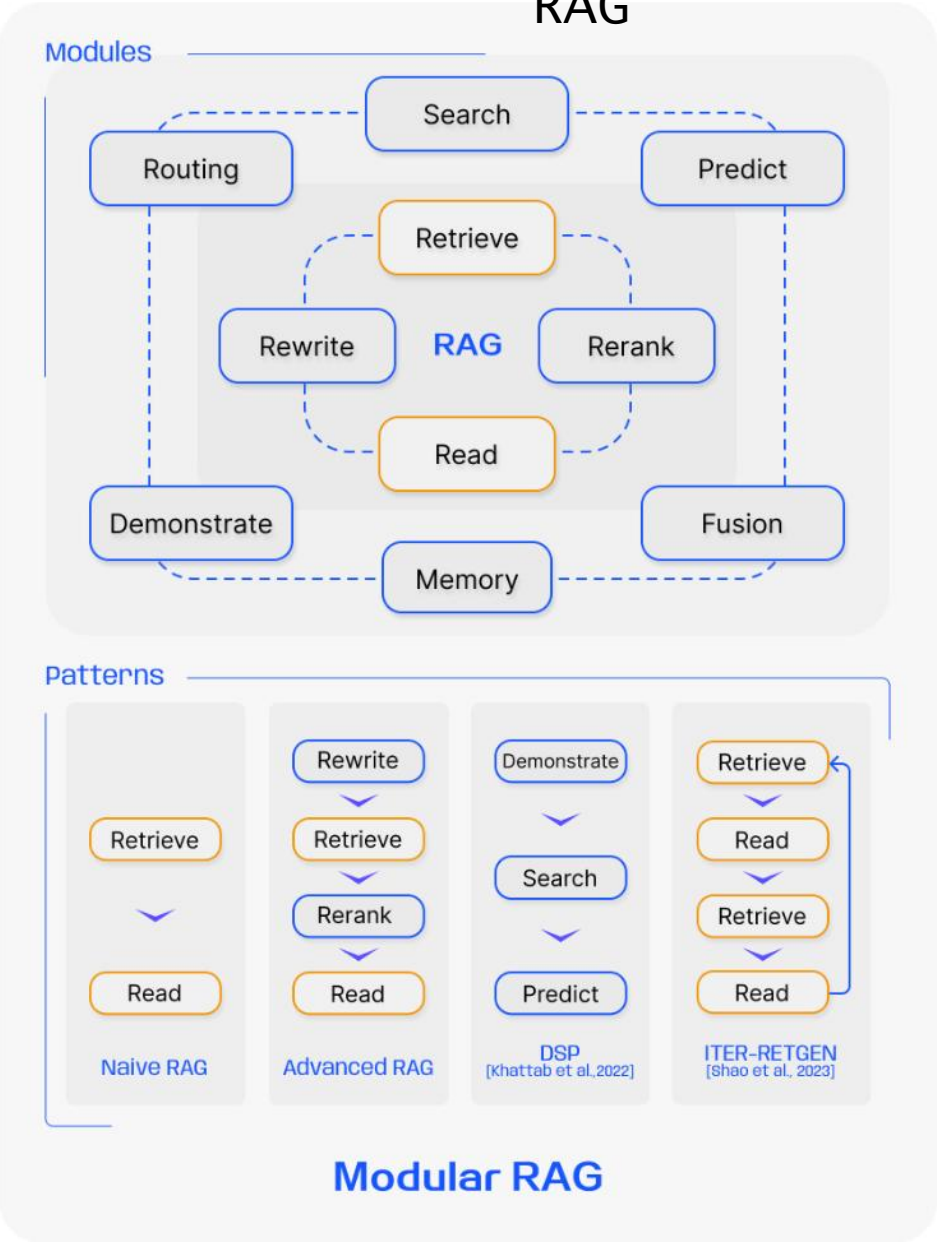
## Post-retrieval

(integrate it effectively with the query )

- rerank chunks
- context compressing

# *Modular RAG*

Modular RAG builds upon the foundational principles of Advanced and Naive RAG




## *New Modules:*

- Search module
- Memory module [18] P9
- RAG- Fusion [16] P11
- Routing
- The Predict module [13] P13
- Task Adapter module

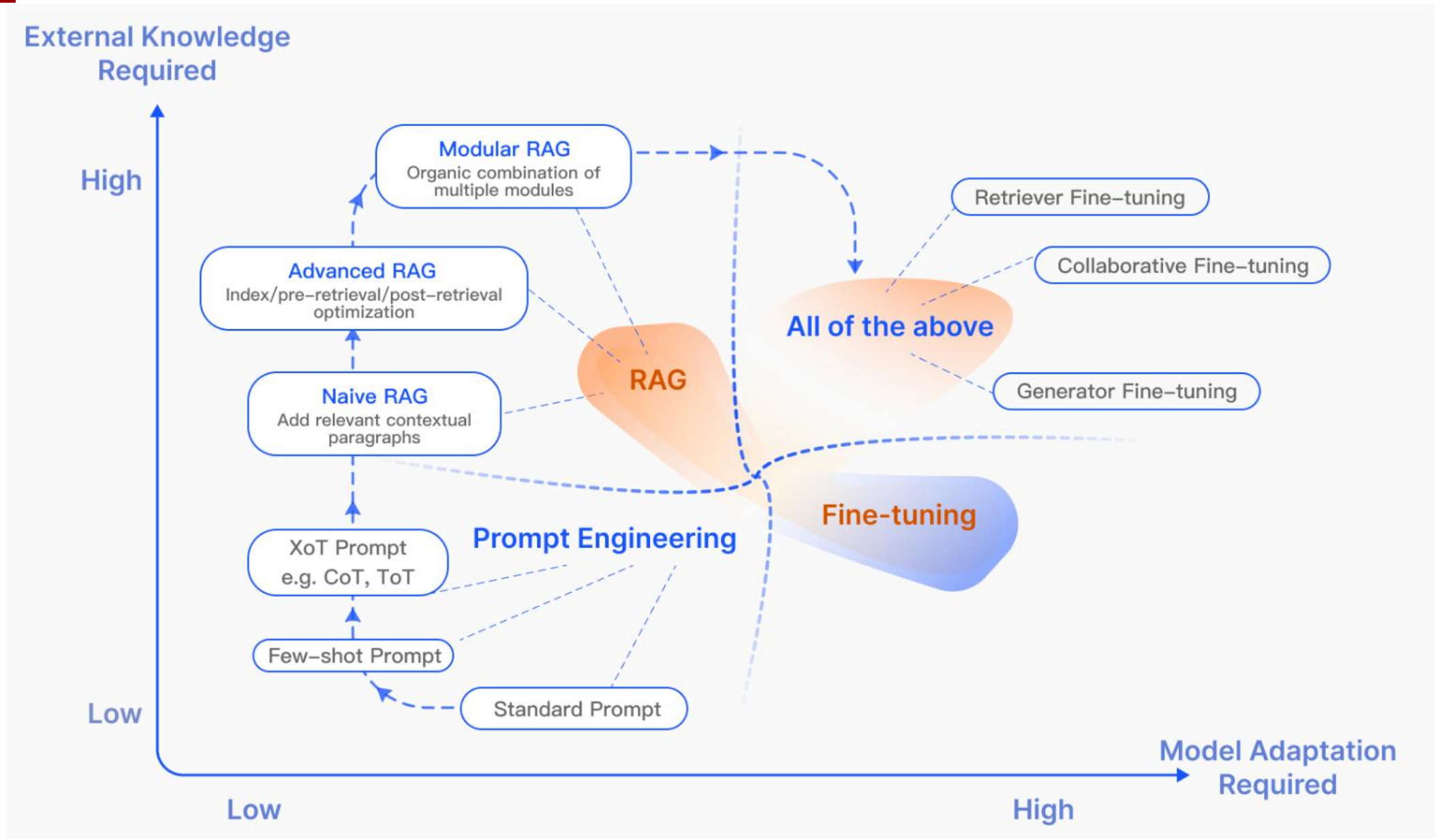
## *New Patterns:(*allowing module **Substitution** or **Reconfiguration**)

### Substitution

- Rewrite-Retrieve-Read [7]model P12
- Generate-Read [13] P13
- Recite- Read [22] P14

### Reconfiguration

- ITER- RETGEN [14] P16


External Knowledge Required
High
Low
Low
High
Model Adaptation Required
Modular RAG
Organic combination of multiple modules
Advanced RAG
Index/pre-retrieval/post-retrieval optimization
Naive RAG
Add relevant contextual paragraphs
RAG
XoT Prompt
e.g. CoT, ToT
Few-shot Prompt
Standard Prompt
Prompt Engineering
Retriever Fine-tuning
Collaborative Fine-tuning
All of the above
Generator Fine-tuning
Fine-tuning

## A. Retrieval Source

A. RAG relies on external knowledge to enhance LLMs, while the type of **retrieval source** and **the granularity of retrieval** units both affect the final generation results.
B. semi-structured data (PDF) , structured data (Knowledge Graph, KG) , Wikipedia, combination of text and table, database, knowledge graphs, *LLMs-Generated Content*
C. KnowledGPT [15] P17 , SKR [58] P18, GenRead [13] P13

## B. Indexing Optimization

A. Documents will be processed, segmented, and transformed into Embeddings to be stored in a vector database.
*B. Chunking Strategy:* Small2Big[90], Sliding window[90] P19
*C. Hierarchical index structure, Knowledge Graph index*

## C. Query Optimization

A. Naive RAG is its direct reliance on the user's original query as the basis for retrieval.
*B. Query Expansion: Multi-Query , Sub-Query, Chain-of-Verification*
*C. Query Transformation:* RRR (Rewrite-retrieve-read) [7], HyDE [11] P20

## D. Embedding

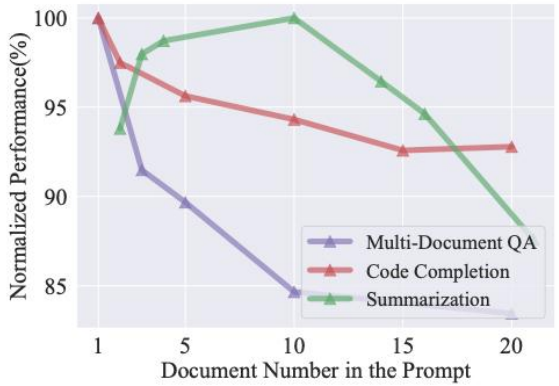
A. This mainly includes a sparse encoder (BM25) and a dense retriever (BERT architecture Pre-training language models)
*B. Mix/hybrid Retrieval*
*C. Fine-tuning Embedding Model:* PROMPTAGATOR [21] P21 , LLM-Embedder [97] ,

# A.什么时候用RAG?

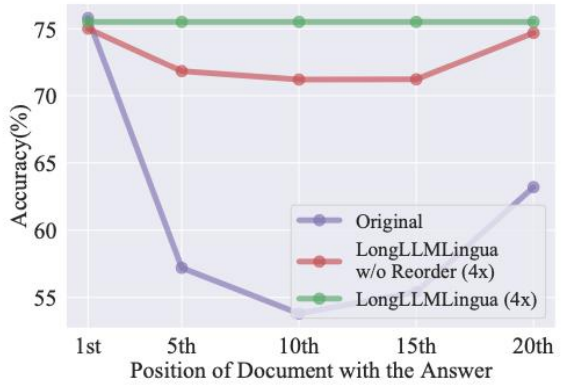
A. 模型越来越强，参数化知识越来越多，很多情况下单独使用LLM比使用RAG还好
B. 找一种优雅的方法来判断这个query要不要使用RAG
C. 例如输入query，不使用RAG和使用RAG分别进行回答，然后选个裁判？（不优雅
D. 搞一个评分？分数小的（简单问题）直接作答，分高的（难）使用RAG

# B. 压缩即智能

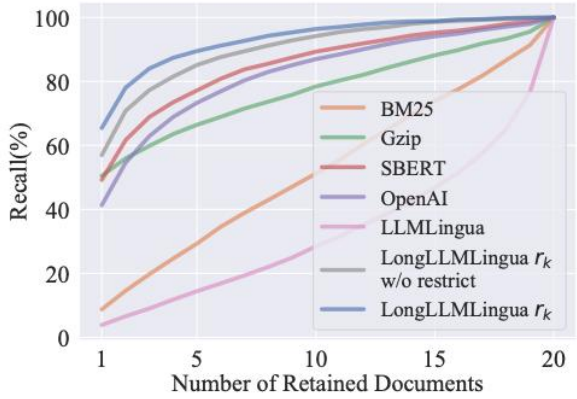
A. RAG搜索到的可能有很多噪声，会给llm的生成带来很大的误差.
B. 找到一种去除搜索噪声的方法，压缩？



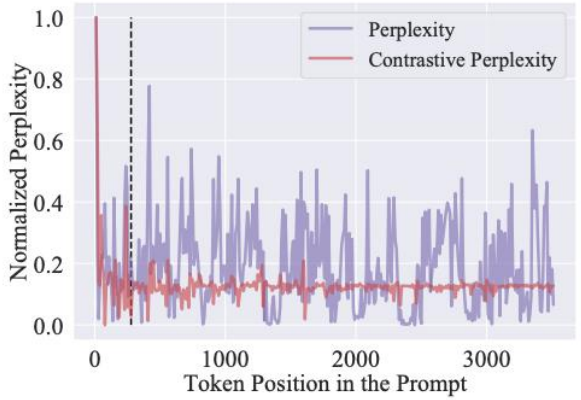

(a) Performance v.s. Document Number




(b) Performance v.s. Key Information Position




(a) Recall Distribution




(b) Perplexity Distribution

# Lift yourself up Retrieval-augmented text generation with self memory

Advances in Neural Information Processing Systems, 2023 cite=18

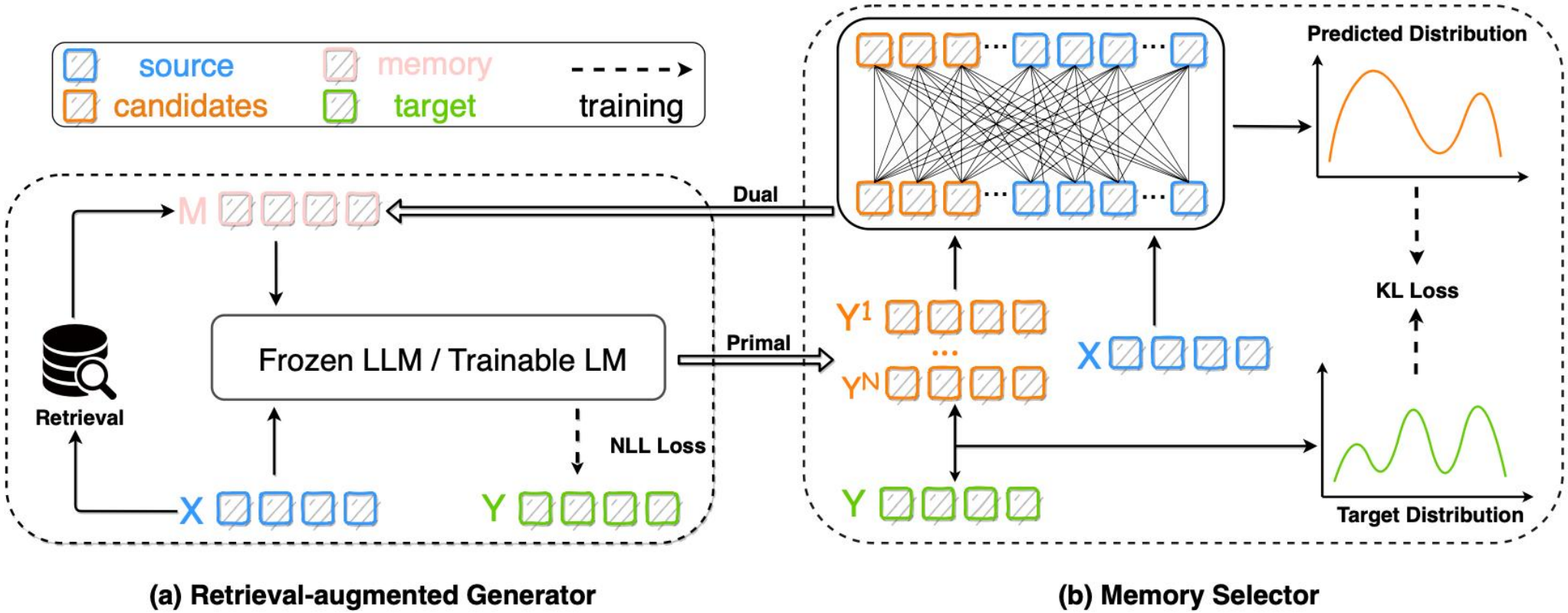
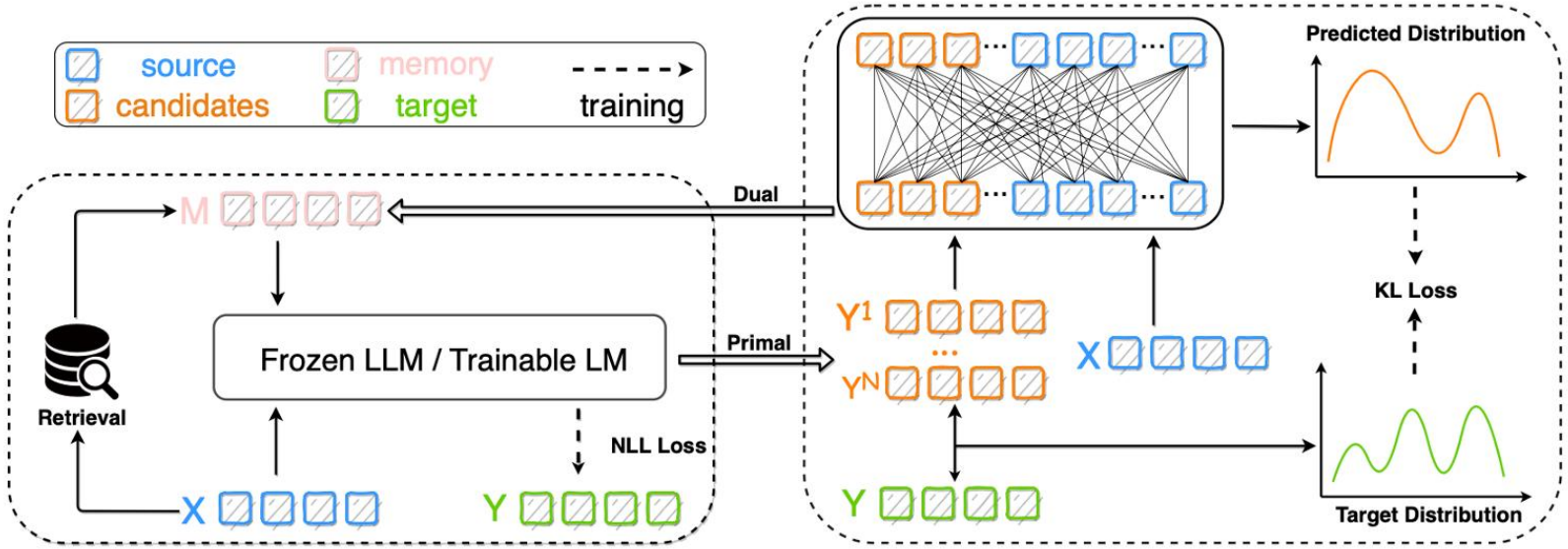



Figure 2: Overall framework. There are two components in `Selfmem`, a retrieval-augmented generator (a) and a memory selector (b). For the primal problem, (a) takes source and memory as input to generate candidates for (b). For the dual problem, (b) takes as input source and generated candidates to select memory for (a).




(a) Retrieval-augmented Generator (b) Memory Selector

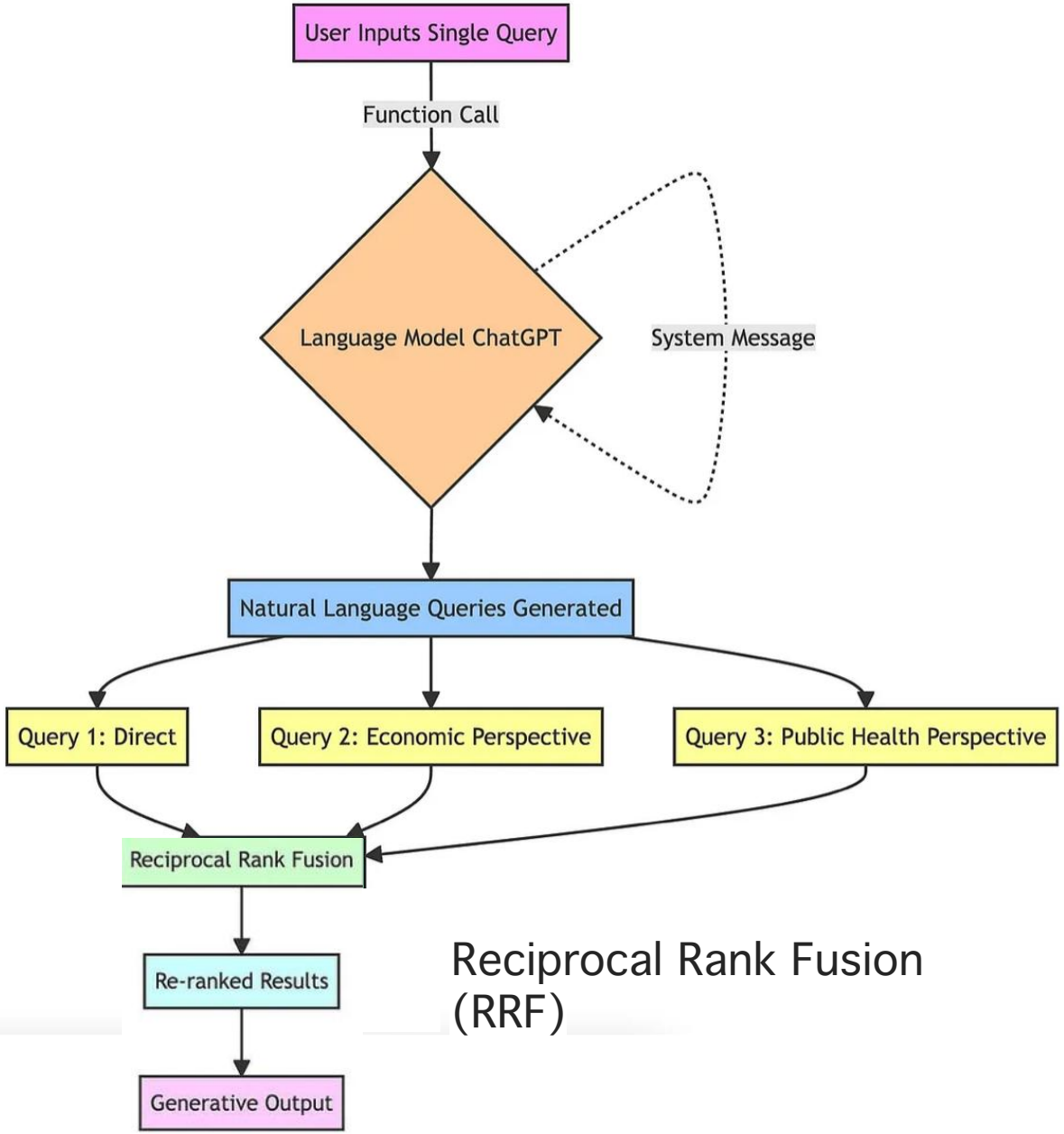
**Algorithm 1** `Selfmem` Framework

**Require:** a dataset $\mathbb{D}$, a retriever $R$, a memory selection metric $\Delta(\cdot,\cdot)$, a retrieval-augmented generator $G_\xi$, and a memory selector $S_\theta$

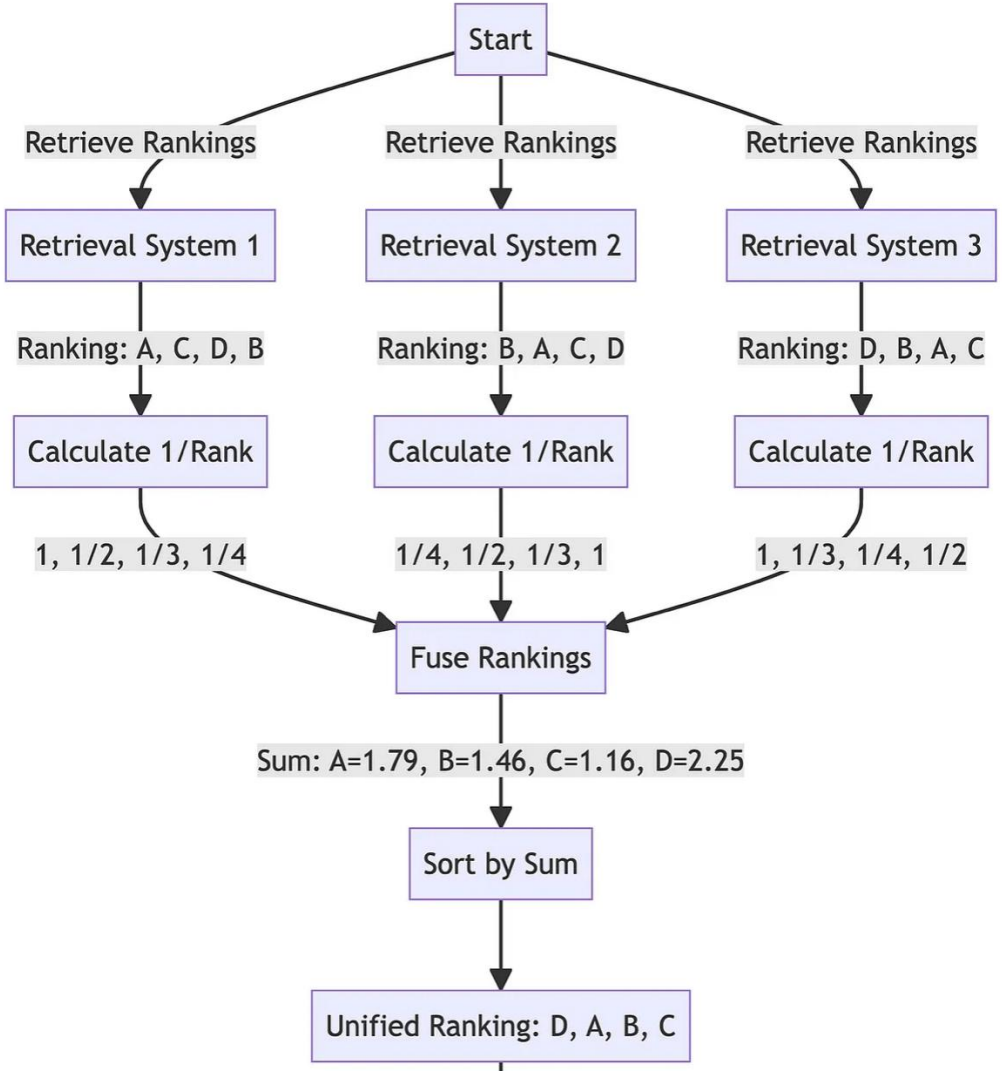
retrieve memory $\mathbb{M}$ in $\mathbb{D}$ with $R$
train $G_\xi$ with $\mathbb{D}$ and $\mathbb{M}$ (if not LLM)
use $G_\xi$ to generate candidate pool $\mathbb{C}$ with $\mathbb{M}$ in candidate mode
train $S_\theta$ on $\mathbb{C}$ with $\Delta(\cdot,\cdot)$
**while** not converged in the validation set **do**
    $S_\theta$ selects memory from $\mathbb{C}$ as $\mathbb{M}$
    $G_\xi$ generates candidate pool $\mathbb{C}$ with $\mathbb{M}$ in candidate mode
**end while**
$G_\xi$ generates the final hypothesis with $\mathbb{M}$ in hypothesis mode

# Forget RAG, the Future is RAG-Fusion

blog




Reciprocal Rank Fusion (RRF)



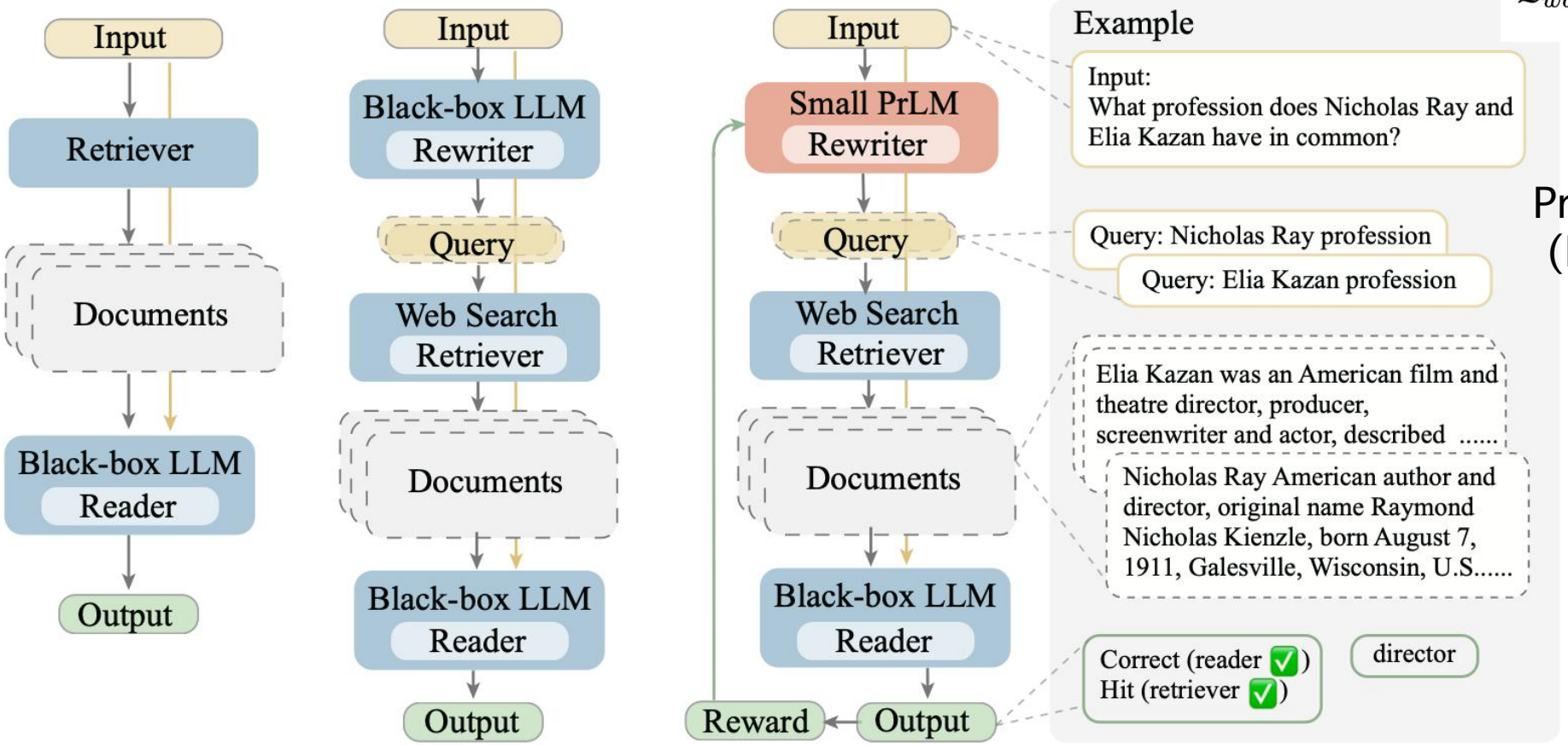
# Query Rewriting for Retrieval-Augmented Large Language Models

arXiv preprint arXiv:2305.14283, 202310 cite=31

$$D_{Train} = \{(x, \tilde{x}) | \hat{y} = y\}$$

$$\mathcal{L}_{warm} = -\sum_{t} logp_\theta(\hat{\tilde{x}}_t \mid \tilde{x}_{<t}, x).$$

Proximal Policy Optimizatio (PPO)




(a) Retrieve-then-read (b)Rewrite-retrieve-read (c) Trainable rewrite-retrieve-read

Figure 1: Overview of our proposed pipeline. From left to right, we show (a) standard *retrieve-then-read* method, (b) LLM as a query rewriter for our *rewrite-retrieve-read* pipeline, and (c) our pipeline with a trainable rewriter.

# Generate rather than retrieve: Large language models are strong context generators

ICLR2023, 202301 cite=168

$$p(a|q) = \sum_i p(a|d_i, q)p(d_i|q)$$

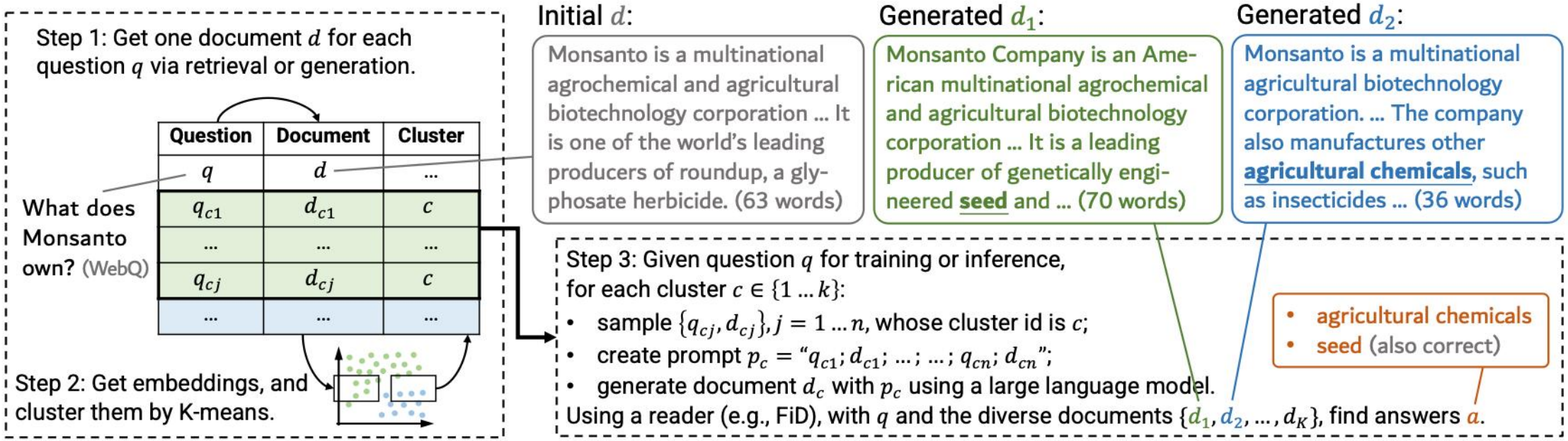



Figure 1: An overall framework of clustering-based prompting method. It leverages distinct question-document pairs sampled from each embedding cluster as in-context demonstrations to prompt a large language model to generate diverse documents, then read the documents to predict an answer.

# Recitation-augmented language models

ICLR2023, 202302 cite=69

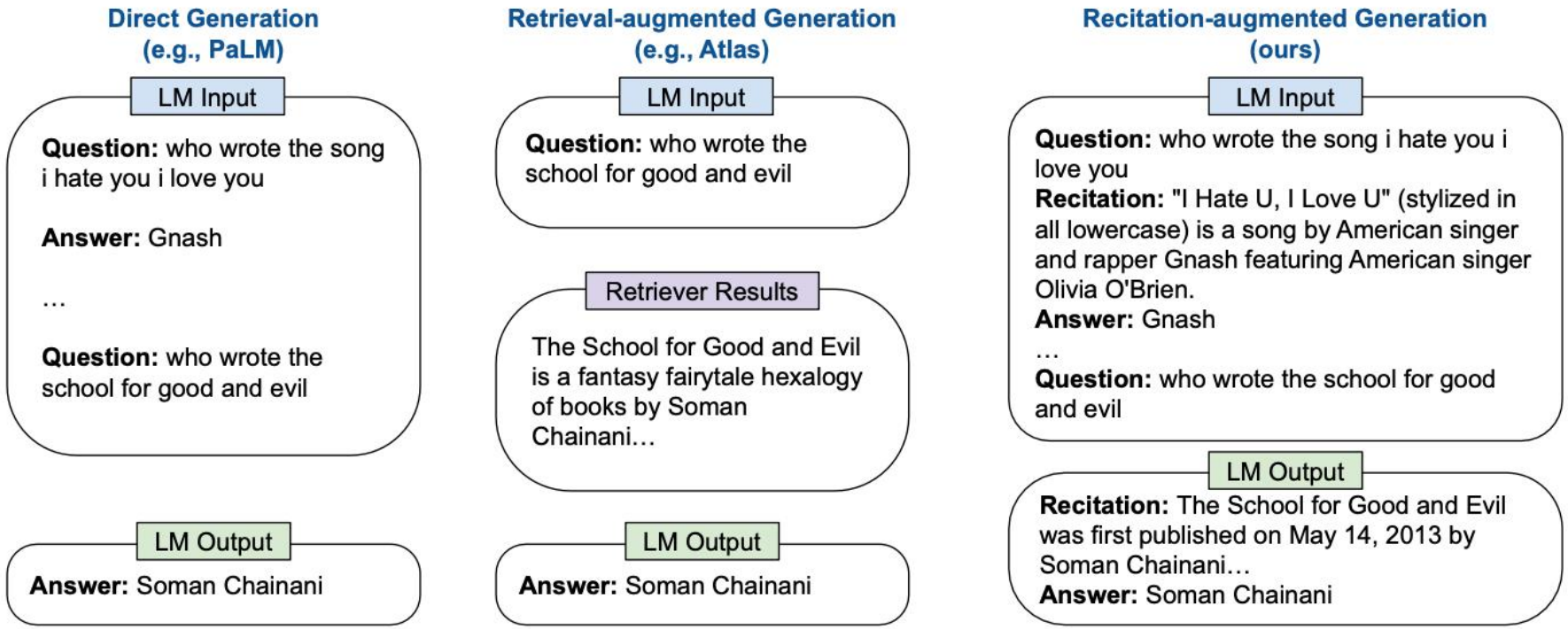
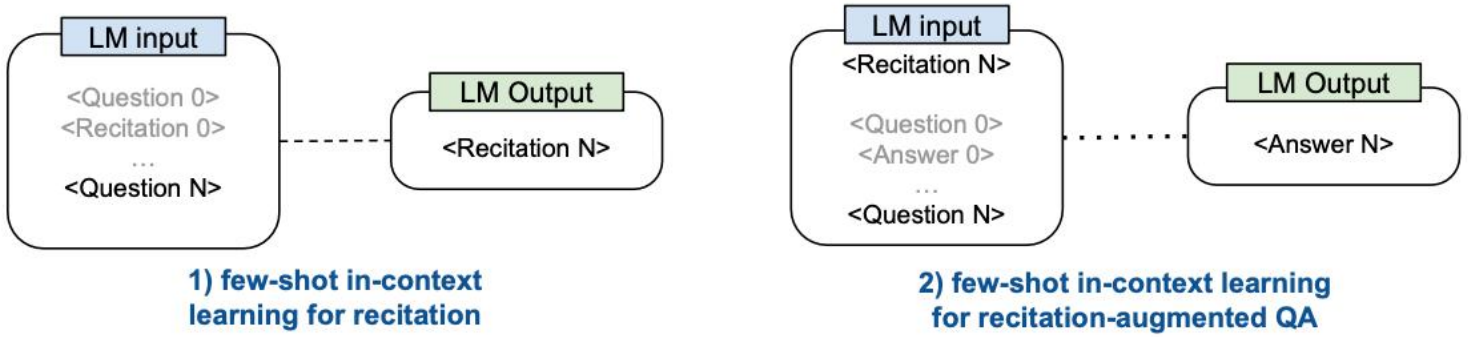
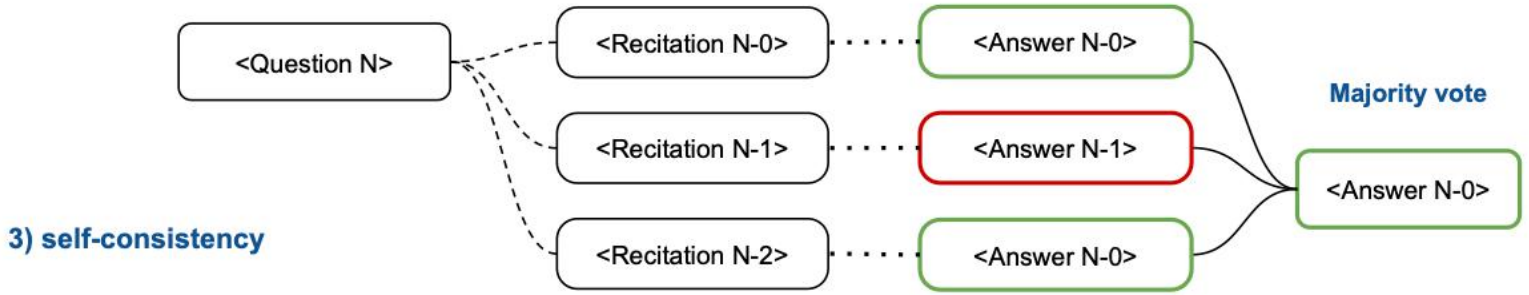


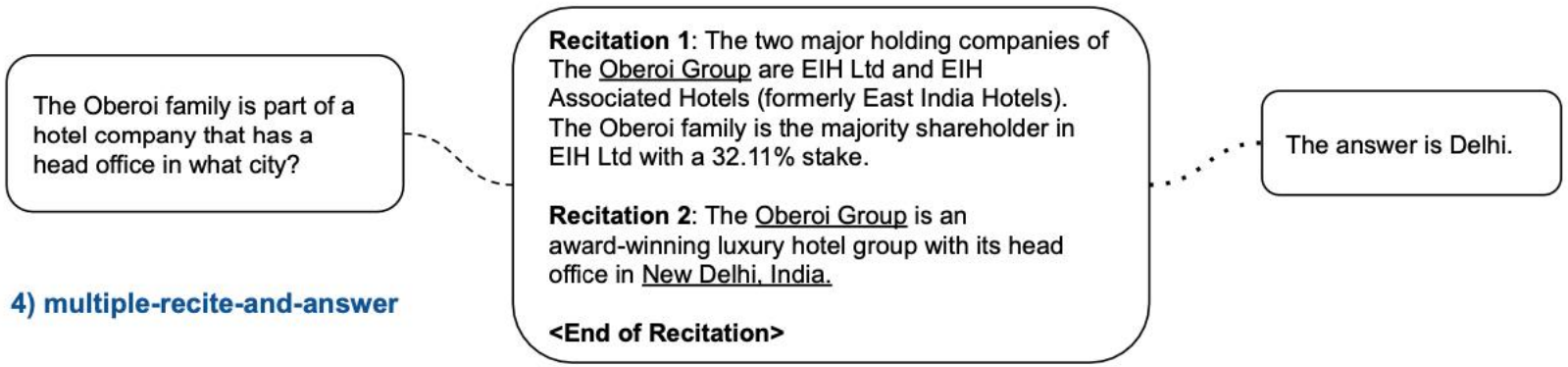

Figure 1: Illustration of evaluating (few-shot) open-domain question answering with (closed-book) direct generation (Chowdhery et al., 2022), (open-book) retrieval-augmented generation (Izacard et al., 2022), and (closed-book) recitation-augmented generation (ours).

# Recitation-augmented language models

ICLR2023, 202302 cite=69



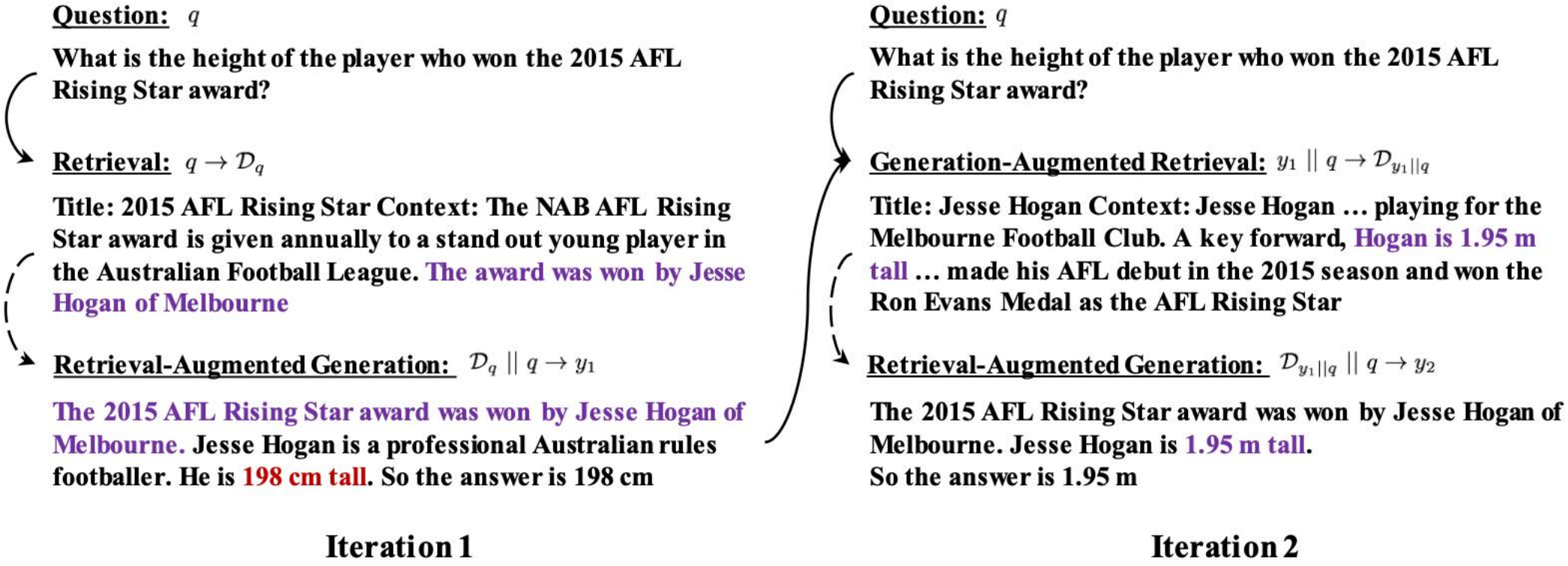



# Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy

arXiv preprint arXiv:2305.15294, 2023, 202310 cite=35

核心:检索增强的生成，生成增强的检索

实线是检索，虚线是生成




$$y_t = \mathcal{M}(y_t|\text{prompt}(\mathcal{D}_{y_{t-1}||q}, q)), \quad \forall 1 \leq t \leq T$$

Therefore, we distill knowledge from a re-ranker to a retriever,训练目标是re-ranker和retriever之间的预测差

# KnowledGPT: Enhancing Large Language Models with Retrieval and Storage Access on Knowledge Bases

Cite=15 202308这些API由相应的KB提供

这些API由相应的KB提供

**Query:** What are the titles of the poet writing *Quiet Night Thoughts (Jing Ye Si)* ?

**Original Answer from GPT-4:** The poet of *Quiet Night Thoughts* is Li Bai from the Tang Dynasty. He has multiple titles, including 'The Immortal Poet', 'The Wandering Poet', and 'The Wine Immortal'. His poetic talent was outstanding, his works are widely circulated, and he has been revered as 'the God in Poetry' by later generations.

**Code for Knowledge Retrieval:**

```
def search():
    messages = ''
    author, msg = find_entity_or_value(entity_aliases = ['Quiet Night Thoughts '], relation_aliases = [‘author’, ‘creator', ‘writor'])
    messages += msg
    if author:
        titles, msg = find_entity_or_value(entity_aliases = author, relation_aliases = ['title', 'also known as', ‘appellation'])
        messages += msg
    return messages
```

**Retrieved Knowledge from KBs:**
[FROM CNDBPedia]
[find_entity_or_value(entity_aliases = ['Quiet Night Thoughts'], relation_aliases = ['author', 'creator', 'writer']) -> ] 'Quiet Night Thoughts' (a poem by Li Bai), Author: Li Bai, The Orderer of Heaven's Mandate
[find_entity_or_value(entity_aliases = ['Li Bai', 'The Orderer of Heaven's Mandate'], relation_aliases = ['title’, also known as', 'appellation']) -> ] Li Bai (famous Tang Dynasty poet), Title: Blue Lotus Hermit, Exiled Immortal.

**Answer with Retrieved Knowledge:** The poet Li Bai, who wrote *'Quiet Night Thoughts'*, has the titles 'Blue Lotus Hermit' and 'Exiled Immortal'.

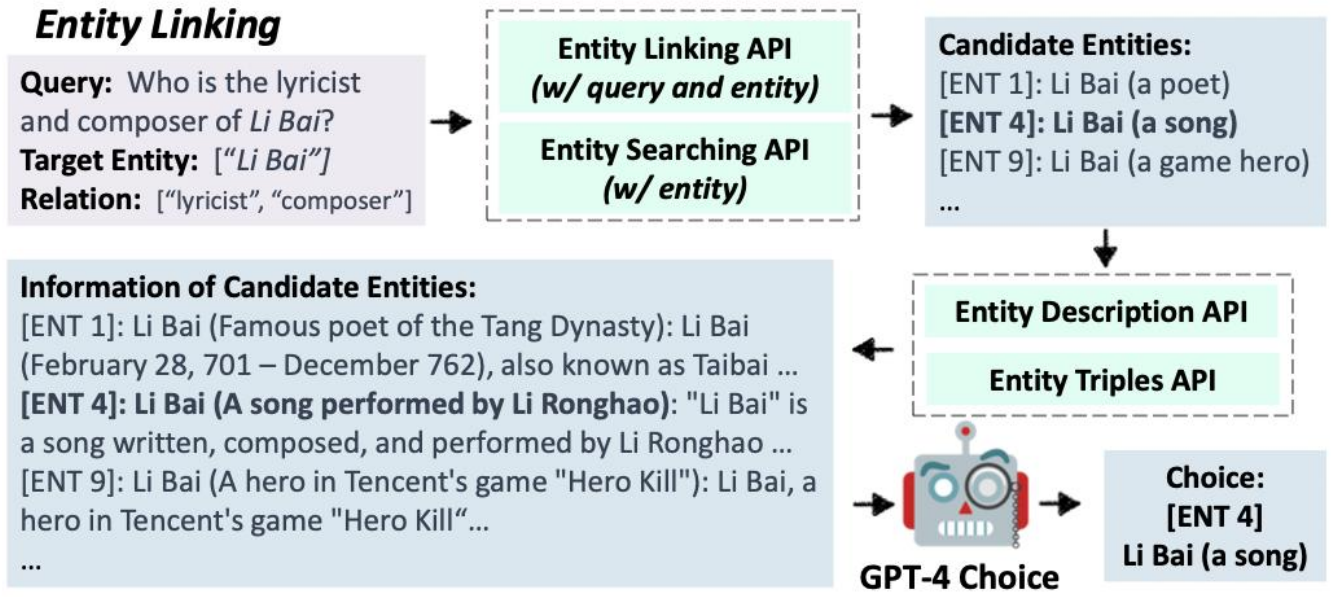
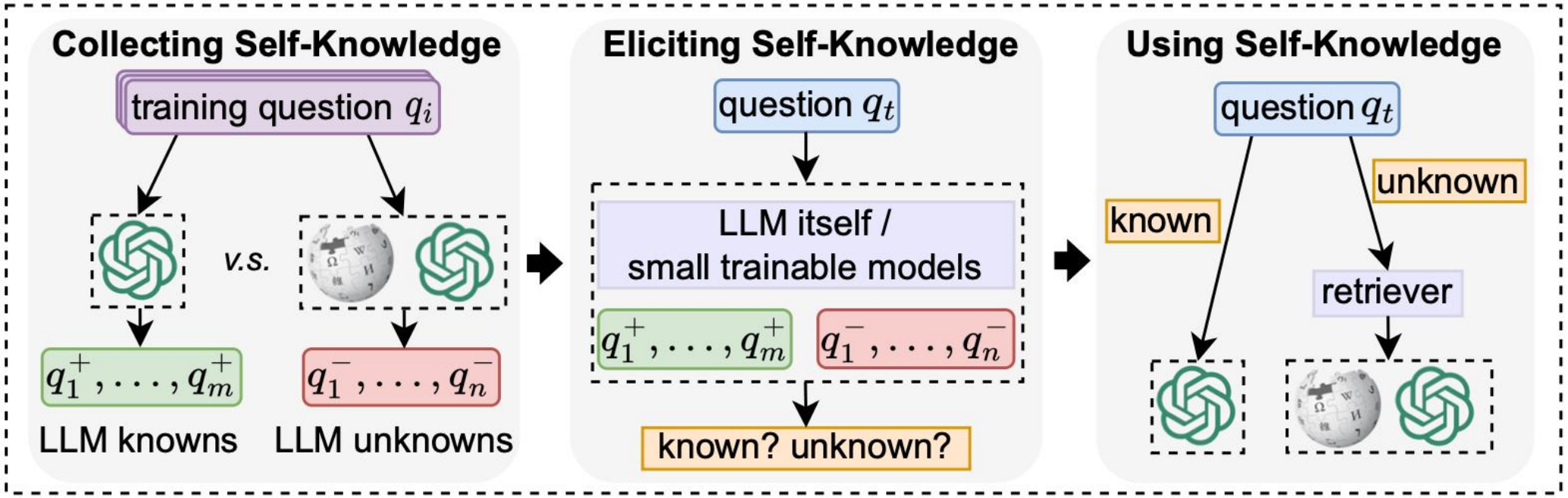



Figure 4: The detailed process of KnowledGPT's entity linking steps for the given example.

利用思维程序（PoT）提示方法，采用LLMs生成的Python代码作为搜索语言，检索到就用，检索不到就让LLMs回答

# Self-Knowledge Guided Retrieval Augmentation for Large Language Models

arXiv preprint arXiv:2310.05002, 2023, 202308 cite=19

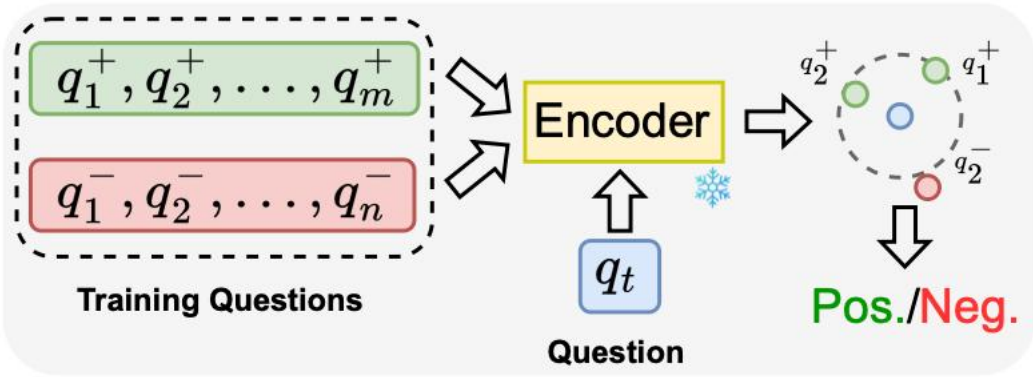



direct prompting
in-context learning
training a classifier
nearest neighbor search

**In-Context Learning**

(prompt)
{$q_1^+$} Q: *Do you need additional information to answer this question?* A: *No, I don't need additional information to answer this question.*
{$q_1^-$} Q: *Do you need additional information to answer this question?* A: *Yes, I need additional information to answer this question.*
.....
{$q_t$} Q: *Do you need additional information to answer this question?* A:

(possible response)
No, I don't need additional information to answer this question. / Yes, I need additional information to answer this question.



# Advanced Method 1: Small-to-big retrieval

(Using smaller chunks **enhances the accuracy** of retrieval, while larger chunks **offer more contextual information**.)

**Step1: Create Smaller Child Chunks**

For each of the text chunks with chunk size 1024, we create even smaller text chunks:

- 8 text chunks of size 128
- 4 text chunks of size 256
- 2 text chunks of size 512

**Step 2: Create Index, retriever, and query engine**

When we ask a question and retrieve the most relevant text chunks, it will actually retrieve the text chunk with the node id **pointing to the parent chunk** and **thus retrieve the parent chunk.**

# Avanced Method 2: Sentence Window Retrieval

Parse the documents into **a single sentence per chunk**, The sentence **"window"** (5 sentences on either side of the original sentence) will be similar to the "parent" chunk concept. In other words, we use the **single sentences during retrieval** and pass the retrieved sentence **with the sentence window to the LLM.**

# Precise Zero-Shot Dense Retrieval without Relevance Labels

arXiv preprint arXiv:2212.10496, 2022, 202212 cite=77

$$\text{sim}(q, d) = \langle \text{enc}_q(q), \text{enc}_d(d) \rangle = \langle \mathbf{v_q}, \mathbf{v_d} \rangle$$

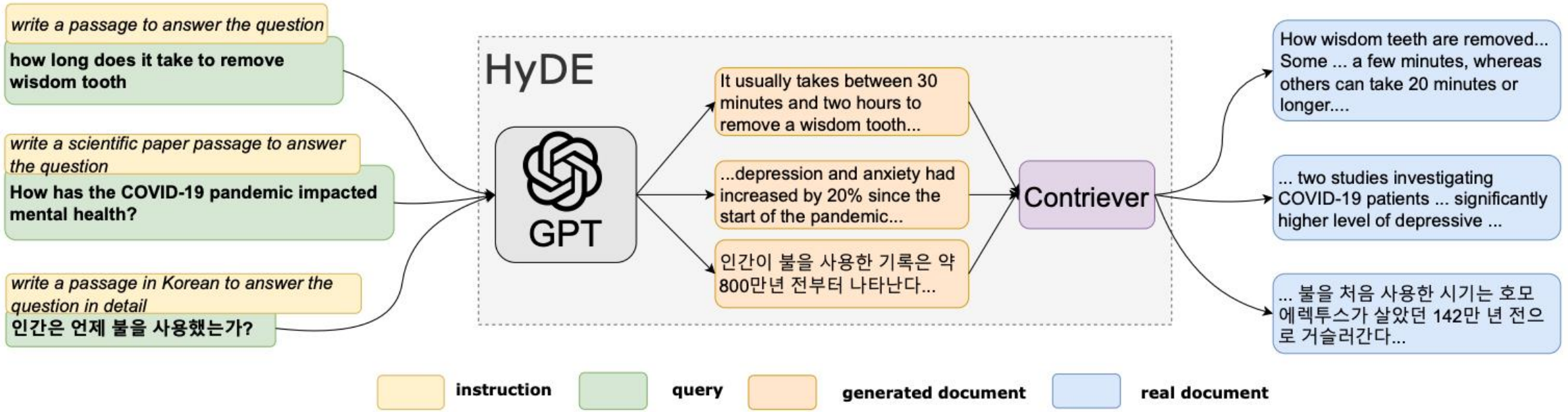



Figure 1: An illustration of the HyDE model. Documents snippets are shown. HyDE serves all types of queries without changing the underlying GPT-3 and Contriever/mContriever models.

$$g(q, \text{INST}) = \text{InstructLM}(q, \text{INST})$$

$$\mathbb{E}[\mathbf{v}_{q_{ij}}] = \mathbb{E}[f(g(q_{ij}, \text{INST}_i))]$$

$$\hat{\mathbf{v}}_{q_{ij}} = \frac{1}{N} \sum_{\hat{d}_k \sim g(q_{ij}, \text{INST}_i)} f(d_k)$$

$$= \frac{1}{N} \sum_{k=1}^{N} f(\hat{d}_k)$$

$$\hat{\mathbf{v}}_{q_{ij}} = \frac{1}{N+1} [\sum_{k=1}^{N} f(\hat{d}_k) + f(q_{ij})]$$

# Promptagator: Few-shot dense retrieval from 8 examples

*ICLR 2023*, 202209 cite=119

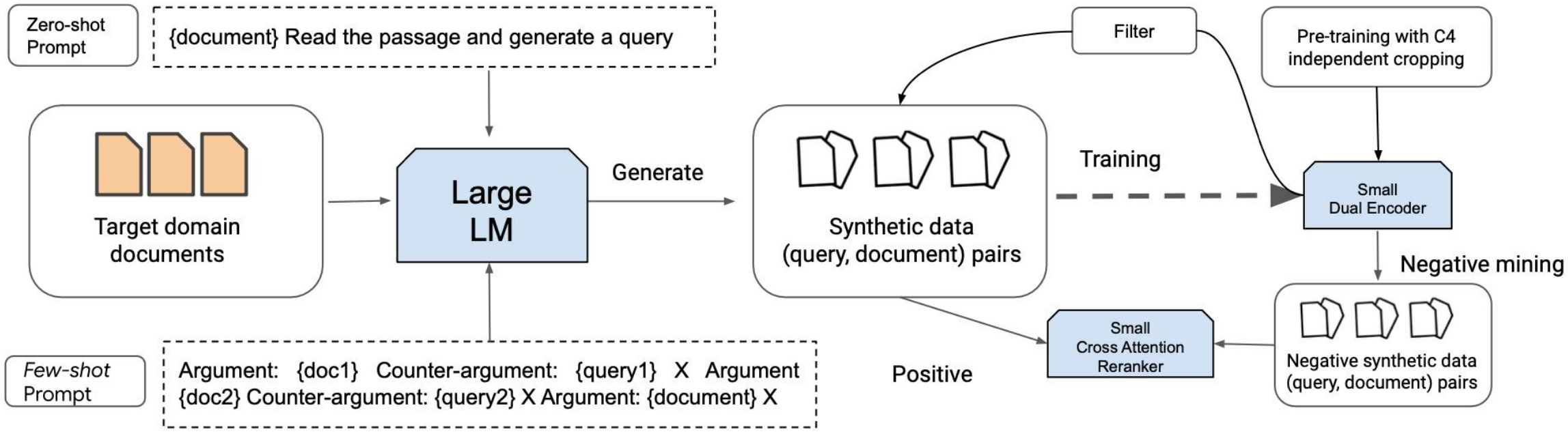



Figure 5: PROMPTAGATOR++ Training pipeline.

# Replug: Retrieval-augmented black-box lan- guage models

ACL2023 202303, cite=45

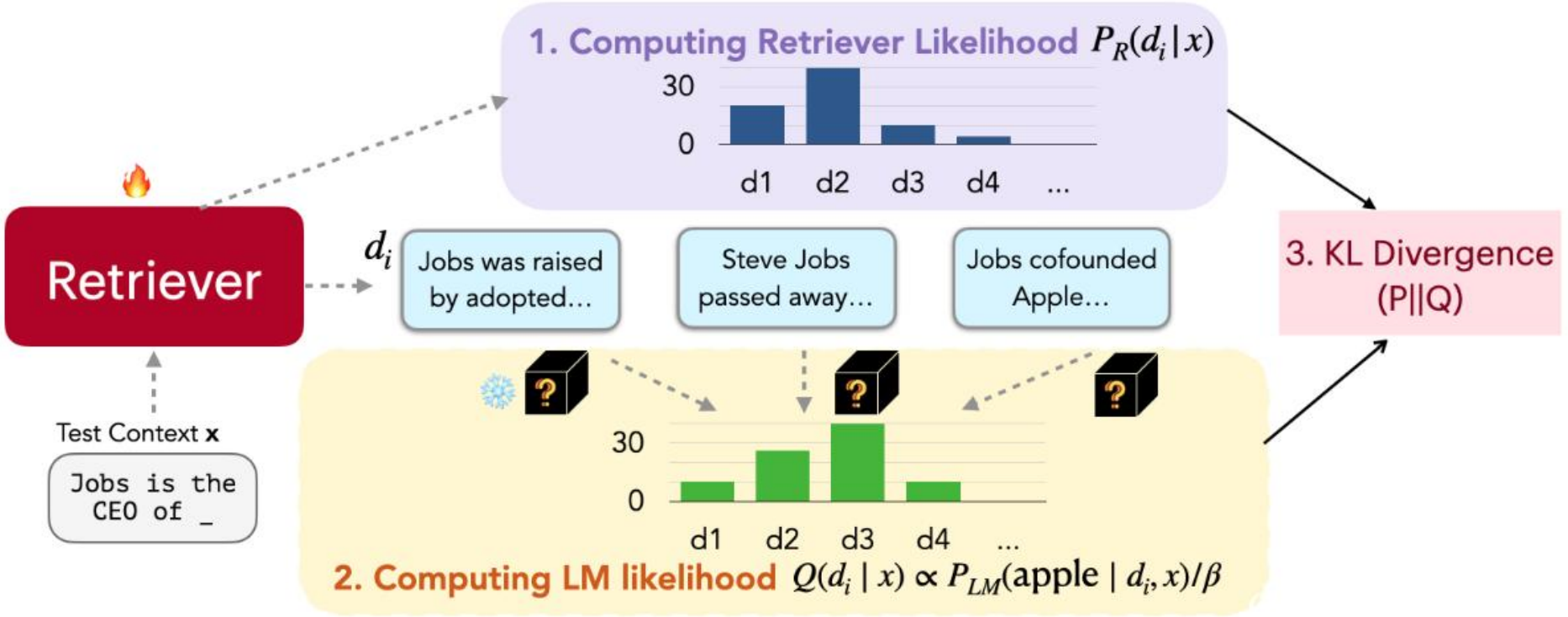



*Figure 3.* **REPLUG LSR training process (§4).** The retriever is trained using the output of a frozen language model as supervision signals.

$$P_R(d \mid x) = \frac{e^{s(d,x)/\gamma}}{\sum_{d \in \mathcal{D}'} e^{s(d,x)/\gamma}} \qquad Q(d \mid x, y) = \frac{e^{P_{LM}(y|d,x)/\beta}}{\sum_{d \in \mathcal{D}'} e^{P_{LM}(y|d,x)/\beta}} \qquad \mathcal{L} = \frac{1}{|\mathcal{B}|} \sum_{x \in \mathcal{B}} KL\Big(P_R(d \mid x) \parallel Q_{\text{LM}}(d \mid x, y)\Big),$$

# Replug: Retrieval-augmented black-box lan- guage models

ACL2023 202303, cite=45

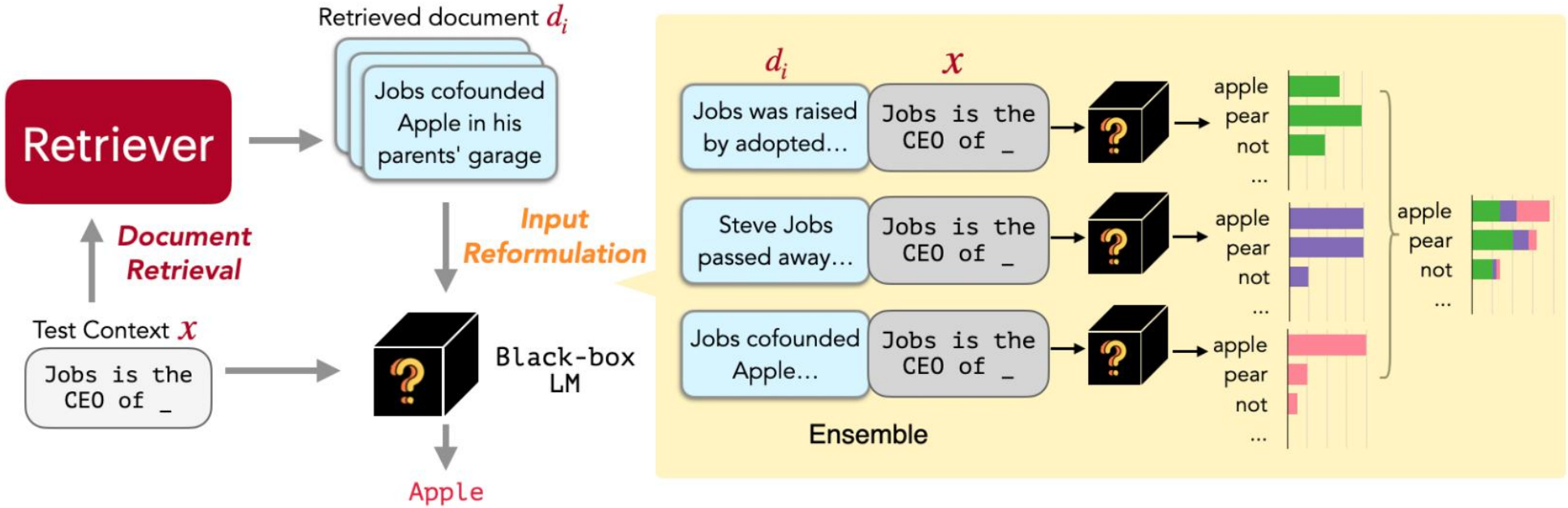



*Figure 2.* **REPLUG at inference** (§3). Given an input context, REPLUG first retrieves a small set of relevant documents from an external corpus using a retriever (§3.1 *Document Retrieval*). Then it prepends each document separately to the input context and ensembles output probabilities from different passes (§3.2 *Input Reformulation*).

$$p(y \mid x, \mathcal{D}') = \sum_{d \in \mathcal{D}'} p(y \mid d \circ x) \cdot \lambda(d, x),$$

$$\lambda(d, x) = \frac{e^{s(d,x)}}{\sum_{d \in \mathcal{D}'} e^{s(d,x)}}$$

# Retrieve Anything To Augment Large Language Models

202310, cite=32

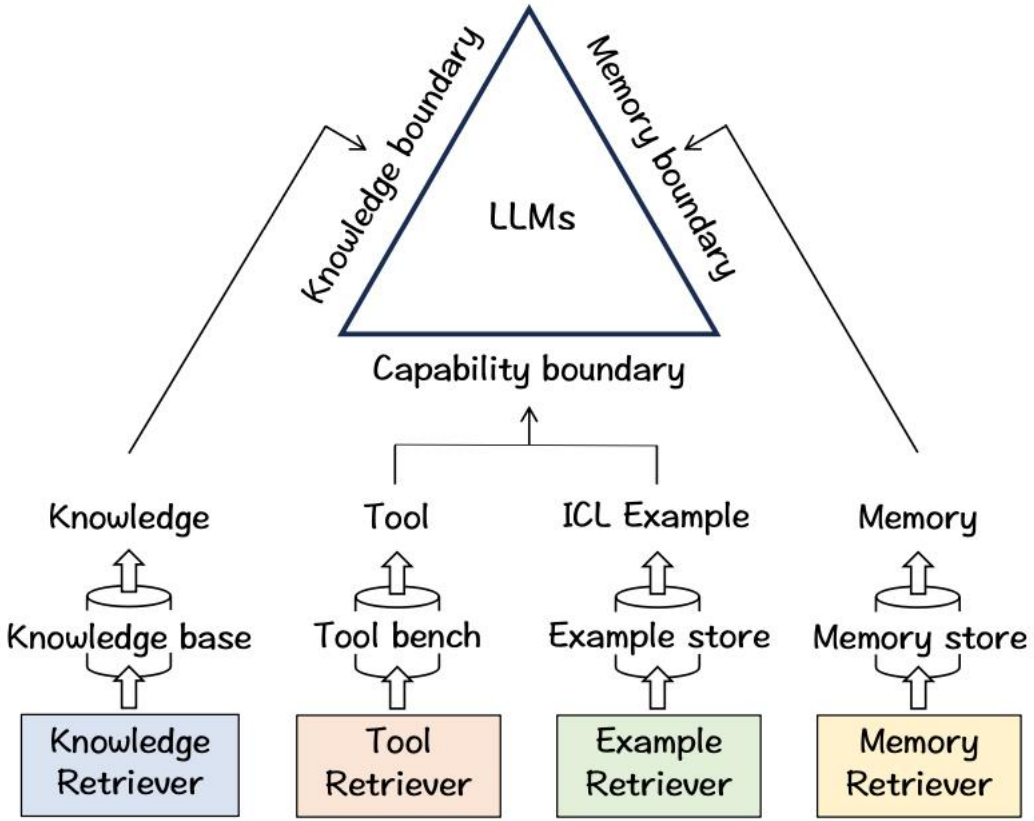



**Figure 1: Confront the threefold inherent boundaries of LLMs on top of retrieval augmentation.**

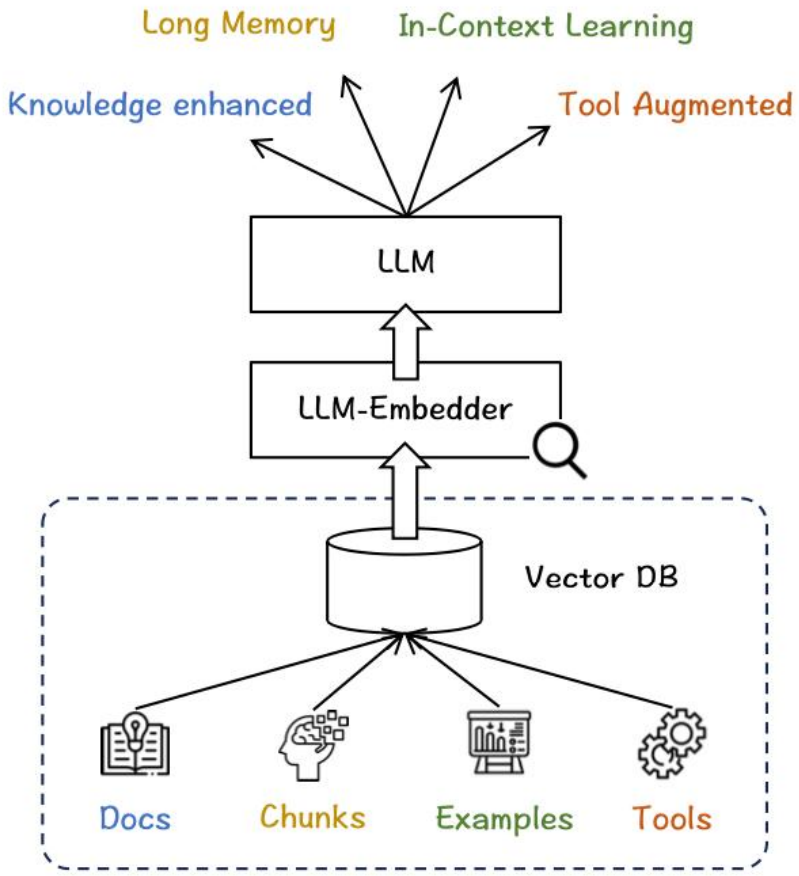



**Figure 2: Retrieval augmentation with LLM-Embedder.**

# Open-source Large Language Models are Strong Zero-shot Query Likelihood Models for Document Ranking

202310, cite=11

$$S_{QLM}(\boldsymbol{q}, d) = \frac{1}{|\boldsymbol{q}|} \sum_{t} \log \mathrm{LLM}(q_t | \boldsymbol{p}, d, \boldsymbol{q}_{<t}) \quad (1)$$

$$S(\boldsymbol{q}, d) = \alpha \cdot S_{BM25}(\boldsymbol{q}, d) + (1 - \alpha) \cdot S_{QLM}(\boldsymbol{q}, d), \quad (2)$$

BM25：每个文档的得分基于查询中的词与该文档中的词的匹配程度来计算

利用LLMs作为查询概率模型（QLMs）。本质上，QLMs应该理解文档和查询的语义，并估计每个文档回答某个查询的可能性。

不仅依赖LLMs估计的查询可能性分数，还通过使用加权分数和线性插值QLM分数与第一阶段检索器的BM25分数

$$\mathrm{Score}(D, Q) = \sum_{i \in Q} \mathrm{IDF}(q_i) \times \frac{\mathrm{TF}(q_i, D) \times (k_1 + 1)}{\mathrm{TF}(q_i, D) + k_1 \times (1 - b + b \times \frac{|D|}{\mathrm{avgdl}})}$$

# Llmlingua: Compressing prompts for accelerated inference of large language models

202312, cite=37

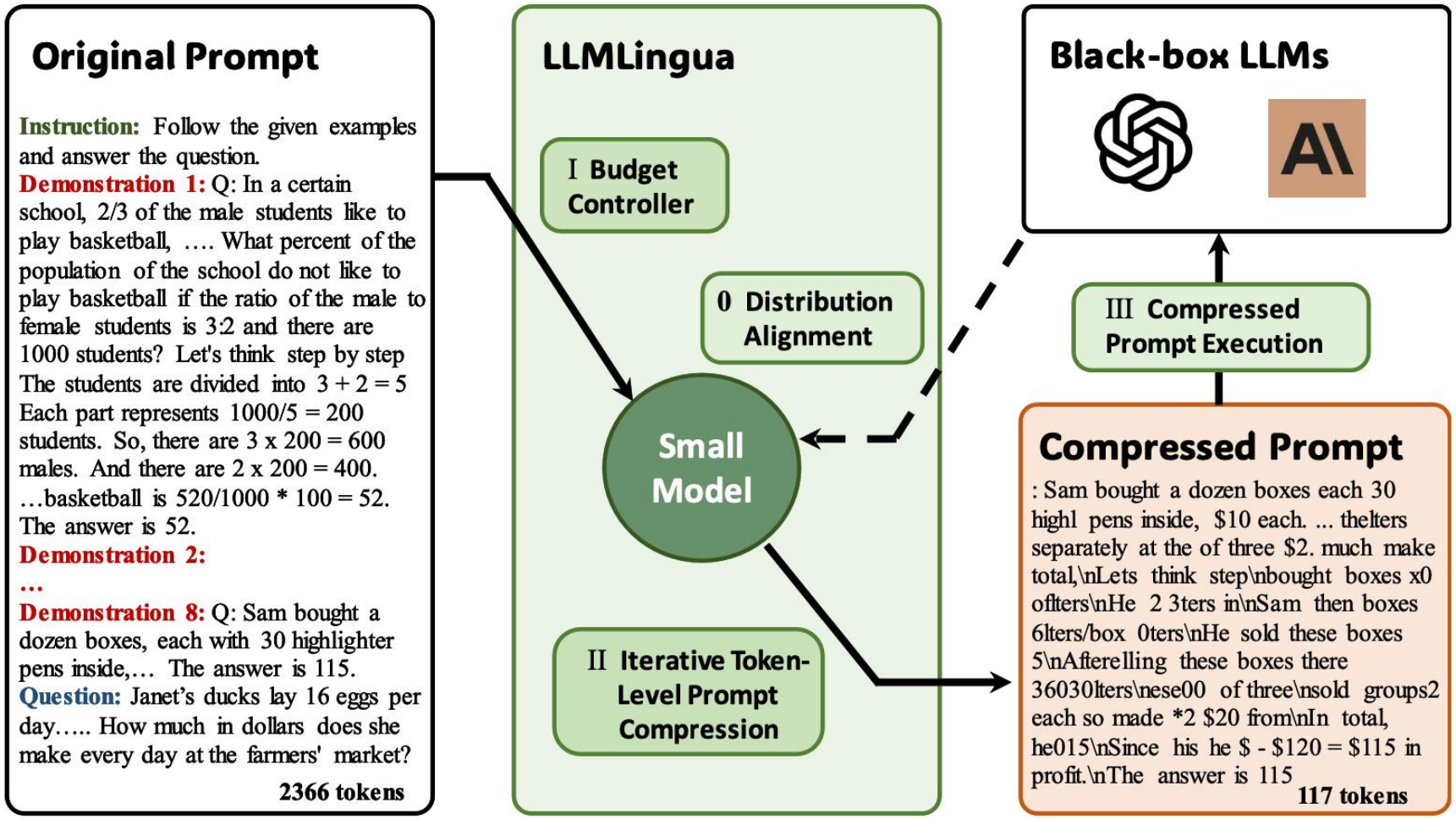



Figure 1: Framework of the proposed approach *LLMLingua*.

$$\text{Perplexity} = 2^{-\sum_i p(x_i|x_{<i}) \log_2 p(x_i|x_{<i})} \qquad r_k = 1/N_k \sum_i^{N_k} p(x_{k,i}^{\text{doc}}) \log p(x_{k,i}^{\text{doc}}), k \in \{1, 2, \cdots, K\}$$

# Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression

202310, cite=25

$$\min_{\widetilde{\mathbf{x}}} D\left(\mathbf{y}, \widetilde{\mathbf{y}}\right) + \lambda \|\widetilde{\mathbf{x}}\|_0,$$

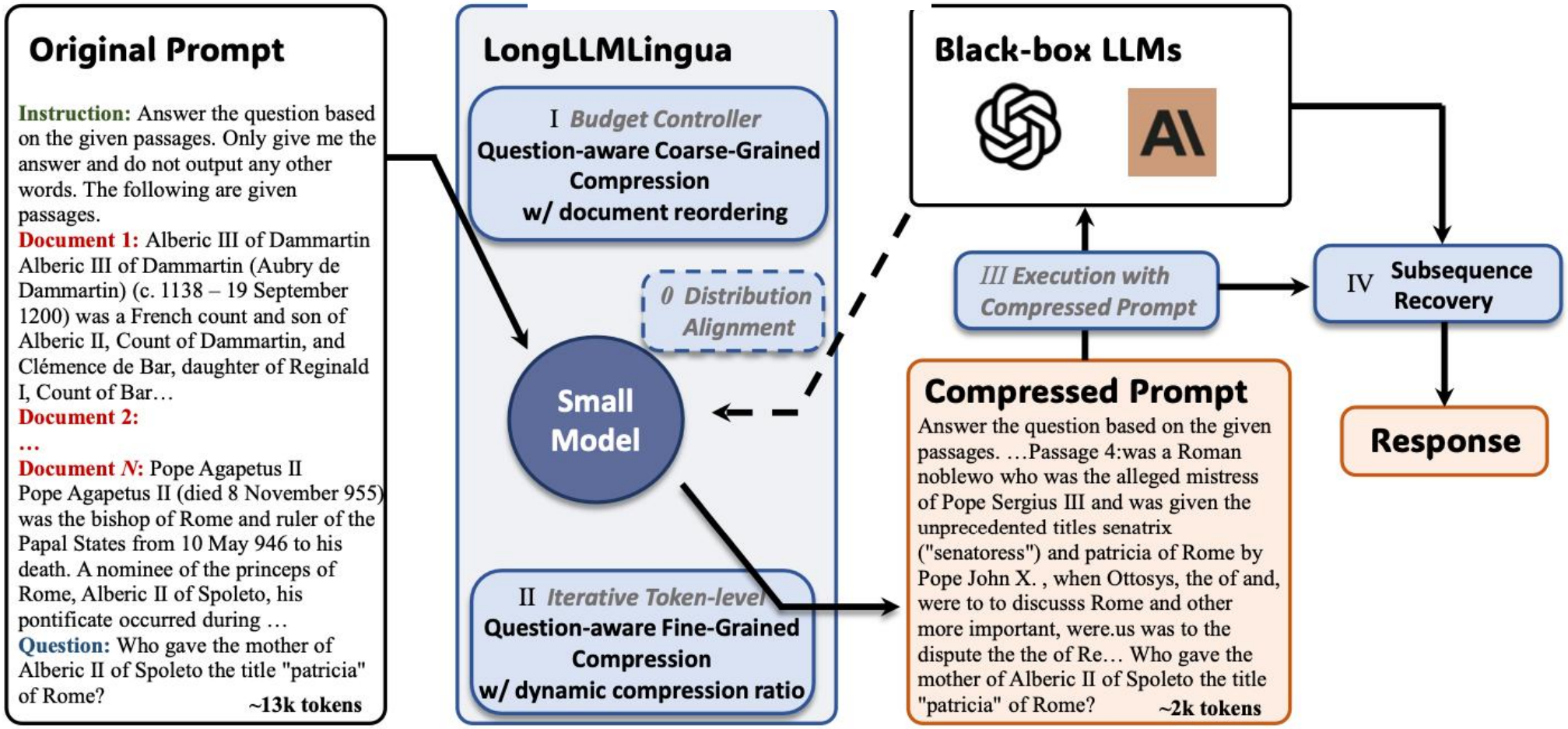
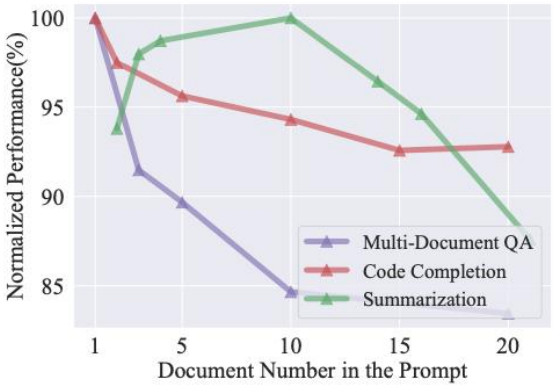


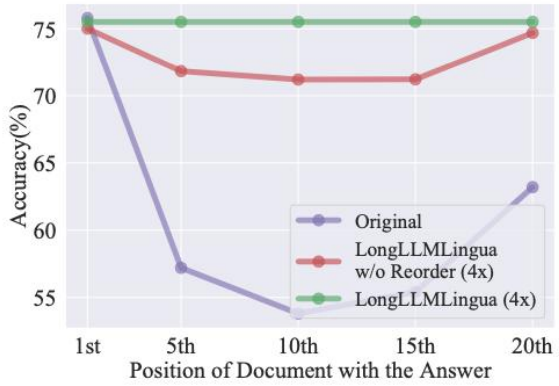

Figure 2: Framework of *LongLLMLingua*. Gray *Italic* content: As in LLMLingua.

| | | |
|---|---|---|
| Document [1](Title: List of Nobel laureates in Physics) The first Nobel Prize in Physics was awarded in 1901 to {Wilhelm Conrad Röntgen}{Wilhelm Con rad Rö nt gen}, of Germany,… **Original Prompt** | Document [1](Title: List of Nobelates in Physics) The first Nobel1 {Wilhelmgen}{Wilhelm gen}, of, who received, …. **Compressed Prompt** | {Wilhelmgen} {Wilhelm gen} **LLMs' Response** |

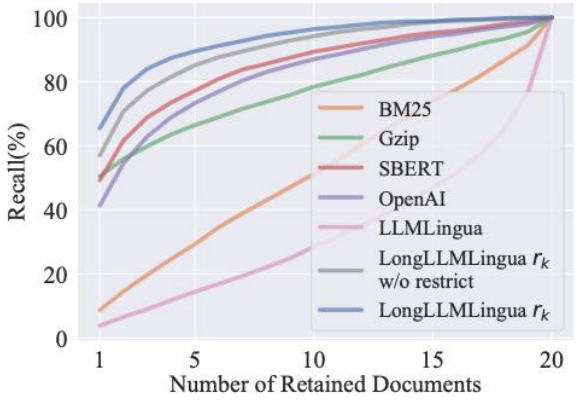
# Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression



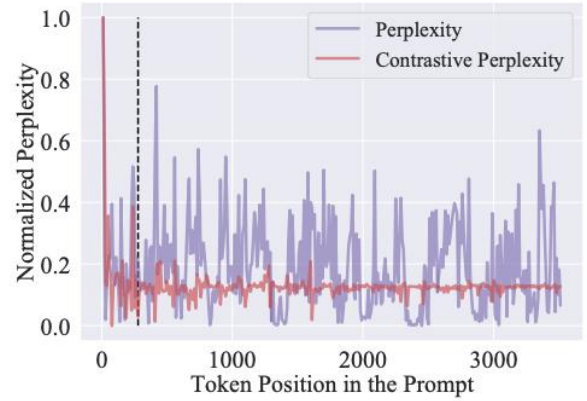

(a) Performance v.s. Document Number




(b) Performance v.s. Key Information Position




(a) Recall Distribution




(b) Perplexity Distribution

$$r_k = \frac{1}{N_c}\sum_{i}^{N_c} p(x_i^{\text{que,restrict}}|\mathbf{x}_k^{\text{doc}}) \log p(x_i^{\text{que,restrict}}|\mathbf{x}_k^{\text{doc}}), k \in \{1, 2, \cdots, K\},$$

$$s_i = \text{perplexity}(x_i|x_{<i}) - \text{perplexity}(x_i|x^{\text{que}}, x_{<i}).$$

$$(\mathbf{x}^{\text{ins}}, \mathbf{x}_1^{\text{doc}}, \cdots, \mathbf{x}_{K'}^{\text{doc}}, \mathbf{x}^{\text{que}}) \xrightarrow{r_k} (\mathbf{x}^{\text{ins}}, \mathbf{x}_{r1}^{\text{doc}}, \cdots, \mathbf{x}_{rK'}^{\text{doc}}, \mathbf{x}^{\text{que}})$$

$$\tau_i = \tau_k^{\text{doc}}, \quad x_i \in \mathbf{x}_k^{\text{doc}},$$

$$\tau_k^{\text{doc}} = \max(\min((1 - \frac{2I(r_k)}{N_d})\delta\tau + \tau^{\text{doc}}, 0), 1),$$

**Algorithm 1** Pseudo code of Token-level Subsquence Recovery.

**Input**: The original prompt $\boldsymbol{x}$; the compressed prompt $\widetilde{\boldsymbol{x}}$; the generation response of LLMs $\boldsymbol{y}$.

Set the final response list $\boldsymbol{y}_{\text{rec}} = \phi$, the left token index of subsquence $l$ to 0.
**while** $l < \boldsymbol{y}.len()$ **do**
  **if** Substring $y_l \in \widetilde{\boldsymbol{x}}$ **then**
    Find the longer substring $\widetilde{\boldsymbol{y}}_{\text{key},l} = \{y_l, y_{l+1}, ..., y_r\} \in \widetilde{\boldsymbol{x}}$.
    Find the maximum common shortest subsequence $\boldsymbol{x}_{i,j} = \{x_i, x_{i+1}, ..., x_j\}$ in the original prompt $\boldsymbol{x}$.
    Add the subsequence $\boldsymbol{x}_{i,j} = \{x_i, x_{i+1}, ..., x_j\}$ to the response $\boldsymbol{y}_{\text{rec}}$.
    Set the left index $l$ to $r + 1$.
  **else**
    Add the token $y_l$ to the response $\boldsymbol{y}_{\text{rec}}$.
    Set the left index $l$ to $l + 1$.
  **end if**
**end while**

**Output**: The final response list $\boldsymbol{y}_{\text{rec}}$.

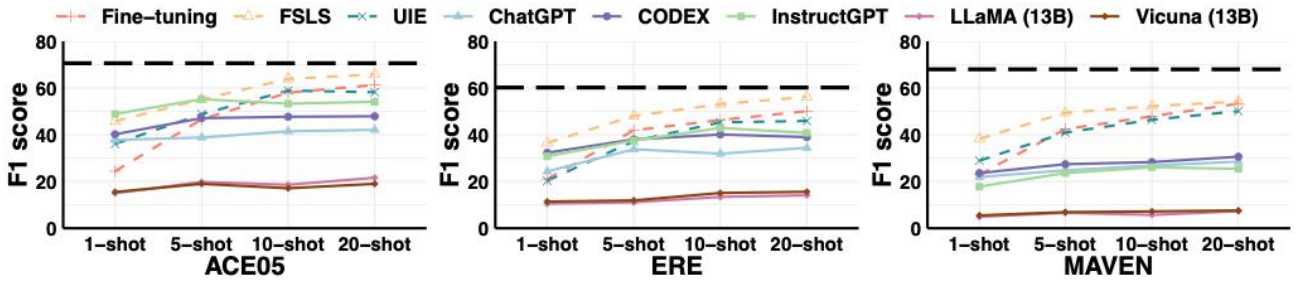
# Large Language Model Is Not a Good Few-shot Information Extractor, but a Good Reranker for Hard Sample

202310, cite=64

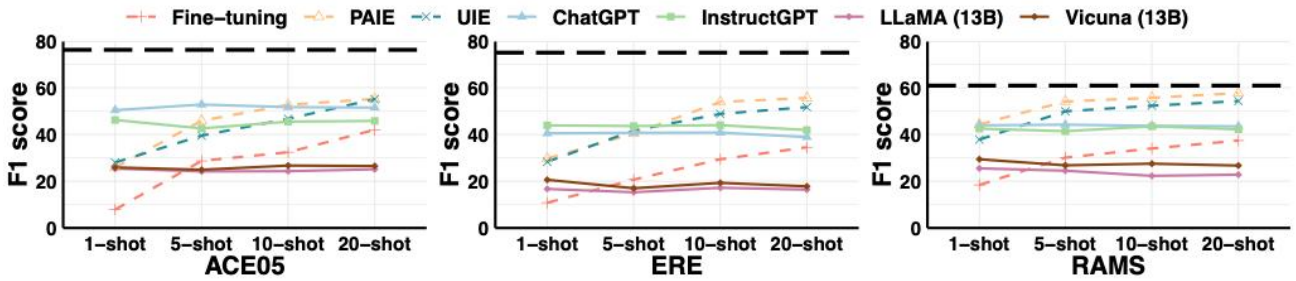
针对任务：（1）命名实体识别，（2）关系抽取，（3）事件检测和（4）事件参数抽取

当标注极为有限时，即标签类型和每个标签的样本极为稀少时，LLMs才会优于SLMs。
当样本较多（例如数百个）时，SLMs明显优于LLMs。
（1）SLMs通常优于LLMs，特别是在有更多训练样本和细粒度标签的情况下
（2）SLMs在时间和成本效率上都更高。
（3）在挑战SLMs的困难样本上，LLMs作为强大的重新排序器。

$$\text{conf}(x) = \max_{l \in L} P_{SLM}(l|x; s)$$




(c) Event Detection (ED)




(d) Event Argument Extraction (EAE)

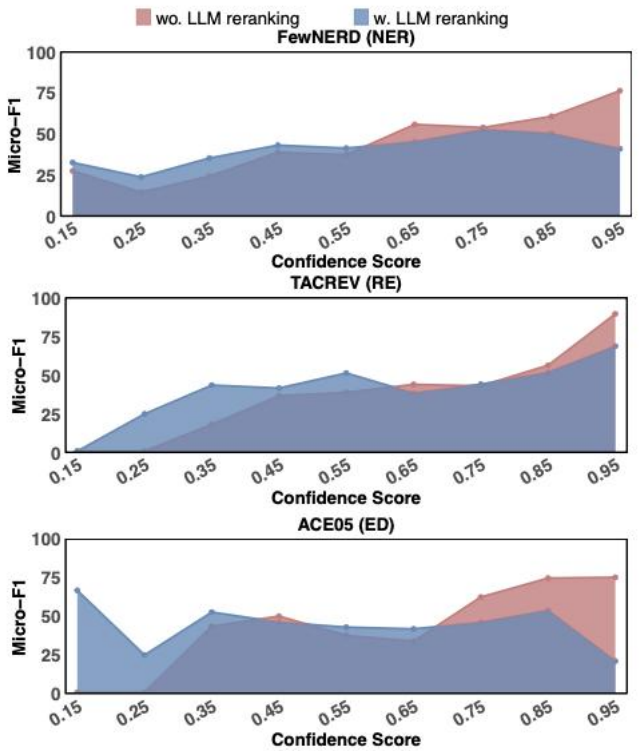



Figure 5: Relationship between confidence scores and performance with/without LLM reranking. We adopt `RoBERTa-large` as filter and InstructGPT as reranker.

# Large Language Model Is Not a Good Few-shot Information Extractor, but a Good Reranker for Hard Sample

202310, cite=64

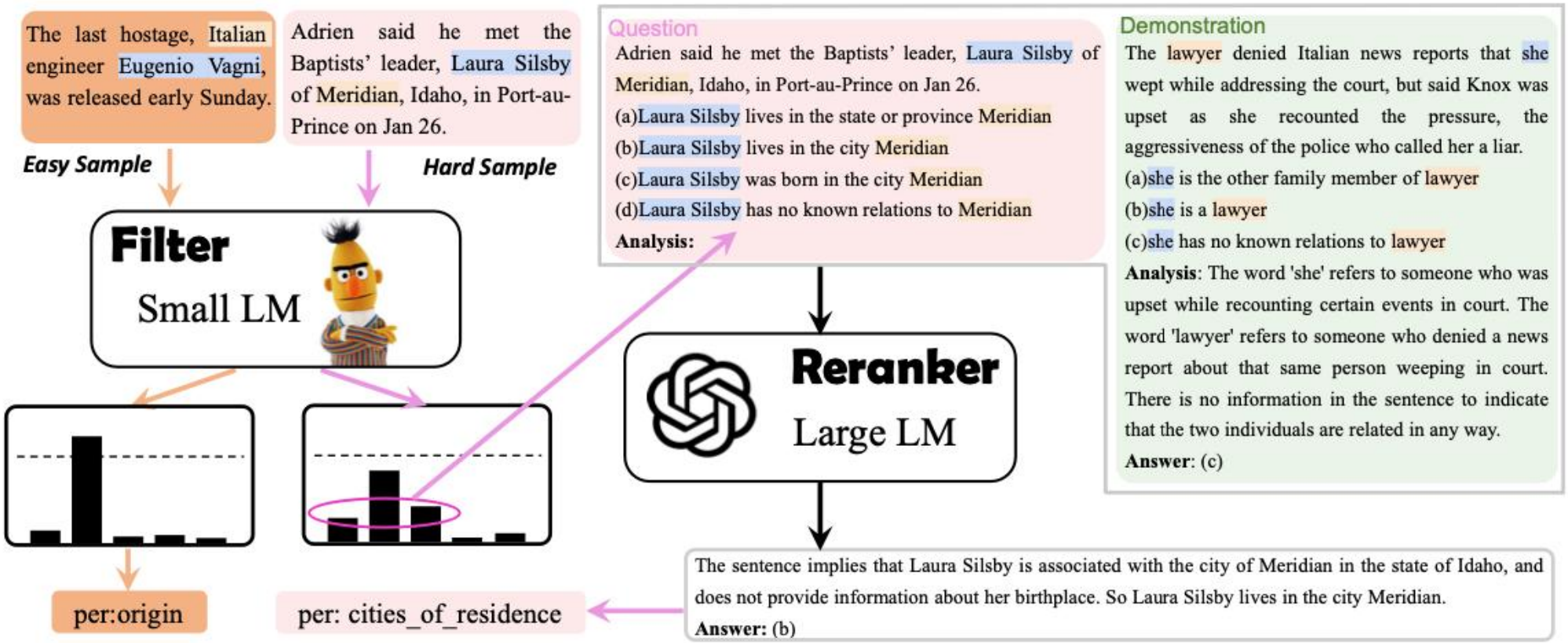



Figure 6: The overall architecture of our adaptive *filter-then-rerank* paradigm. We color easy samples in orange and hard samples in pink. For easy samples, the final predictions are exactly from the SLM-based methods. For hard samples, the top-$N$ predictions from SLMs are fed into LLMs as the format of multiple-choice questions (pink box). The question is paired with demos (green box). LLMs rerank these $N$ candidates and generate the final prediction.