41st IEEE International Conference on Data Engineering

HONG KONG SAR, CHINA | MAY 19 – 23, 2025

THE HONG KONG POLYTECHNIC UNIVERSITY 香港理工大學

Towards Retrieval-Augmented Large Language Models: Data Management and System Design

Website: https://shorturl.at/j5lGX

Survey: https://arxiv.org/pdf/2405.06211

Pangjing Wu, Yujuan Ding, Liangbo Ning, Shijie Wang,

Wenqi Fan, and Qing Li

The Hong Kong Polytechnic University

May 20th (Day 2), 14:00-17:00

ICDE 2025, Hong Kong SAR

Survey

Website

Tutorial Outline

- ⦿ **Part 1: Introduction of Retrieval Augmented Large Language Models (RA-LLMs) (Dr. Yujuan Ding)**
- ○ **Part 2: Architecture** of RA-LLMs and **Main Modules** (Dr. Yujuan Ding)
- ○ **Part 3: Data Management** for RA-LLMs (Pangjing Wu)
- ○ **Part 4: Learning** Approach of RA-LLMs (Liangbo Ning)
- ○ **Part 5: Applications** of RA-LLMs (Shijie Wang)
- ○ **Part 6: Challenges and Future Directions** of RA-LLMs (Liangbo Ning)

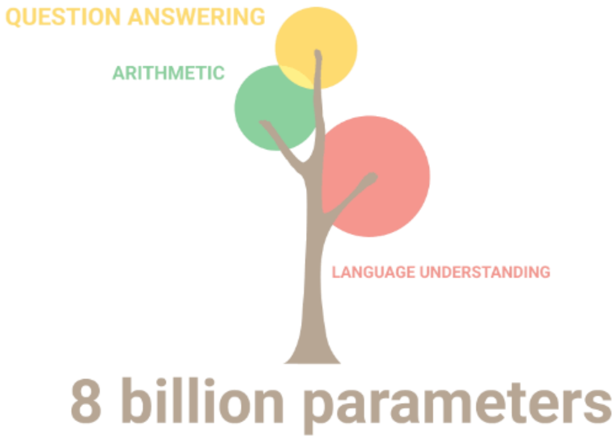
Website of this tutorial
Check out the slides and more information!

Website

Large Language Models (LLMs)

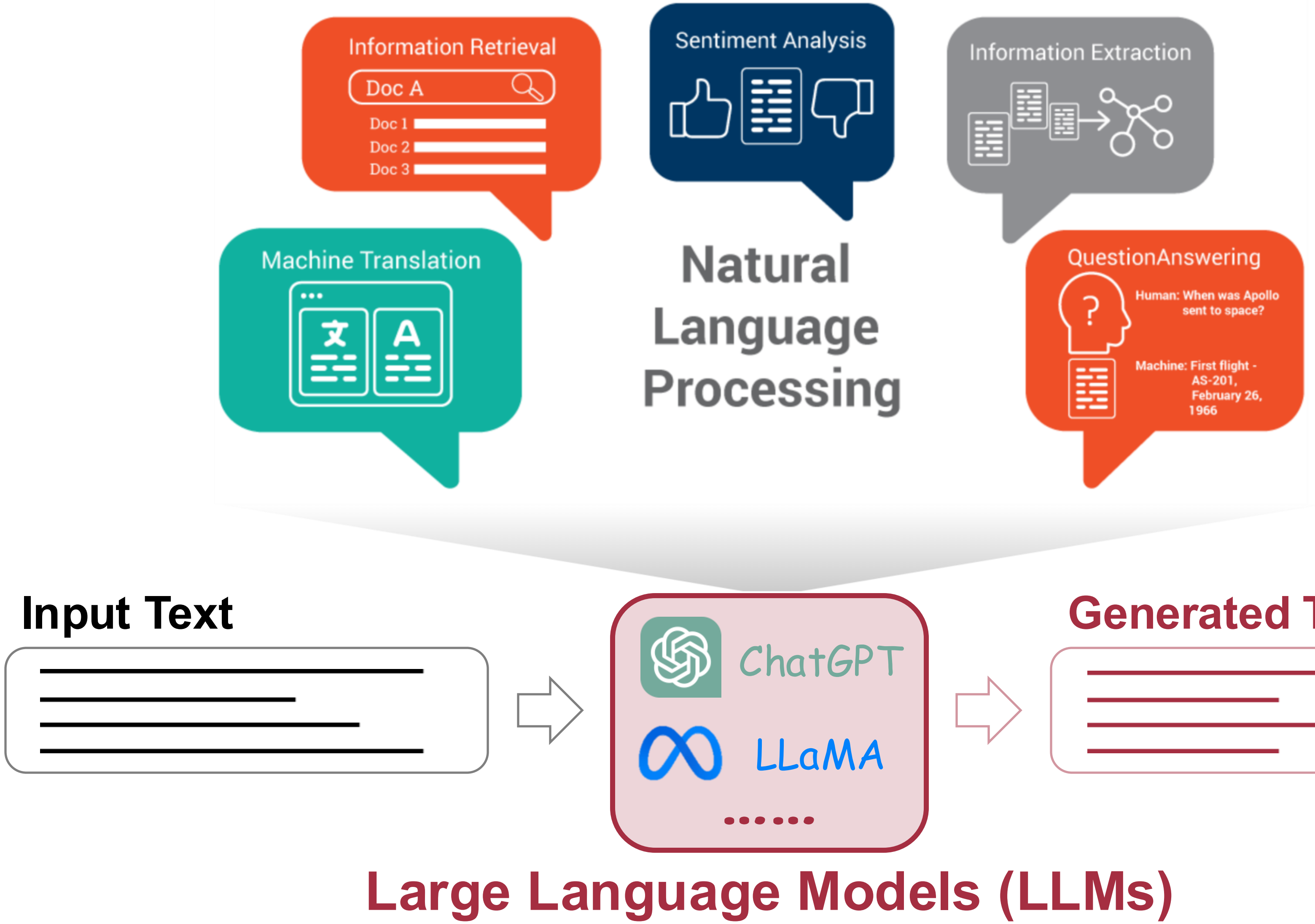
OpenAI | Meta | Google AI | Gemini

deepseek | Qwen | ......

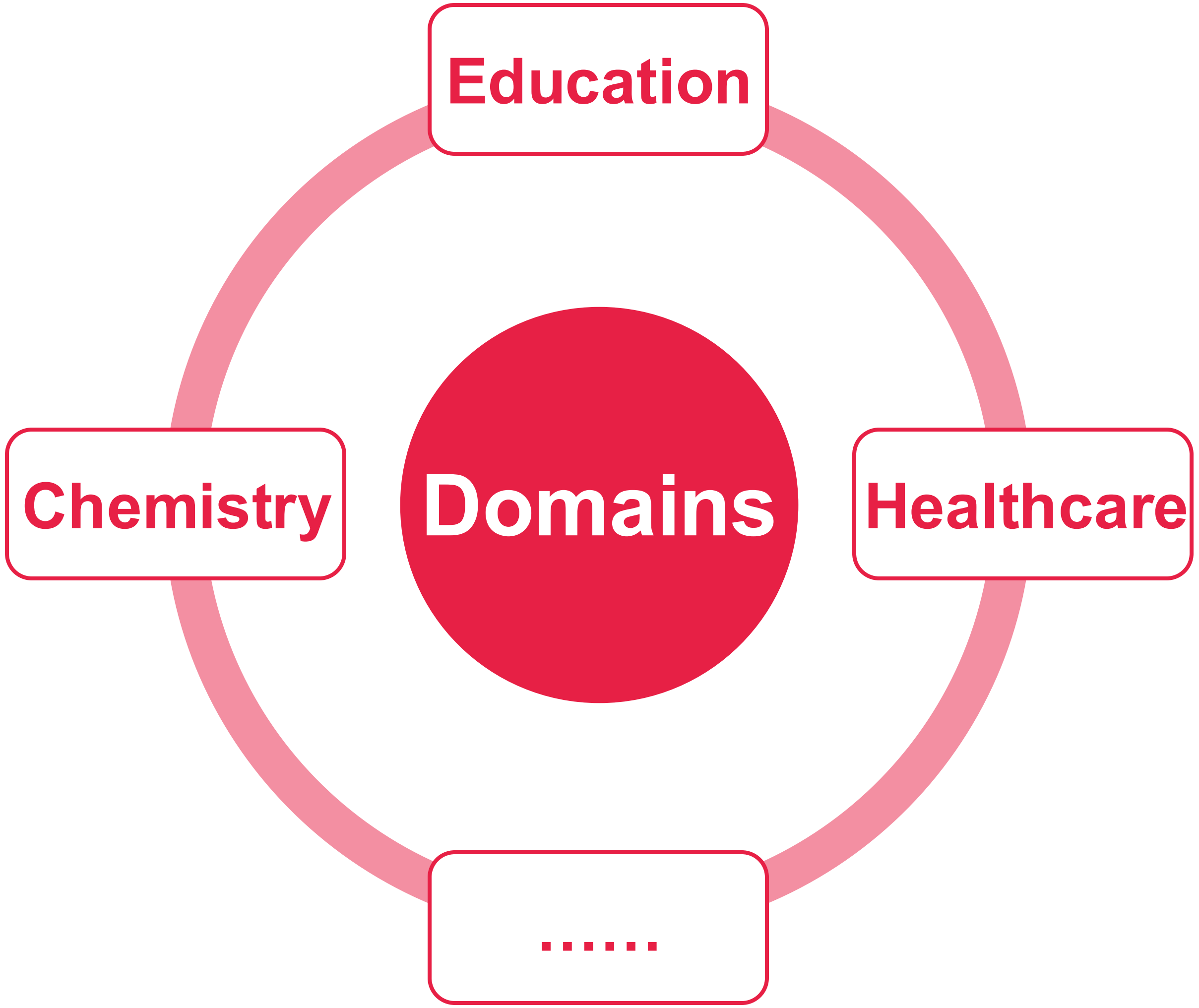


https://github.com/Hannibal046/Awesome-LLM/tree/main

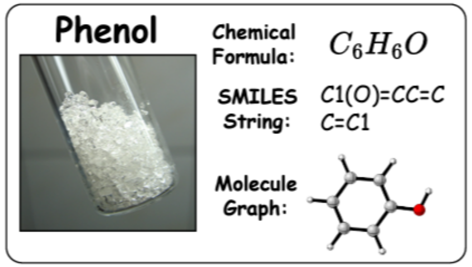
Capability of Large Language Models (LLMs)



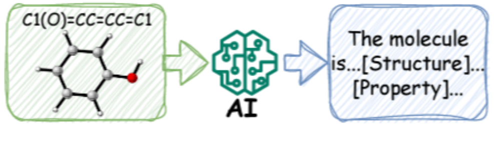
LLMs in Downstream Domains



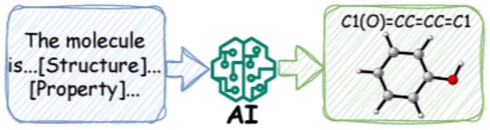
- **Molecule discovery**, etc.

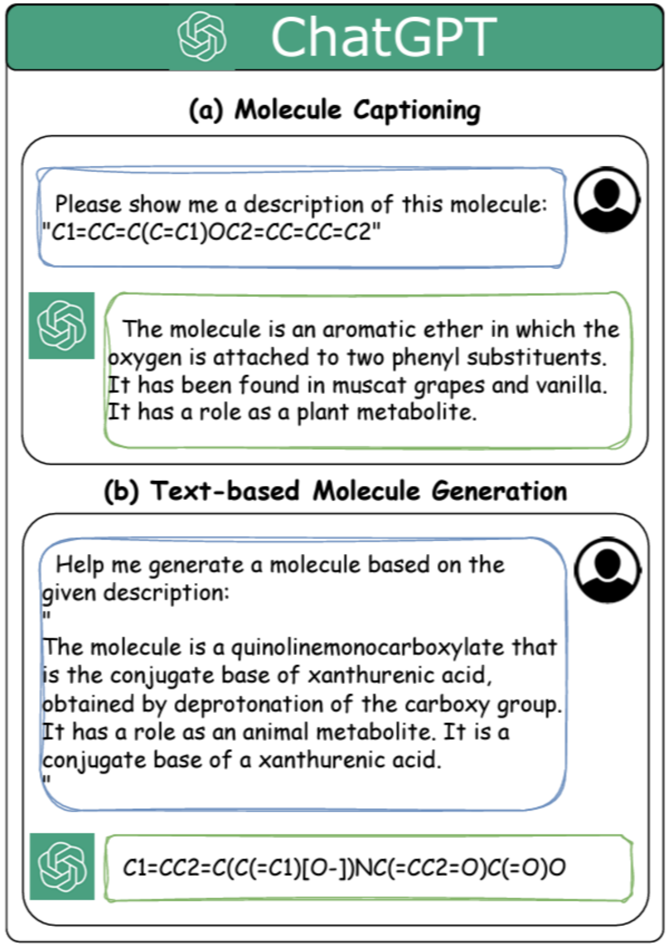


(a) Molecule Representations.



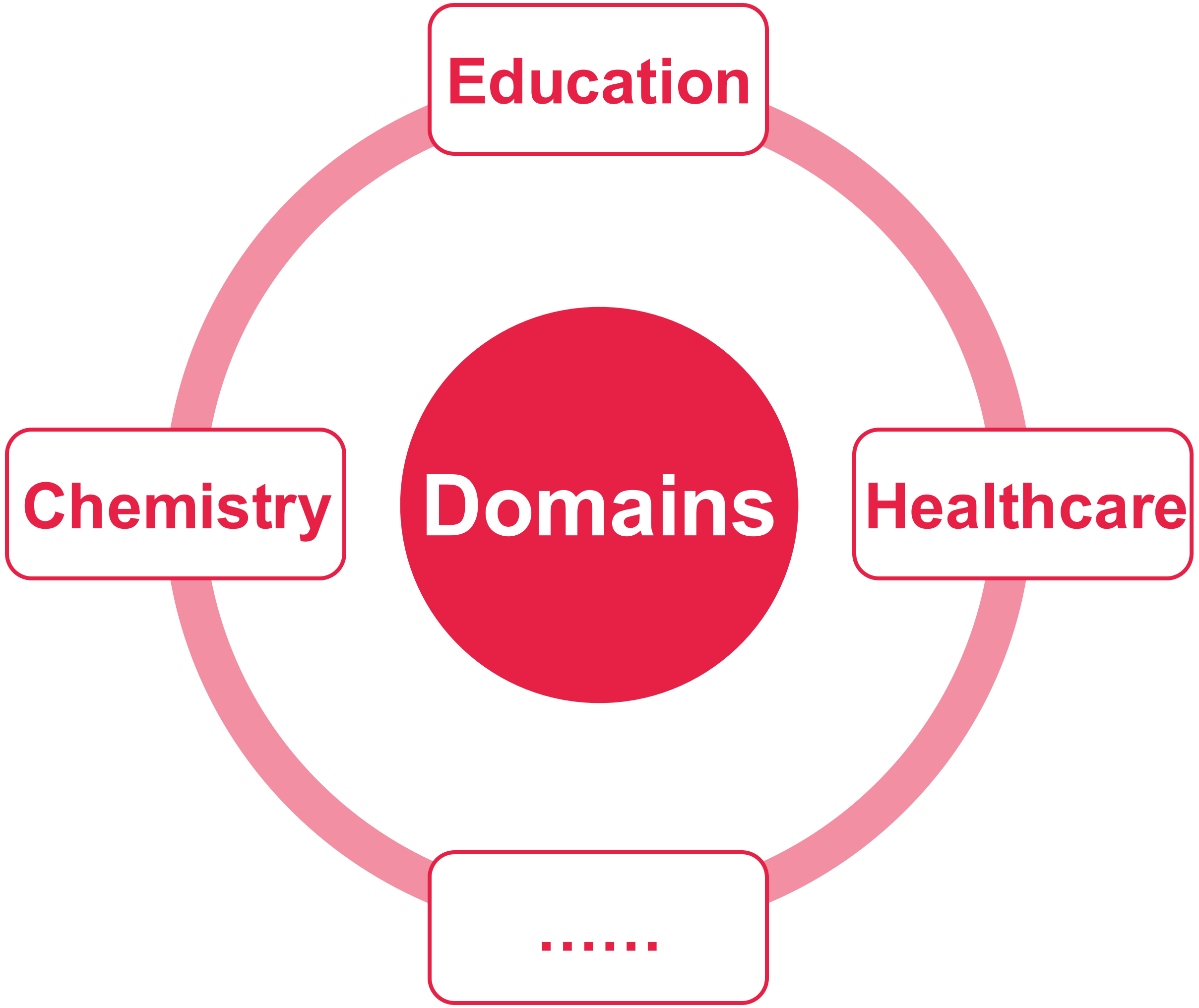
(b) Molecule Captioning.



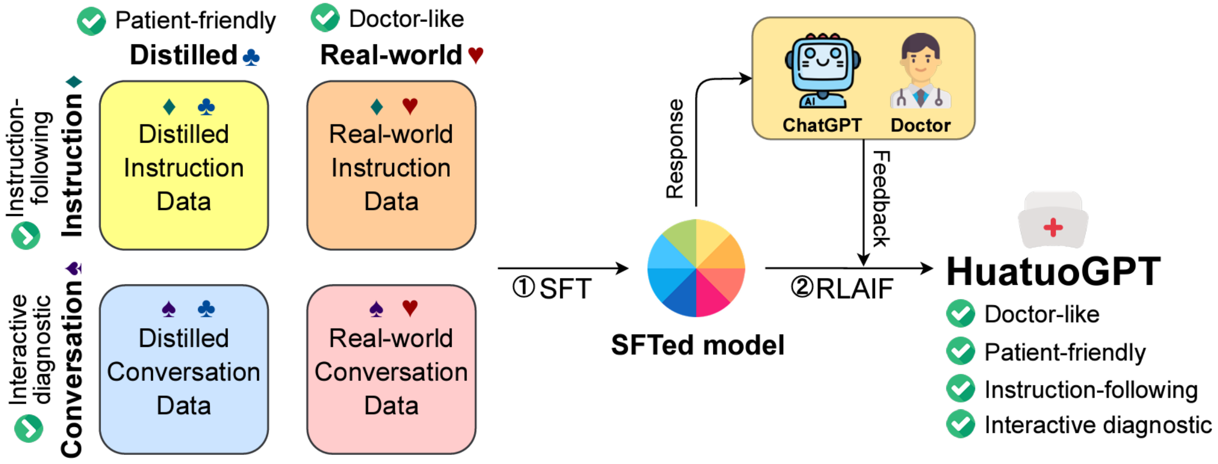


Li et al, 2024, Empowering Molecule Discovery for Molecule-Caption Translation with Large Language Models: A ChatGPT Perspective,
Liu et al., 2024, MolecularGPT: Open Large Language Model (LLM) for Few-Shot Molecular Property Prediction,

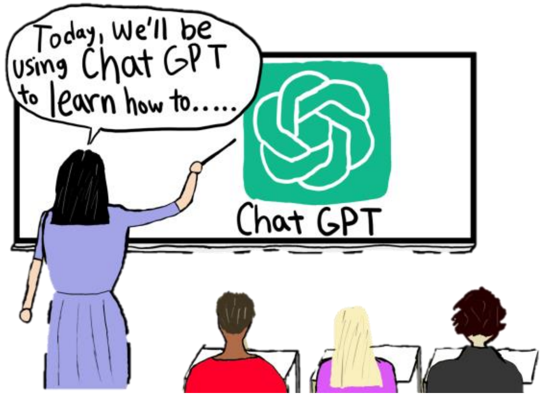
LLMs in Downstream Domains



❑ **Medical consultation**, etc.

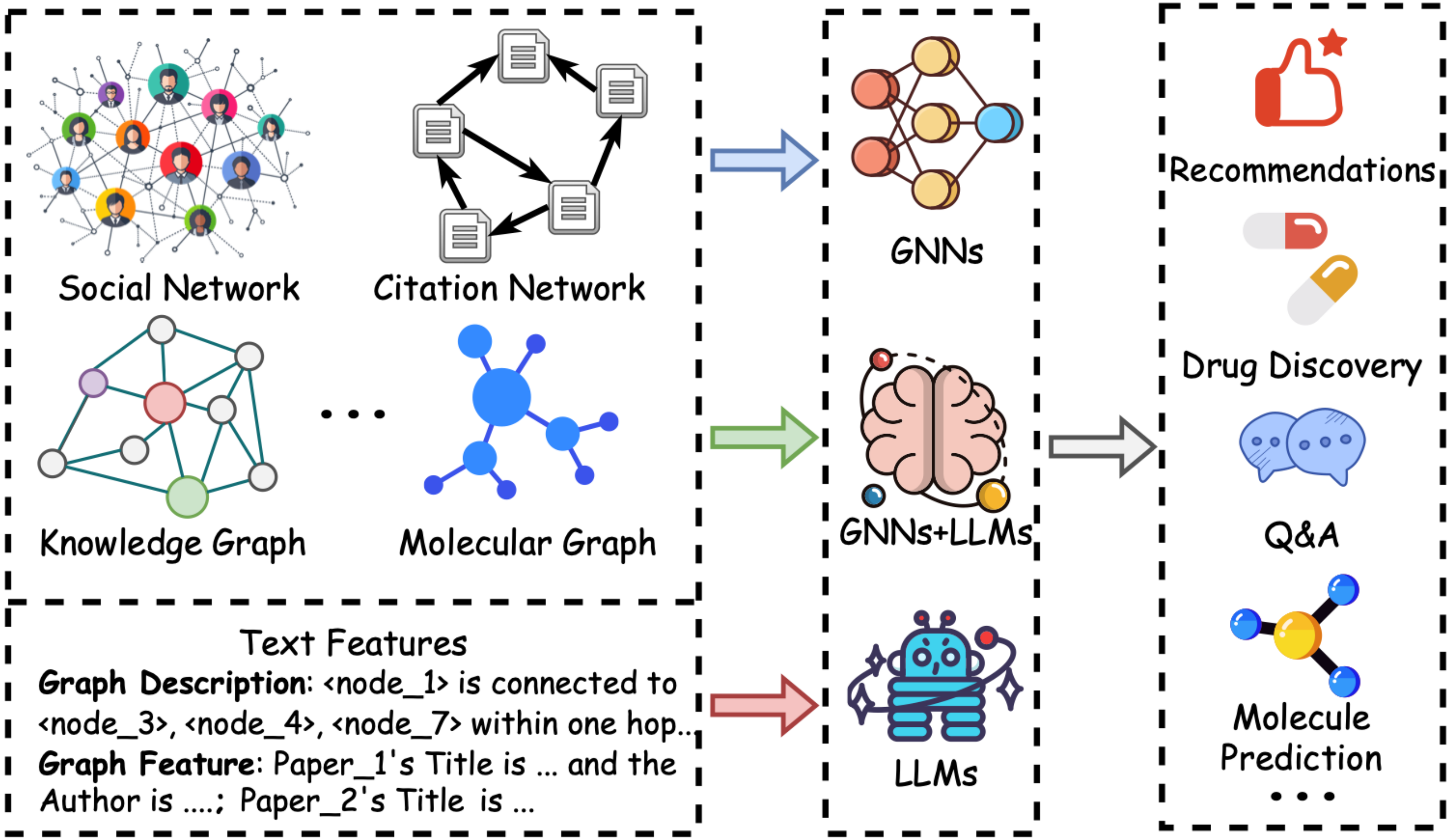


❑ **Curriculum & Teaching**, etc.



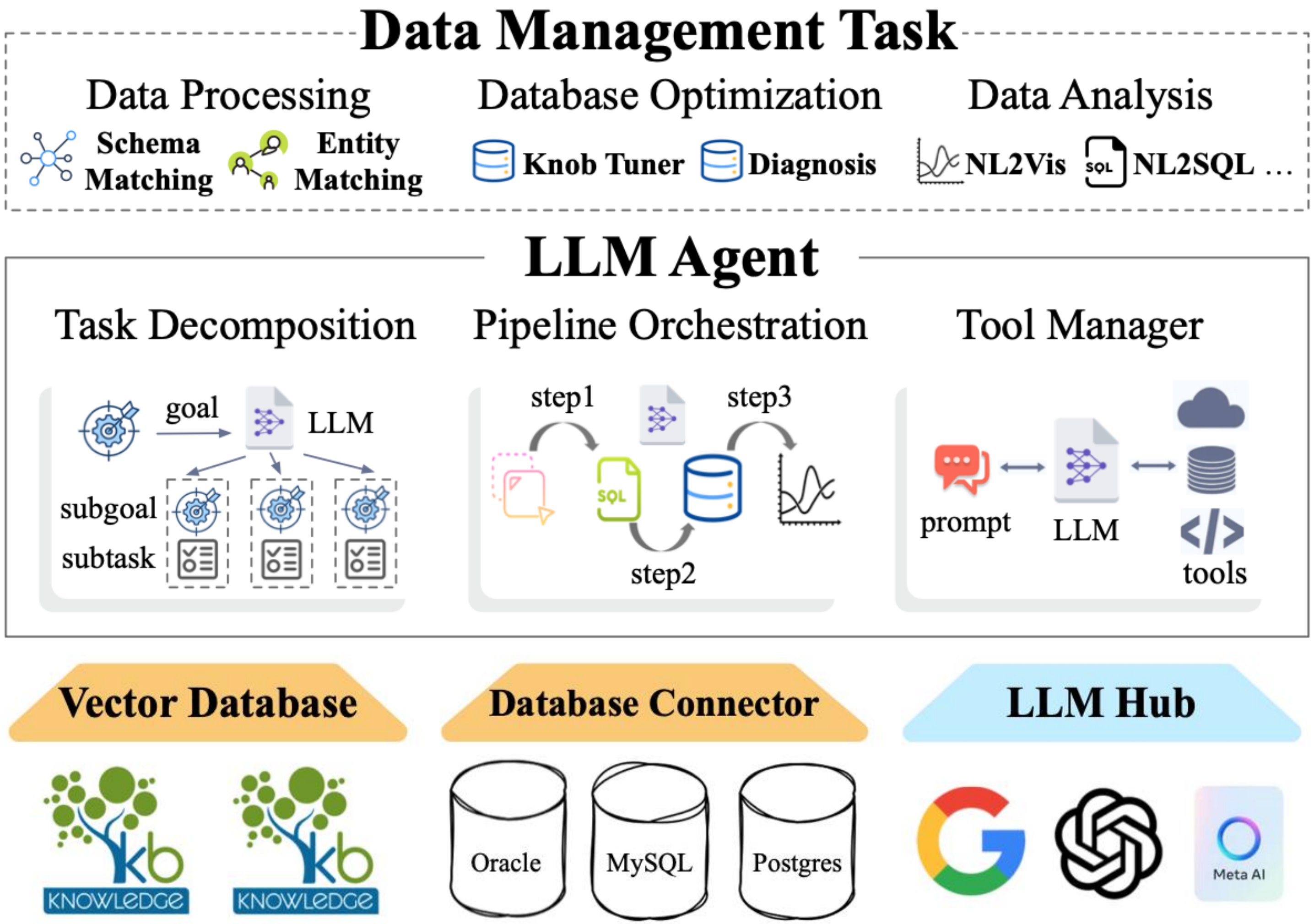
Zhang et al., 2023, HuatuoGPT, towards Taming Language Model to Be a Doctor

LLMs on Graph-structured Data



Fan, et al., 2024, Graph Machine Learning in the Era of Large Language Models (LLMs).

LLMs in Data Engineering Applications

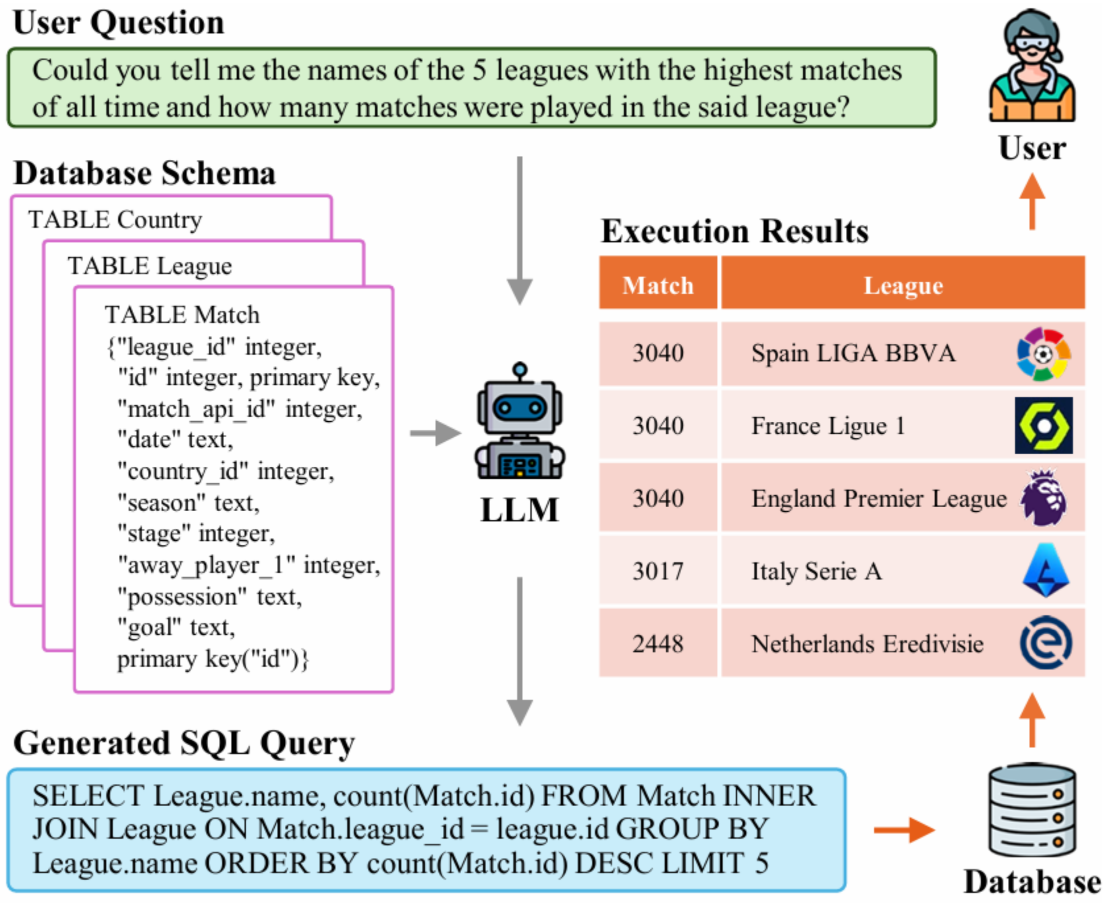


Li, et al., 2024, LLM for Data Management.

LLMs in Data Engineering Applications

Text-to-SQL by LLMs

- LLMs enable natural language querying without SQL skills.
- LLMs handle complex questions with accurate SQL translation.
- LLMs improve query efficiency and reduce manual errors.



Hong, et al., 2024, Next-Generation Database Interfaces: A Survey of LLM-based Text-to-SQL.

Challenges and Risks of LLMs

- **Hallucination**

  the generation of inaccurate, nonsensical, or detached text, posing potential risks and challenges for organizations utilizing these models.

- **Domain-specific knowledge & expertise**

  LLMs cannot perform well in many domain-specific fields like medicine, law, finance and more because of the lack of domain-specific knowledge and expertise.

- **Privacy**

  Various risks to data privacy and security exist at different stages of LLMs, which becomes particularly acute in light of incidents where sensitive internal data was exposed to LLMs.

- **Inconsistency**

  Sometimes they nail the answer to questions, other times they regurgitate random facts from their training data.

LLM challenges in Vertical Domains

❑ Domain of Law

OXFORD

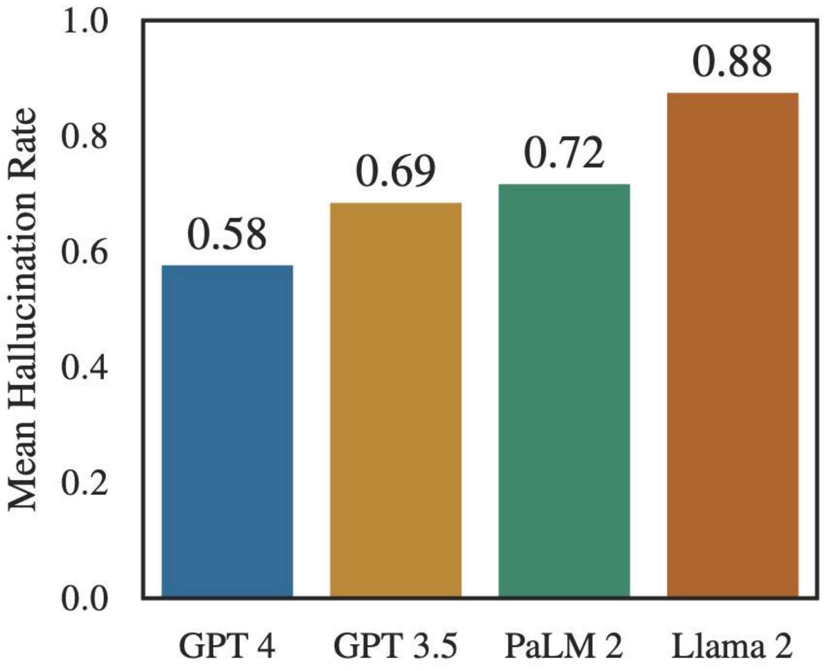
Journal of Legal Analysis, 2024, **16**, 64–93
https://doi.org/10.1093/jla/laae003
Advance access publication 26 June 2024
Article

Large Legal Fictions: Profiling Legal Hallucinations in Large Language Models

Matthew Dahl[*], Varun Magesh[†], Mirac Suzgun[‡], and Daniel E. Ho[§]

*In a new study by **Stanford RegLab** and **Institute for Human-Centered AI** researchers, it is demonstrated that legal hallucinations are pervasive and disturbing: **hallucination rates range from 69% to 88% in response to specific legal queries** for state-of-the-art language models.*

Hallucinations are common across all LLMs when they are asked a direct, verifiable question about a federal court case

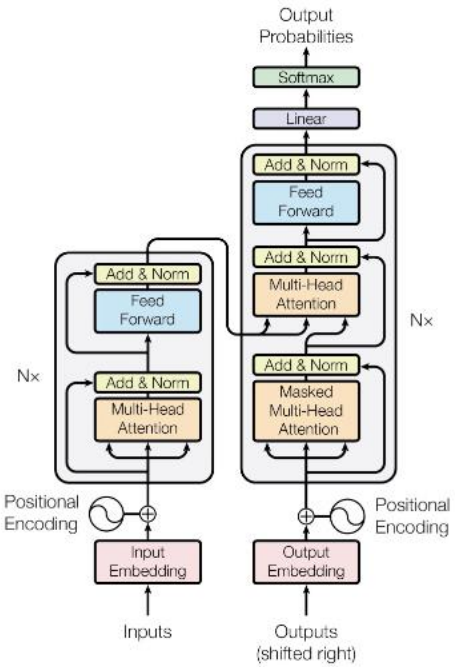


Dahl M, et al. 2024, Large legal fictions: Profiling legal hallucinations in large language models.

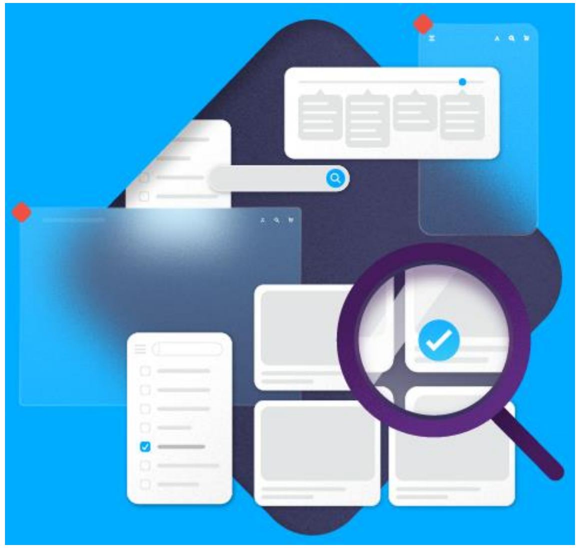
Why Large Language Models Work Well?

- Big Model + Big Training Data

Storing knowledge in the parametric model !



Storing knowledge in the non-parametric model?



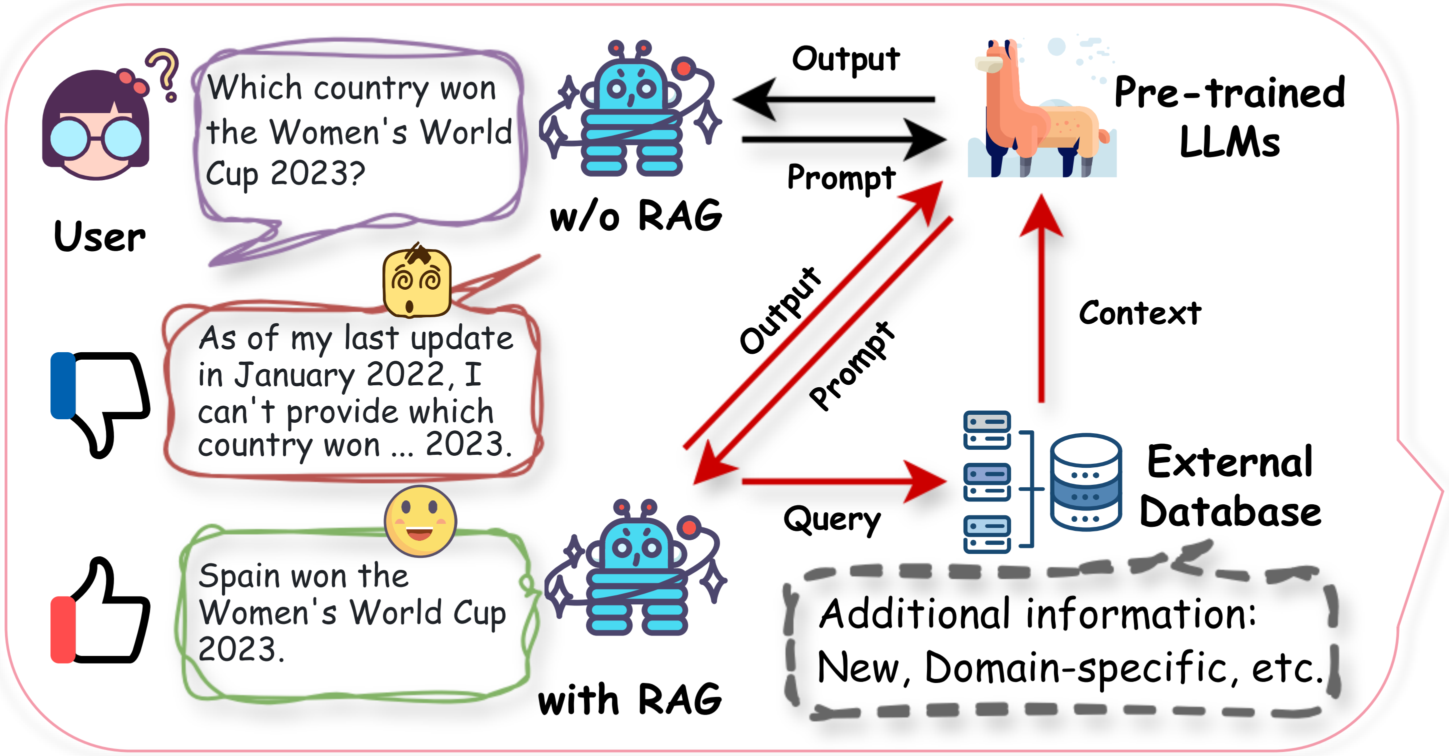
Information Retrieval (IR)

Retrieval-Augmented Large Language Models (RA-LLMs)

- LLMs **cannot memorize all** (particularly long-tail) knowledge in their parameters
- Lack of **domain-specific knowledge**, **updated information**, etc

Hallucination & Unable to answer → Re-training / Finetuning ?

Costly & Heavy Work



Retrieval-Augmented Generation (RAG) for LLM: **RA-LLMs**

Integrating Information Retrieval in Generation: RA-LLM

External Knowledge Base
- High quality
- Specialized
- Limited scale
- Easy-updated

Training Data Base
- Low quality
- General
- Massive
- Hard to update

+

=

Information / Knowledge retrieval

Content generation

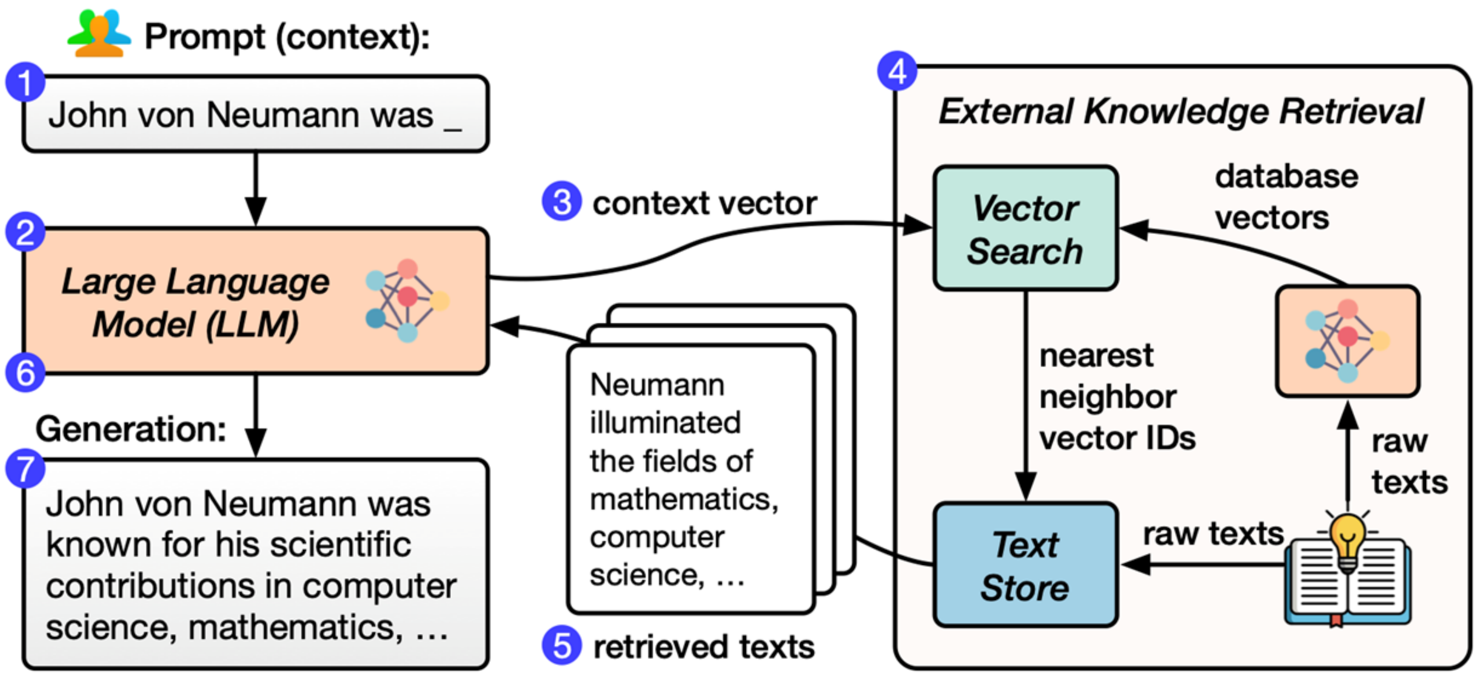
Close-book exam
*(Hard mode, have to **remember everything**)*

Retrieval augmented generation

Open-book exam
(Easy mode, allow to search in reference)

Data Engineering-Powered RAG

- **HNSW on hardware reduces memory bottlenecks in search.**
- **Disaggregated DB design scales retrieval and generation independently.**
- **FPGA-accelerated ANN boosts retrieval speed and throughput.**



Jiang, et al., 2024, Chameleon: a Heterogeneous and Disaggregated Accelerator System for Retrieval-Augmented Language Models.

Integrating Data Management in Generation

Document Management

- Embedding
- Fast indexing
- Optimizing Storage
- Query Optimization

Documents

+

Vector DB

Create outlines of documents, and ***quickly find knowledge****)*

+

Content generation

=

Efficient and accurate retrieval augmented generation

A Comprehensive Survey Paper

A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models

Wenqi Fan
wenqifan03@gmail.com
The Hong Kong Polytechnic University, HK SAR

Yujuan Ding*
dingyujuan385@gmail.com
The Hong Kong Polytechnic University, HK SAR

Liangbo Ning
BigLemon1123@gmail.com
The Hong Kong Polytechnic University, HK SAR

Shijie Wang
shijie.wang@connect.polyu.hk
The Hong Kong Polytechnic University, HK SAR

Hengyun Li
neilhengyun.li@polyu.edu.hk
The Hong Kong Polytechnic University, HK SAR

Dawei Yin
yindawei@acm.org
Baidu Inc, China

Tat-Seng Chua
dcscts@nus.edu.sg
National University of Singapore, Singapore

Qing Li
csqli@comp.polyu.edu.hk
The Hong Kong Polytechnic University, HK SAR

Accepted by KDD'2024
https://arxiv.org/pdf/2405.06211

Survey

Recruitment

- Our research group are actively recruiting self-motivated **postdoc, Ph.D. students, and research assistants**, etc. **visiting scholars, interns, and self-funded students** are also welcome. Send me an email if you are interested.

  - Research areas: machine learning (ML), data mining (DM), artificial intelligence (AI), deep learning (DNNs), large language models (LLMs), graph neural networks (GNNs), computer vision (CV), natural language processing (NLP), etc.

  - Position details: https://wenqifan03.github.io/openings.html

Tutorial Outline

- **Part 1: Introduction** of Retrieval Augmented Large Language Models (RA-LLMs) (Dr. Yujuan Ding)
- **Part 2: Architecture of RA-LLMs and Main Modules (Dr. Yujuan Ding)**
- **Part 3: Data Management** for RA-LLMs (Pangjing Wu)
- **Part 4: Learning** Approach of RA-LLMs (Liangbo Ning)
- **Part 5: Applications** of RA-LLMs (Shijie Wang)
- **Part 6: Challenges and Future Directions** of RA-LLMs (Liangbo Ning)

Website of this tutorial
Check out the slides and more information!

Website

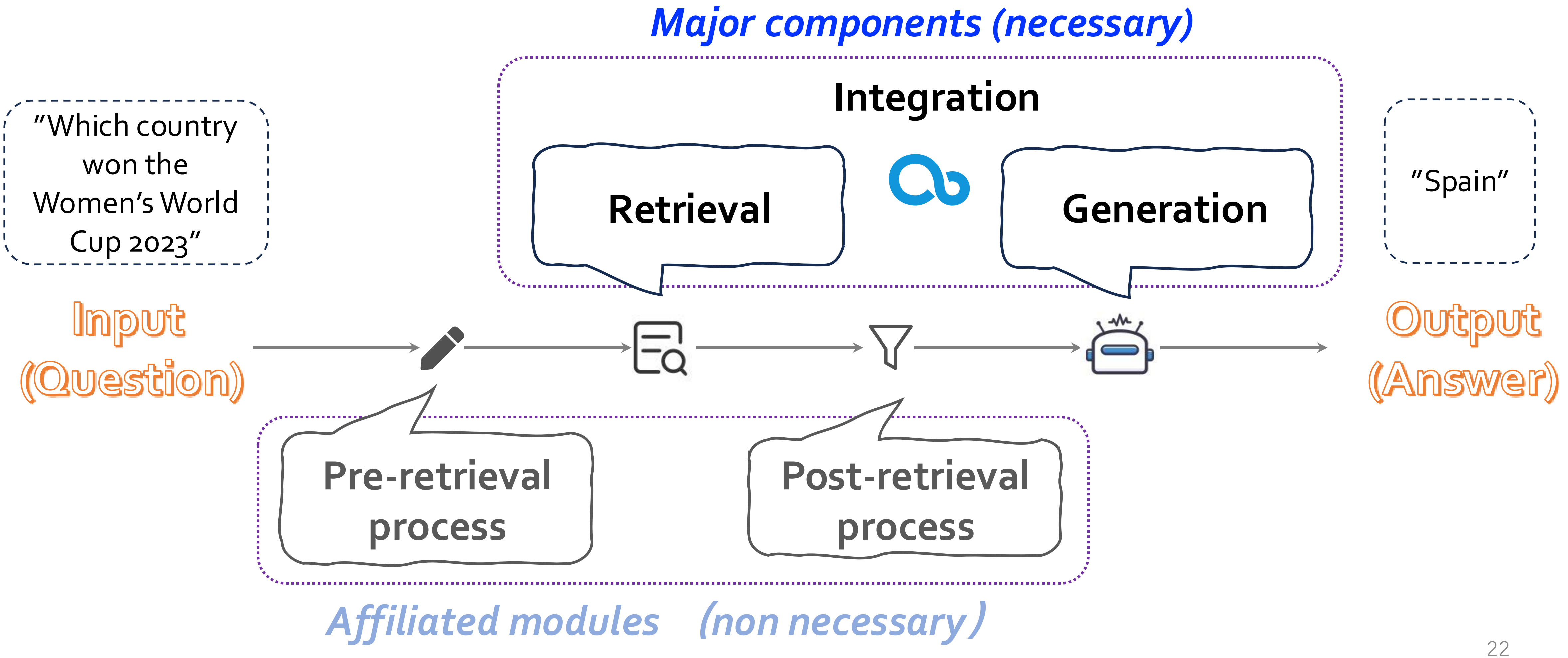
PART 2: Architecture of RA-LLMs and Main Modules

Presenter
Dr. Yujuan DING
HK PolyU

- RA-LLM architecture overview
- Retriever in RA-LLMs
- Retrieval results integration
- Pre/Post-retrieval techniques
- Special RA-LLM paradigms

RA-LLM Architecture: Standard Pipeline

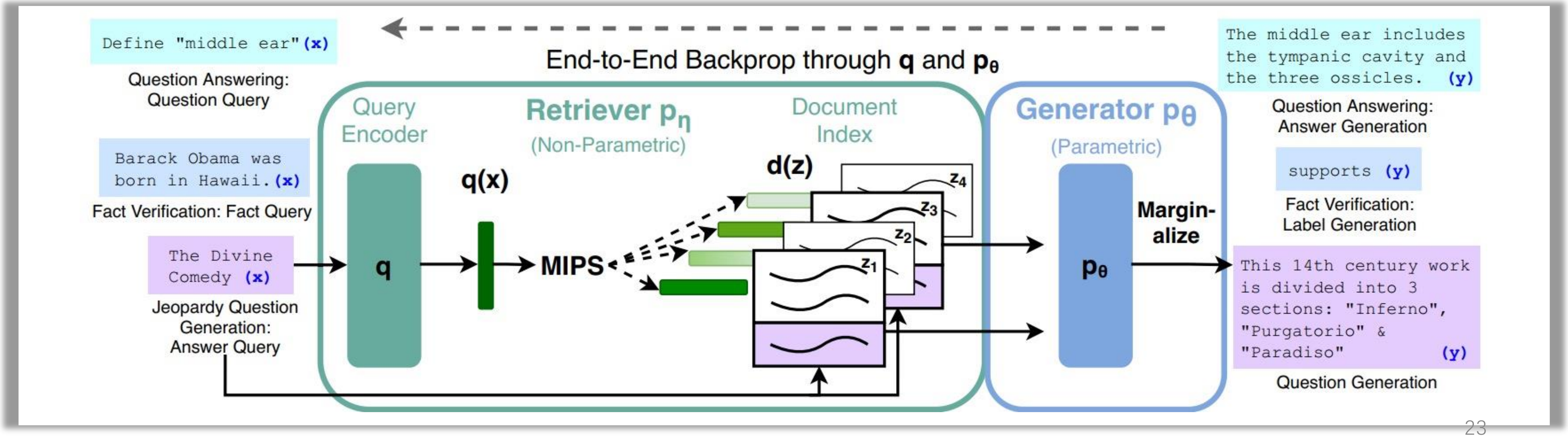
- Technical component illustration in a RA-LLM for the Q&A task



A Simple Retrieval-Augmented Generation Model

- **RAG**

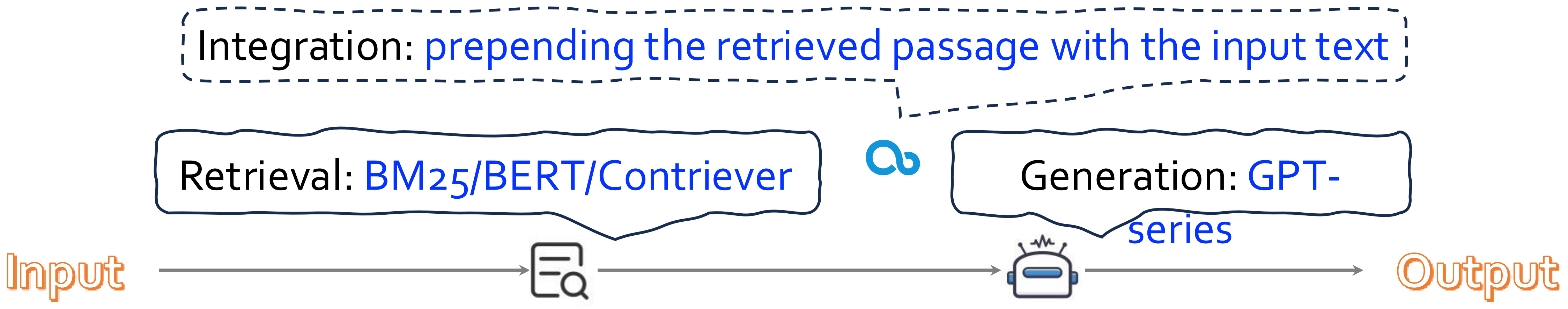
Integration: concatenating each retrieved passage with the question



Lewis et al. 2020. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks"

A Simple Retrieval-Augmented Generation Model

- **In-Context RALM**



Ram et al. 2023, In-Context Retrieval-Augmented Language Models

PART 2: Architecture of RA-LLMs and Main Modules

Slides

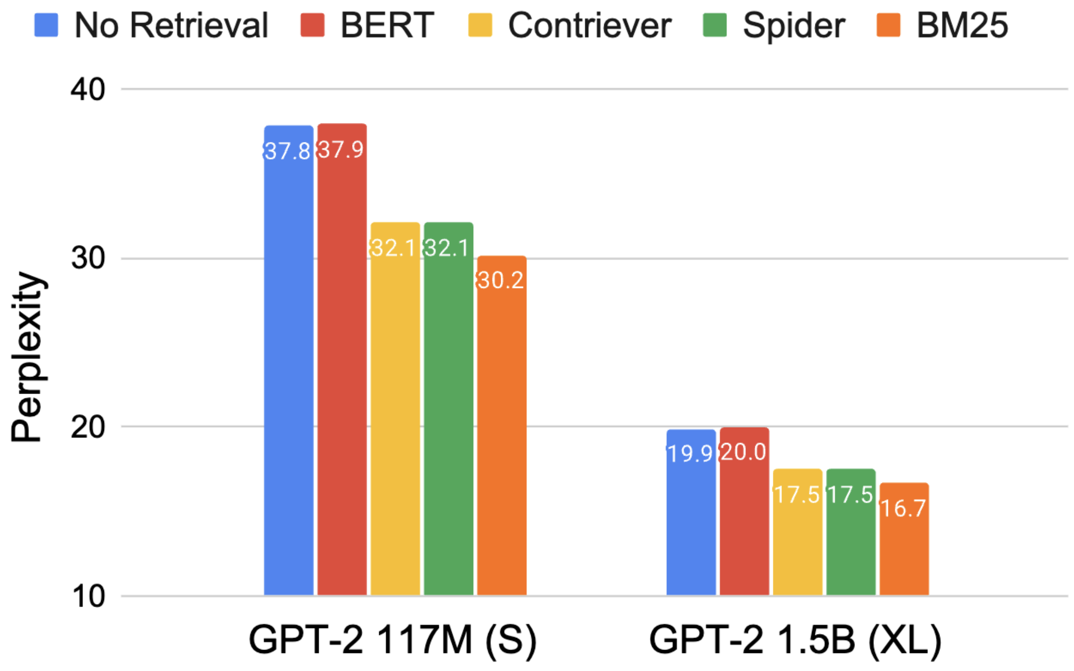
Website

Website of this tutorial

- **RA-LLM architecture overview**
- **Retriever in RA-LLMs**
- **Retrieval results integration**
- **Pre/Post-retrieval techniques**
- **Special RA-LLM paradigms**

RA-LLM Architecture: Retriever Types

- Different types of retriever deliver different generation performance



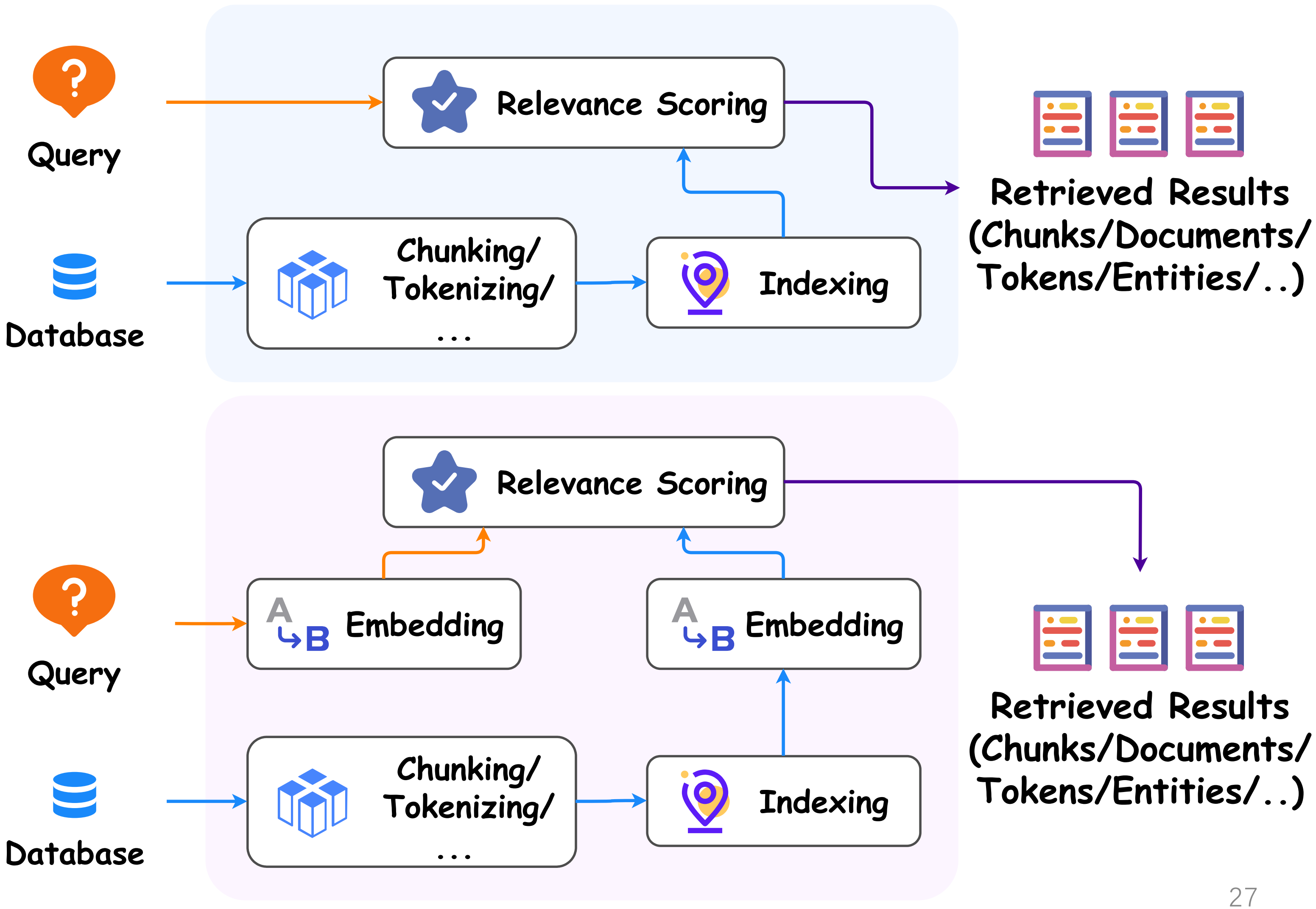
Relevance measurement	Retriever learning
Sparse	Task-specific pre-trained
Dense	General-purpose pre-trained

Ram et al. 2023, In-Context Retrieval-Augmented Language Models

Dense v.s. Sparse Retrievers

Sparse Retrievers (SR)

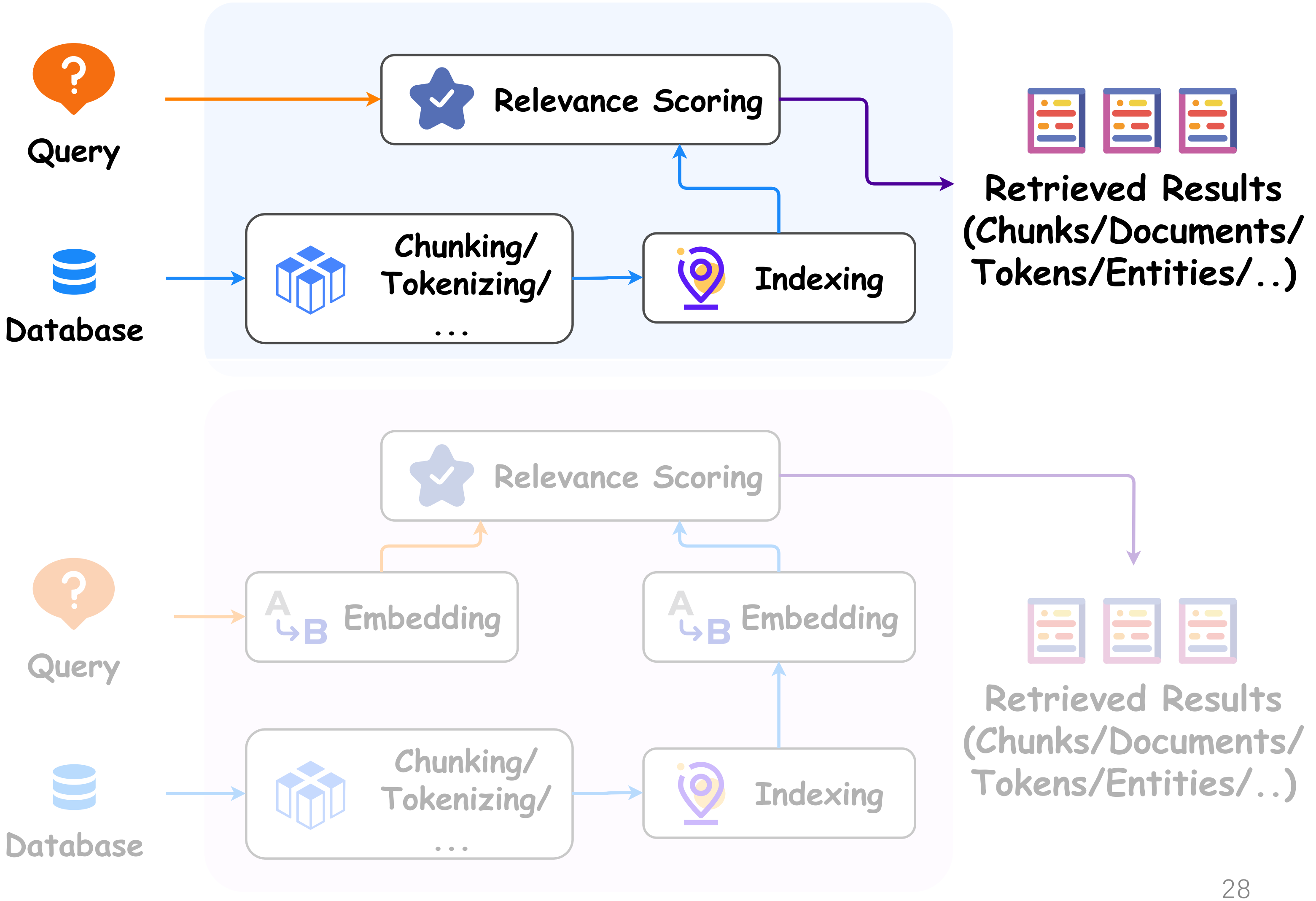
Dense Retrievers (DR)



Dense v.s. Sparse Retrievers

Sparse Retrievers (SR)

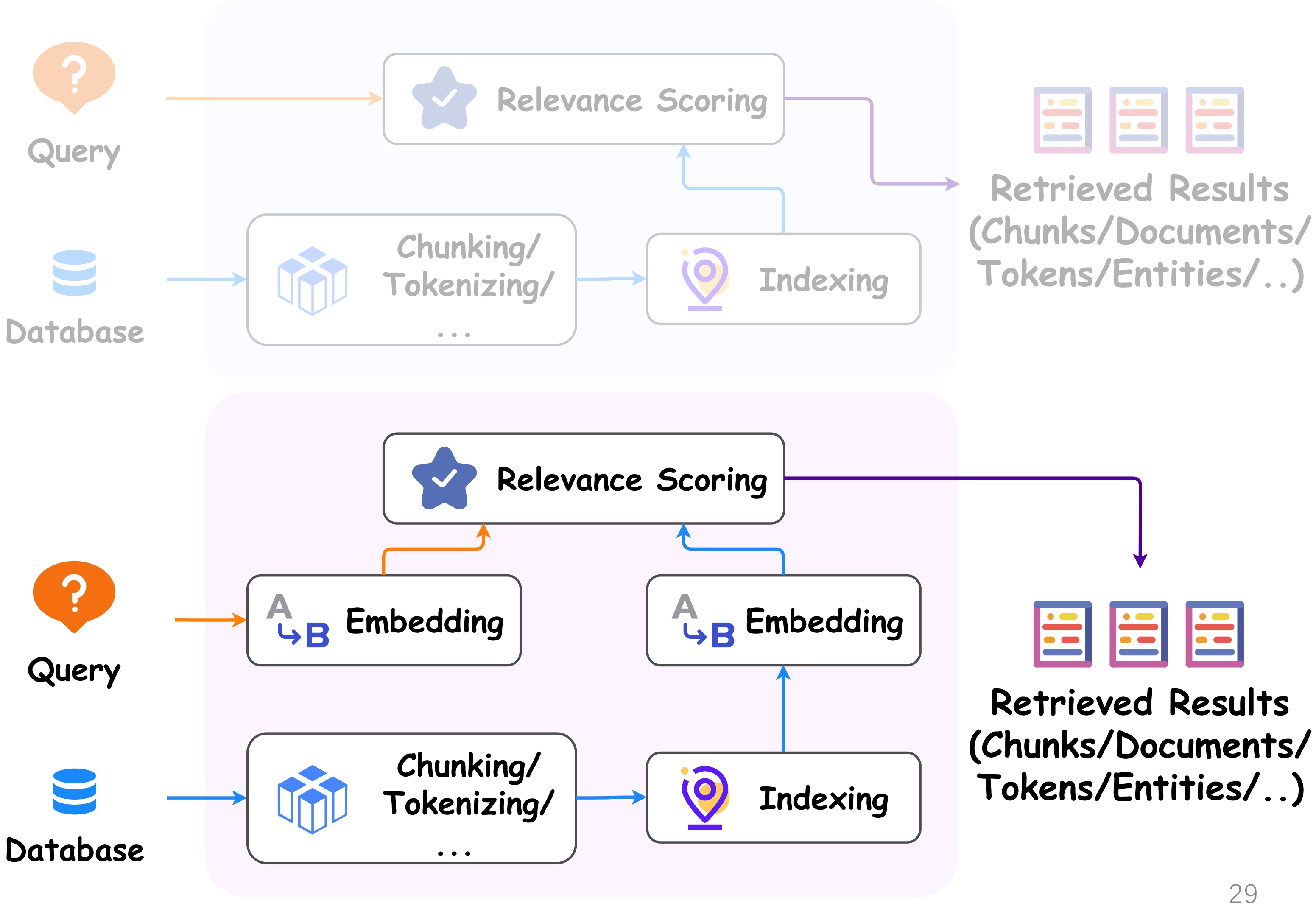
- Feasible to apply
- High efficiency
- Fine performance
- Example: TF-IDF, BM25



Dense v.s. Sparse Retrievers

Dense Retrievers (DR)

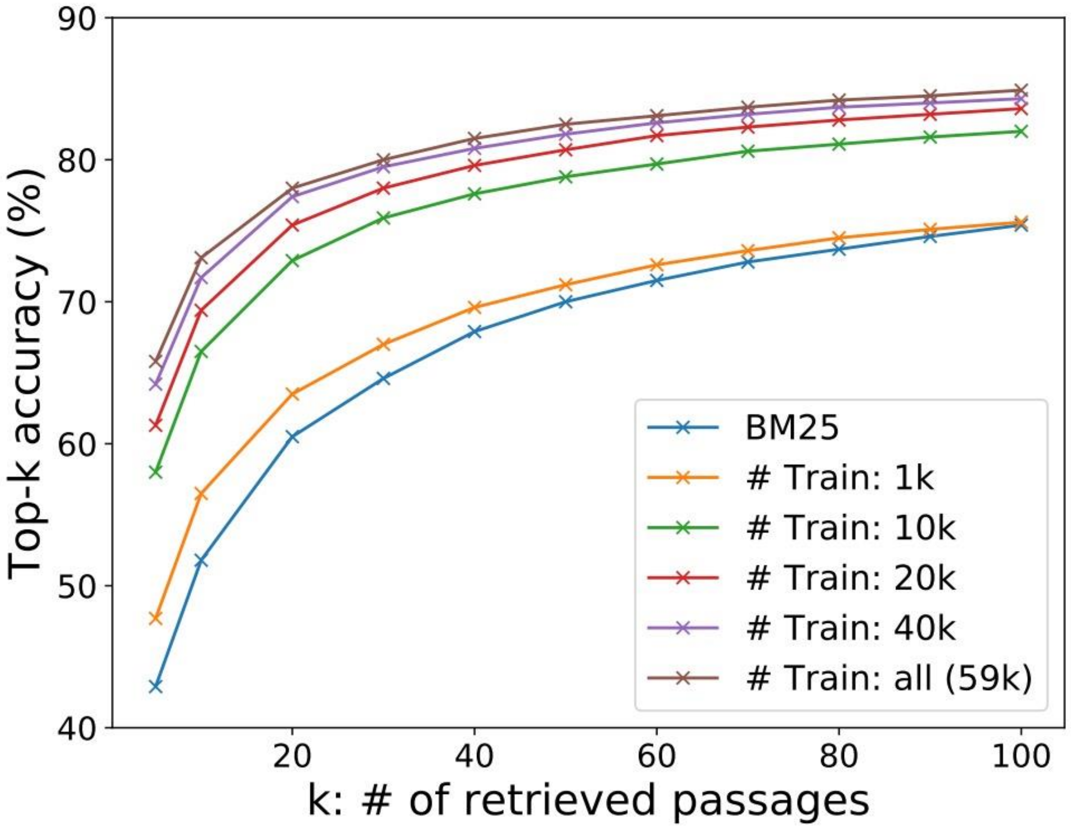
- Allowing fine-tuning
- Better adaptation
- Customizable for more retrieval goals
- Example: DPR, Contriever



Task-Specific Pre-trained Retriever (Supervised)

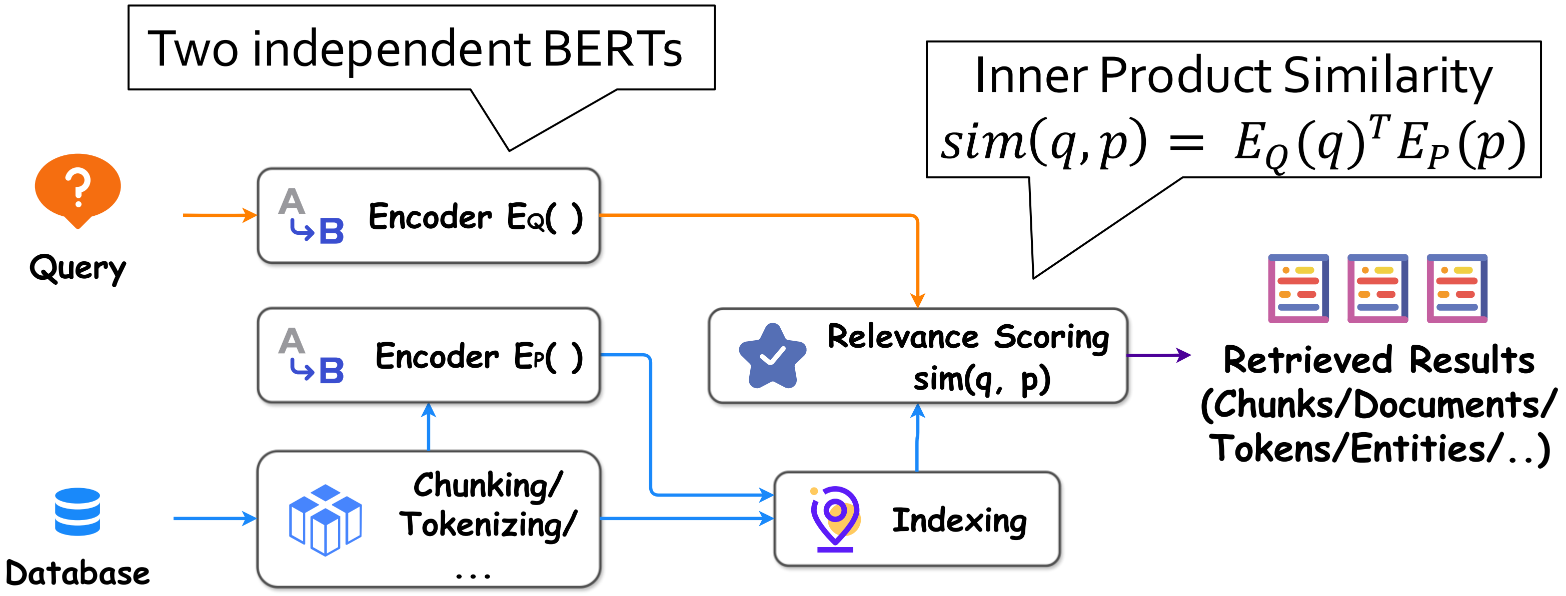
- **Dense Passage Retriever (DPR):** Pretrained for Question Answering (QA)

  - It is generally believed that learning a good dense vector representation needs a large number of labeled pairs of question and contexts.
  - Dense retrieval methods have never be shown to outperform TF-IDF/BM25 for opendomain QA before ORQA (Lee et al., 2019), which is unfortunately computationally intensive for pre-training and not specifically trained on Q&A data



Task-Specific Pre-trained Retriever (Supervised)

- **Dense Passage Retriever (DPR):** Pretrained for Question Answering (QA)



Karpukhin et al. 2020. “Dense Passage Retrieval for Open-Domain Question Answering”

Task-Specific Pre-trained Retriever (Supervised)

❑ Dense Passage Retriever (DPR): **Pretrained for Question Answering (QA)**

- Learning Objective

$$L(q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^-)$$
$$= -\log \frac{e^{\mathrm{sim}(q_i, p_i^+)}}{e^{\mathrm{sim}(q_i, p_i^+)} + \sum_{j=1}^{n} e^{\mathrm{sim}(q_i, p_{i,j}^-)}}$$

- Training data: Question-Passage Sets

$$\mathcal{D} = \{\langle q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^- \rangle\}_{i=1}^{m}$$

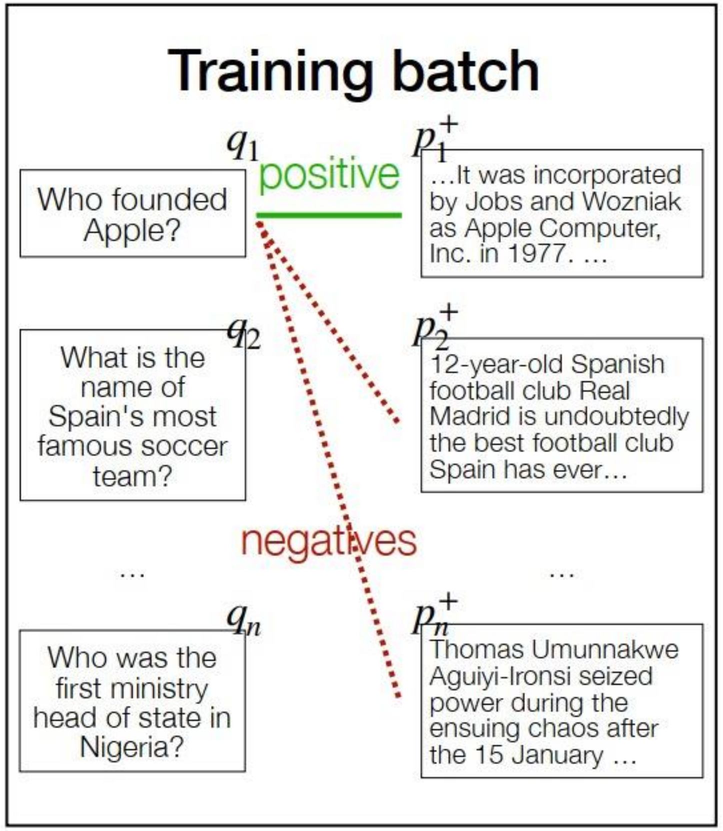
Question

Relevant passage

Irrelevant passages

Negative sample selection?

- Training with in-batch negatives



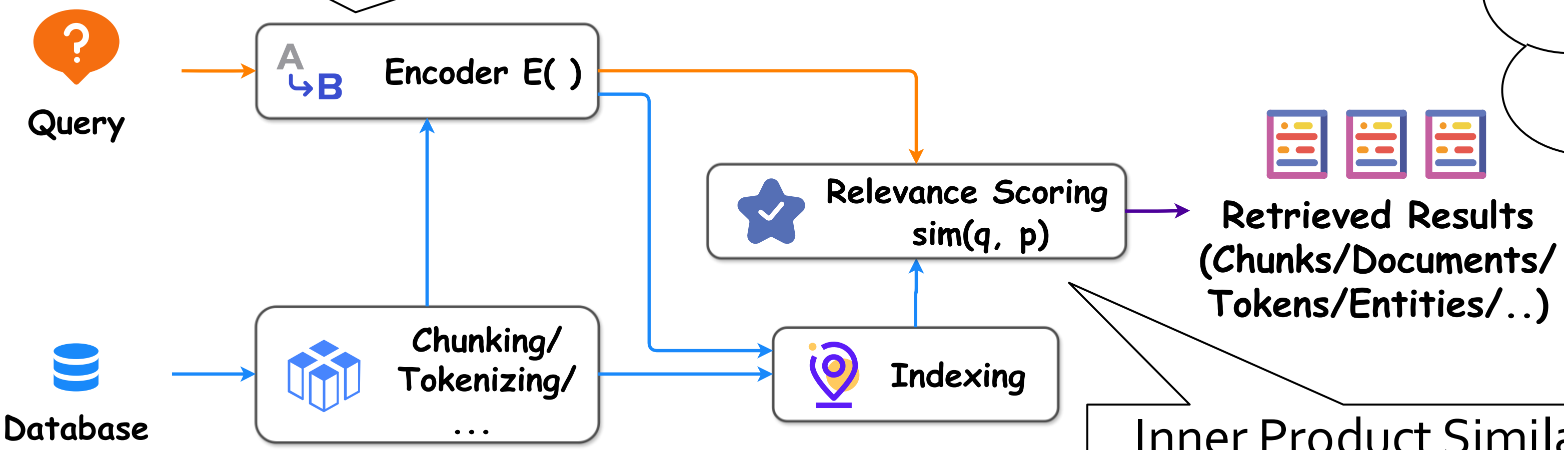
Karpukhin et al. 2020. “Dense Passage Retrieval for Open-Domain Question Answering”

General-Purpose Pre-trained Retriever (Unsupervised)

- **Contriever:** Pre-trained with unsupervised learning

Same BERT encoder for queries and documents

Might improve robustness in the context of zero-shot transfer or few-shot learning, while having no impact on other settings



Inner Product Similarity

$sim(q,p) = E(q)^T E(p)$

Contrastive learning with unaligned documents

$$\mathcal{L}(q, k_+) = -\frac{\exp(s(q,k_+)/\tau)}{\exp(s(q,k_+)/\tau) + \sum_{i=1}^{K} \exp(s(q,k_i)/\tau)}$$

Izacard et al. 2022. "Unsupervised Dense Information Retrieval with Contrastive Learning"

DPR & Contriever Performance on OpenQA Tasks

End-to-end QA (Exact Match) Accuracy

Training	Model	NQ	TriviaQA	WQ	TREC	SQuAD
Single	BM25+BERT (Lee et al., 2019)	26.5	47.1	17.7	21.3	33.2
Single	ORQA (Lee et al., 2019)	33.3	45.0	36.4	30.1	20.2
Single	HardEM (Min et al., 2019a)	28.1	50.9	-	-	-
Single	GraphRetriever (Min et al., 2019b)	34.5	56.0	36.4	-	-
Single	PathRetriever (Asai et al., 2020)	32.6	-	-	-	**56.5**
Single	REALM$_{Wiki}$ (Guu et al., 2020)	39.2	-	40.2	46.8	-
Single	REALM$_{News}$ (Guu et al., 2020)	40.4	-	40.7	42.9	-
	BM25	32.6	52.4	29.9	24.9	38.1
Single	DPR	**41.5**	56.8	34.6	25.9	29.8
	BM25+DPR	39.0	57.0	35.2	28.0	36.7
Multi	DPR	**41.5**	56.8	**42.4**	49.4	24.1
	BM25+DPR	38.8	**57.9**	41.1	**50.6**	35.8

Both widely applied in RAG and RA-LLMs

DPR in

RAG, FiD, RETRO, EPR, UDR, …

Contriever in

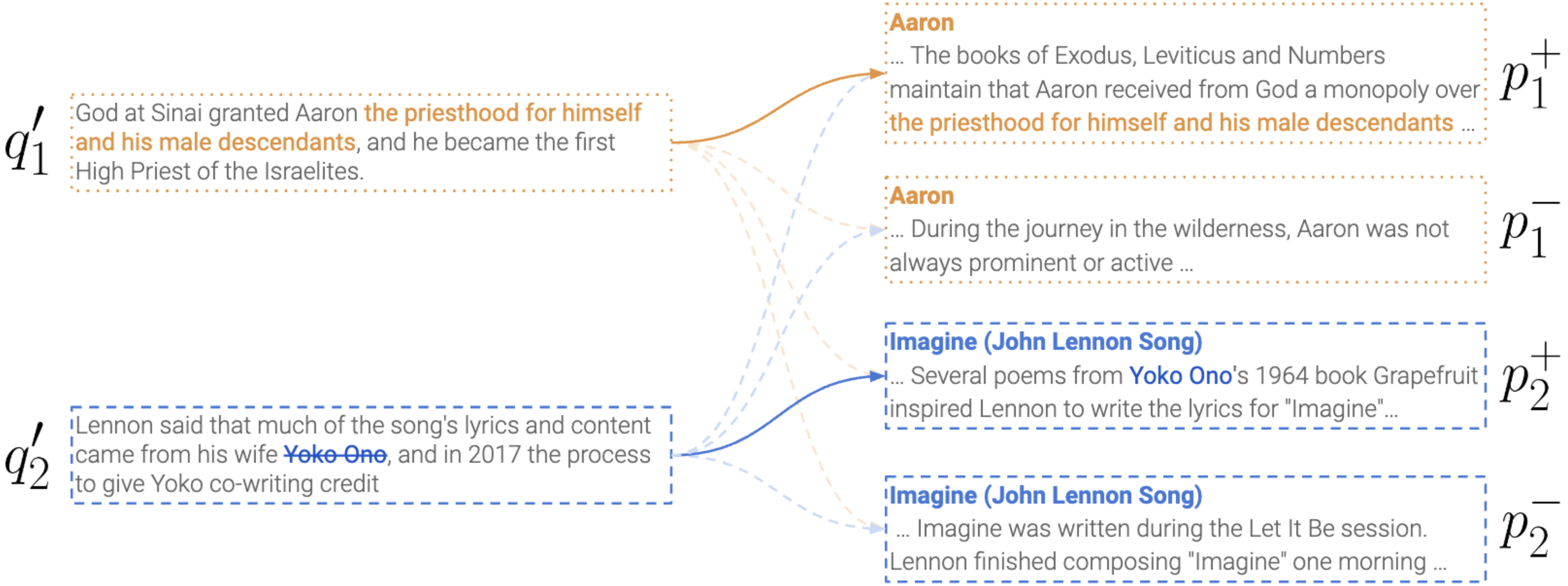
Self-RAG, Atlas, RAVEN, …

Both better than the sparse retriever!

	NaturalQuestions			TriviaQA		
	R@5	R@20	R@100	R@5	R@20	R@100
Inverse Cloze Task (Sachan et al., 2021)	32.3	50.9	66.8	40.2	57.5	73.6
Masked salient spans (Sachan et al., 2021)	41.7	59.8	74.9	53.3	68.2	79.4
BM25 (Ma et al., 2021)	-	62.9	78.3	-	**76.4**	**83.2**
Contriever	**47.8**	**67.8**	**82.1**	**59.4**	74.2	**83.2**
supervised model: DPR (Karpukhin et al., 2020)	-	78.4	85.4	-	79.4	85.0

Task-Specific Pre-trained Retriever (Unsupervised)

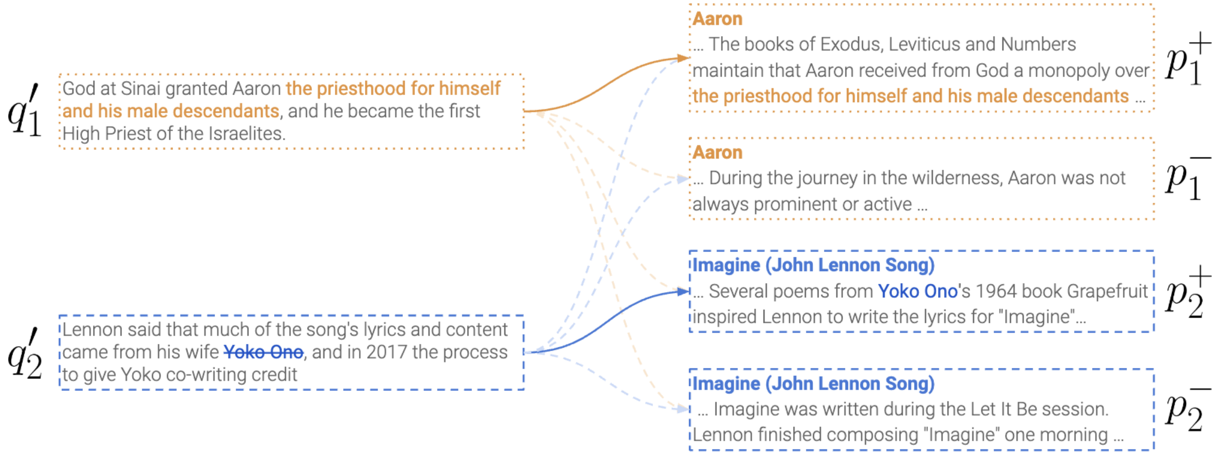
- **Spider** (Span-based unsupervised dense retriever)
  - **Recurring Span Retrieval**: It is based on the notion of recurring spans within a document: given two paragraphs with the same recurring span, we construct a query from one of the paragraphs, while the other is taken as the target for retrieval



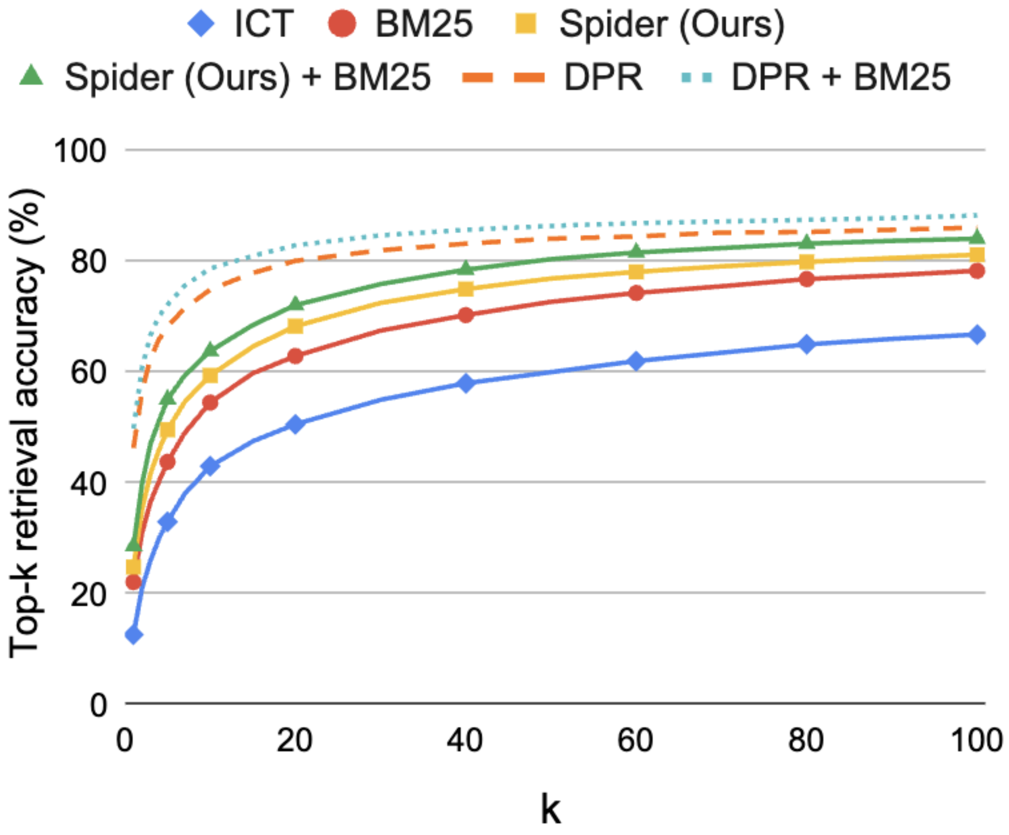
Ram et al., 2022, Learning to Retrieve Passages without Supervision

Task-Specific Pre-trained Retriever (Unsupervised)

- Learning and results of Spider

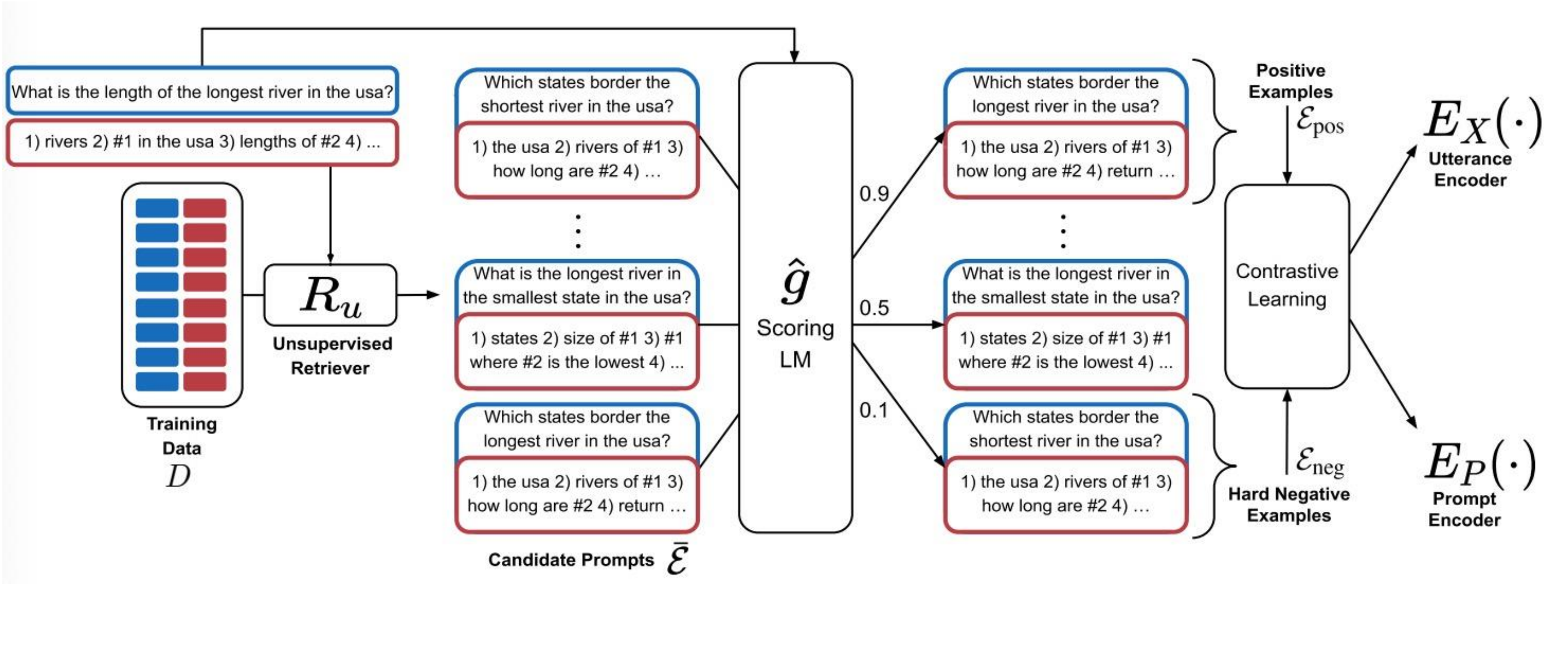


$$-\log \frac{\exp\left(s(q_i', p_i^+)\right)}{\sum_{j=1}^{m}\left(\exp\left(s(q_i', p_j^+)\right) + \exp\left(s(q_i', p_j^-)\right)\right)}$$

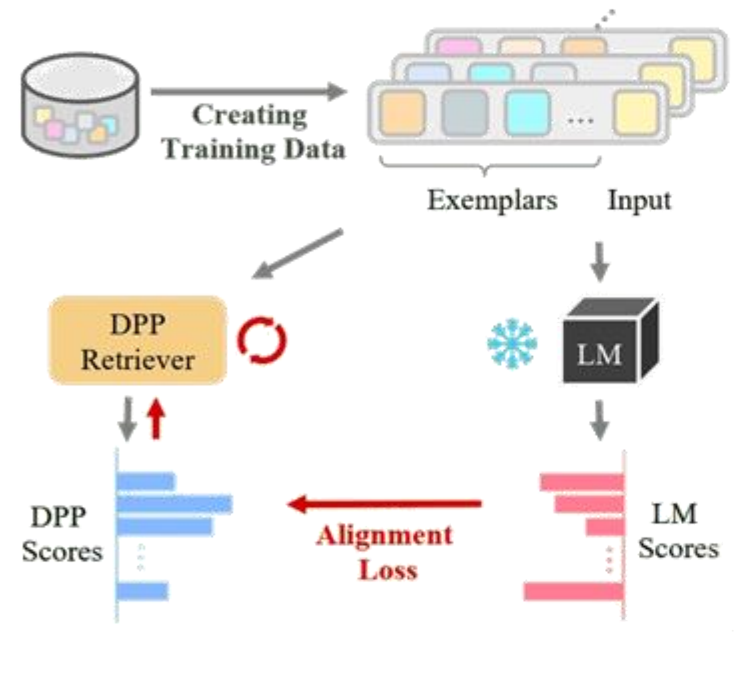


Retrievers for In-Context Learning of LLMs

- Prompt Retriever



- Examplar Retriever (CEIL)



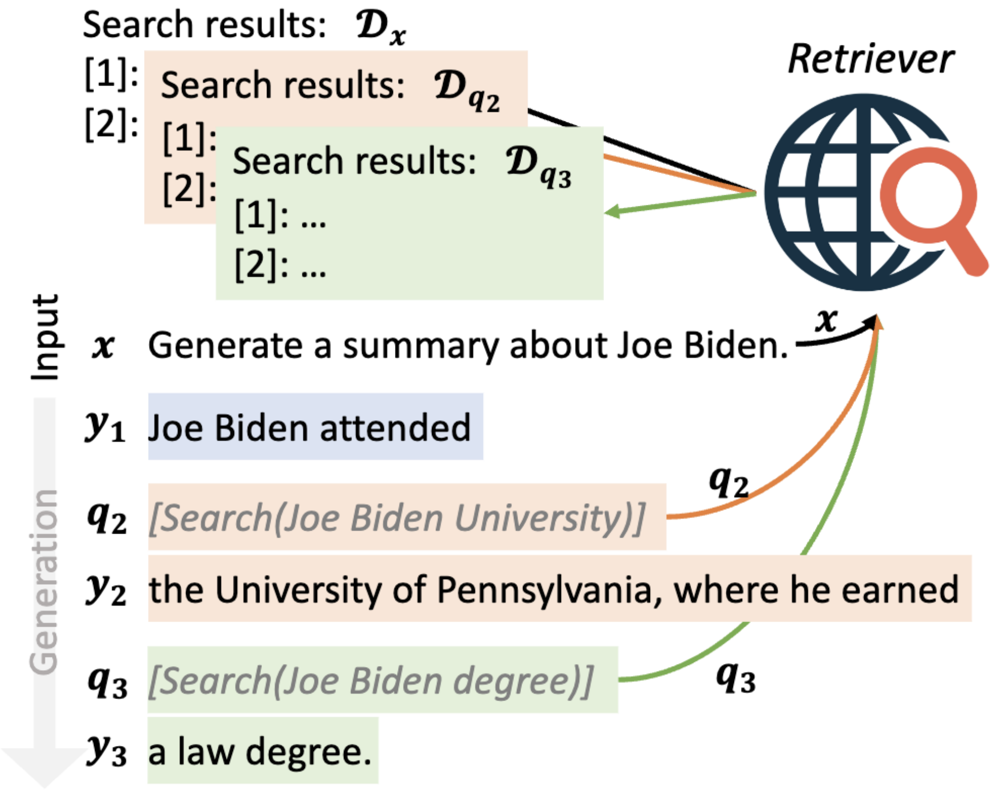
Rubin et al. 2023. "Learning To Retrieve Prompts for In-Context Learning"
Ye et al. 2023. "Compositional Exemplars for In-context Learning"

Search Engine as Retrievers

❑ Traditional retrieval methods

Google Bing

- May be difficult to update to real-time web documents
- May be a limit to the number of documents storable in the pre-defined database
- Will not take advantage of the high quality ranking that has been finely tuned in Internet Search engines over decades of use

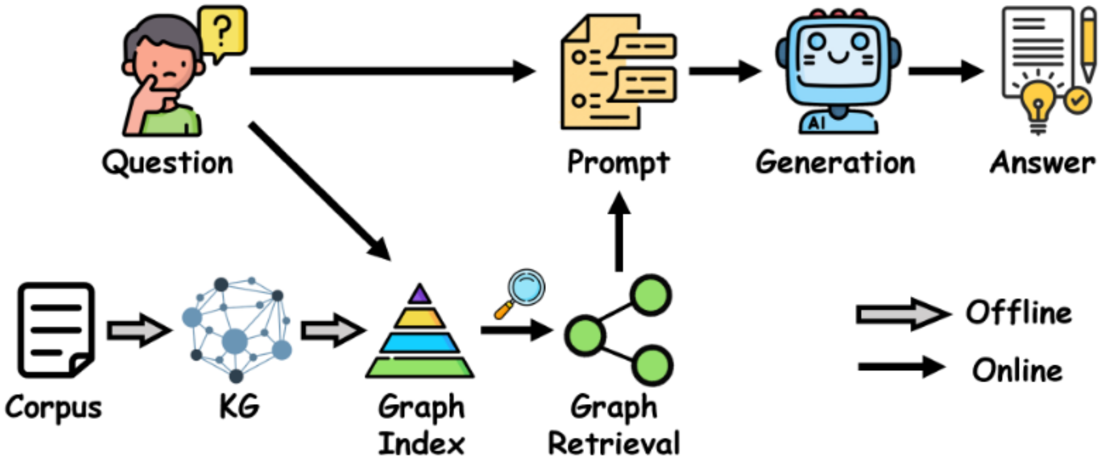


Komeili et al., 2021, Internet-Augmented Dialogue Generation
Jiang et al. 2023, Active Retrieval Augmented Generation

Knowledge Graph (KG) for Retrieval

❑ Text-based retrieval

- ☒ Measuring only **semantic similarity** between two texts
- ☒ Overlooking knowledge-level association
- ☒ Cannot capture structured relationship



e.g., Global Financial Crisis and The 2008 Recession

Harry Potter and Fantastic Beasts which are both written by J.K. Rowling

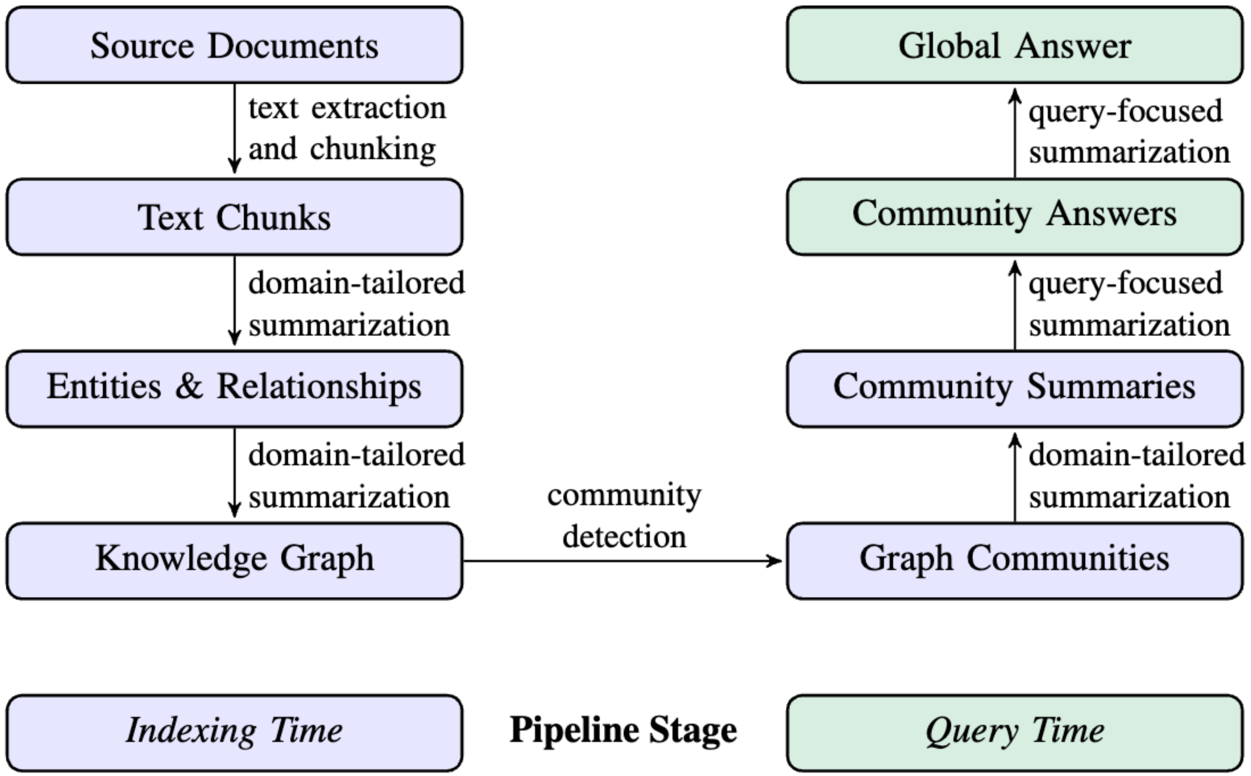
Ma S, et al., 2025, Think-on-Graph 2.0: Deep and Faithful Large Language Model Reasoning with Knowledge-guided Retrieval Augmented Generation

Knowledge Graph (KG) for Retrieval

- **GraphRAG**: aggregates nodes into communities, and produces a community report to capture global information

☒ RAG fails on global questions directed at an entire text corpus

e.g., What are the main themes in the dataset?

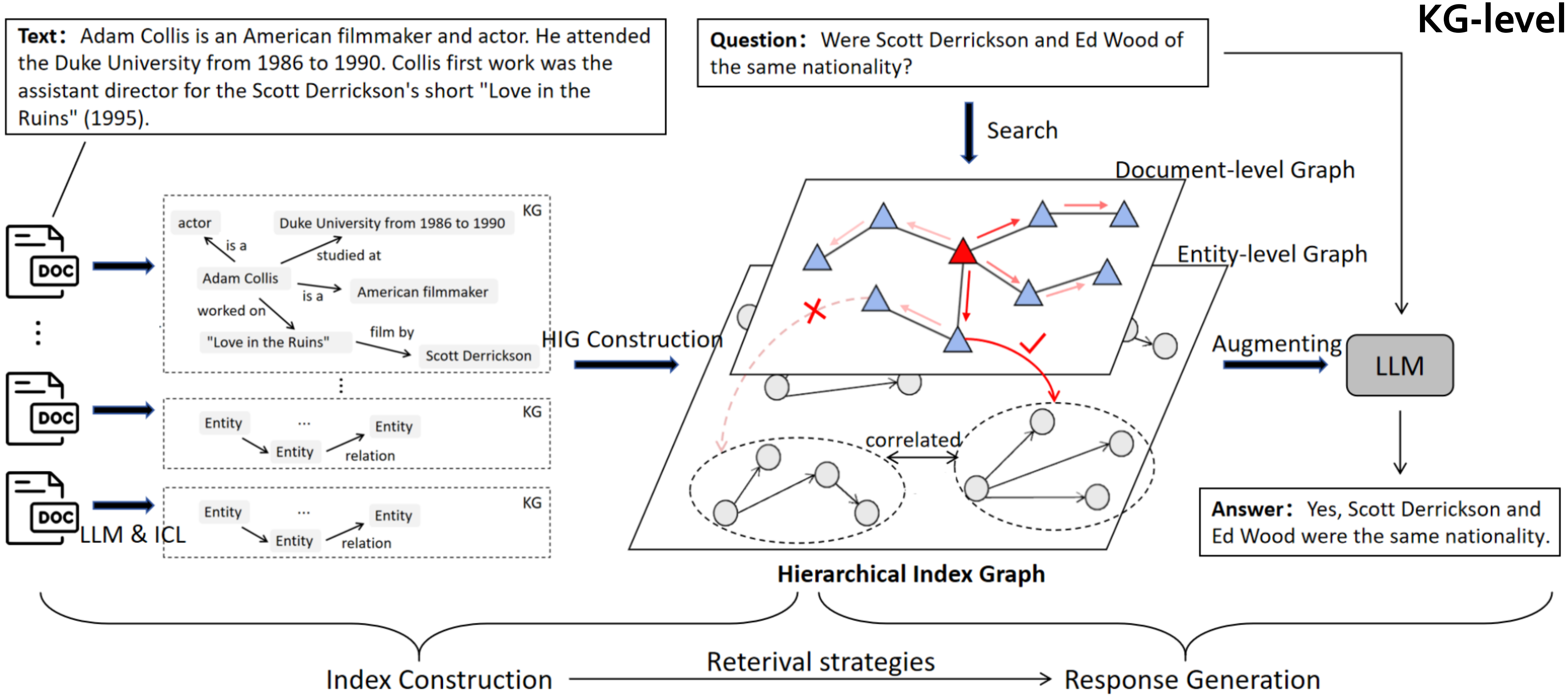


Edge D, et al., 2024 From local to global: A graph rag approach to query-focused summarization.

Knowledge Graph (KG) for Retrieval

❑ KG-Retriever with Hierarchical Index Graph (HIG)

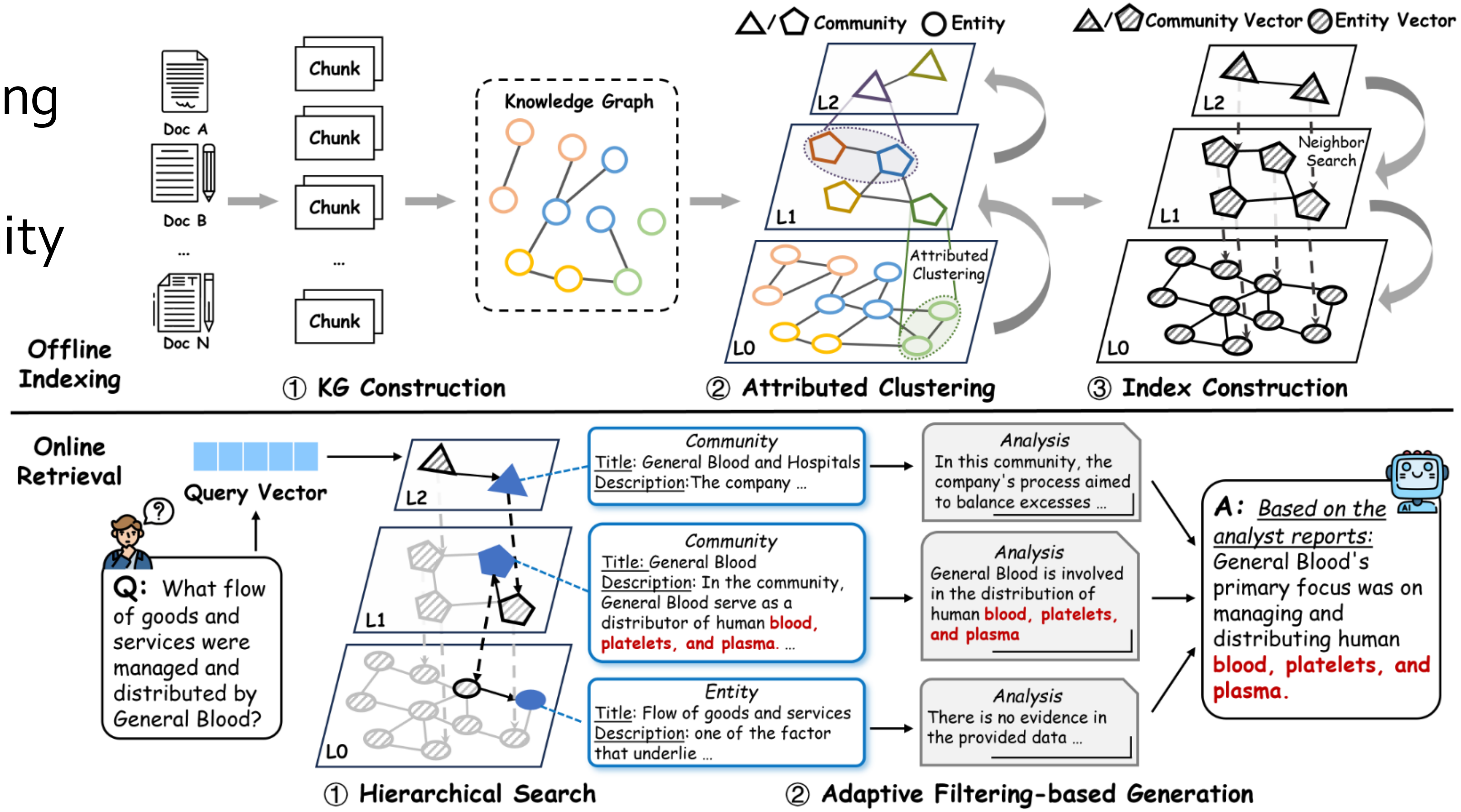
Document-level Retrieval
+
KG-level Retrieval



Chen W, et al., 2024, Kg-retriever: Efficient knowledge indexing for retrieval-augmented large language models

Knowledge Graph (KG) for Retrieval

- **ArchRAG** (Attributed Community-based Hierarchical RAG): augmenting the question using attributed communities
  - Hierarchical indexing
  - Attribute community
  - Efficient HNSW



Wang S, et al., 2025 ArchRAG: Attributed Community-based Hierarchical Retrieval-Augmented Generation.

PART 2: Architecture of RA-LLMs and Main Modules

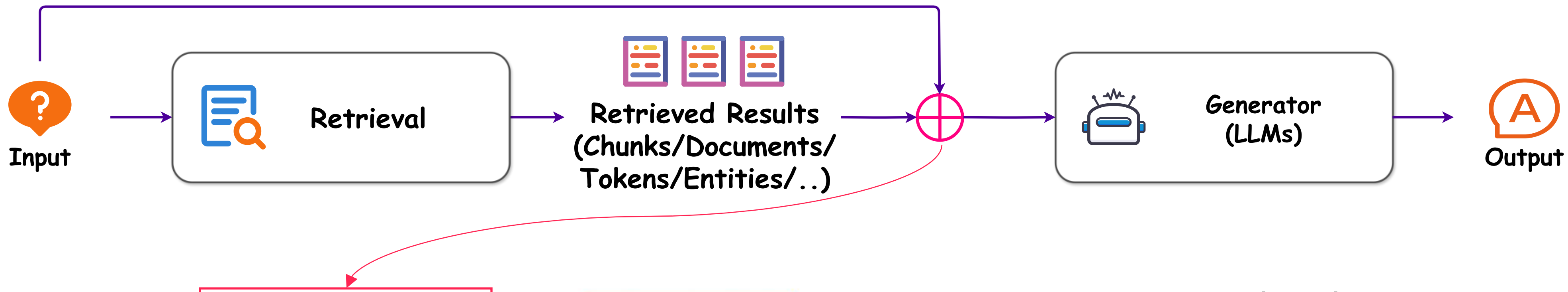
Slides

Website of this tutorial

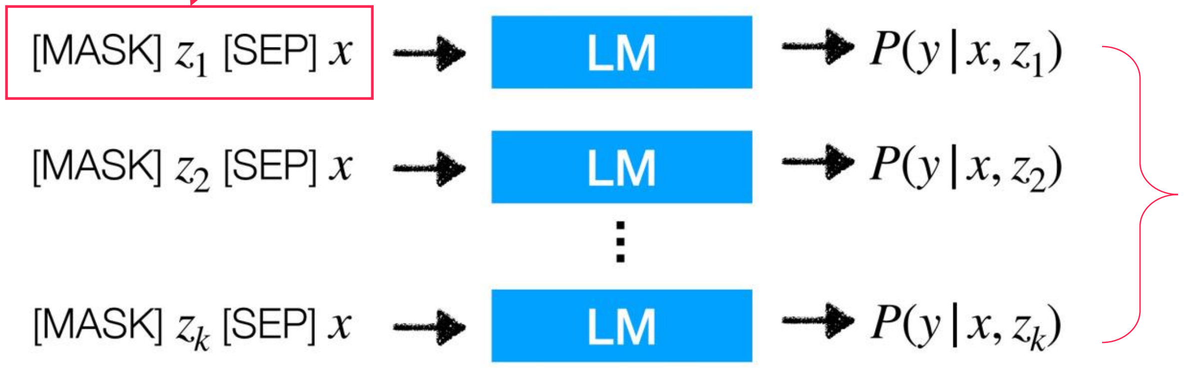
- RA-LLM architecture overview
- Retriever in RA-LLMs
- Retrieval results integration
- Pre/Post-retrieval techniques
- Special RA-LLM paradigms

Retrieved Results Integration: Input-layer Integration

❑ REALM



Integrating the retrieved passage ***z*** and ***x*** the original input



Weighted aggregating the prediction results based on all retrieved passages

$$\sum_{z\in\mathcal{D}} P(z|x)P(y|x,z)$$

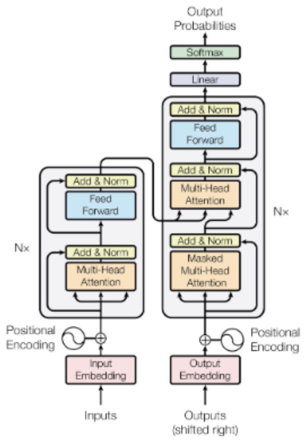
Guu et al. 2020. "REALM: Retrieval-Augmented Language Model Pre-Training"

Retrieval-Augmented Generator

Typical encoder: $p(y|x)$

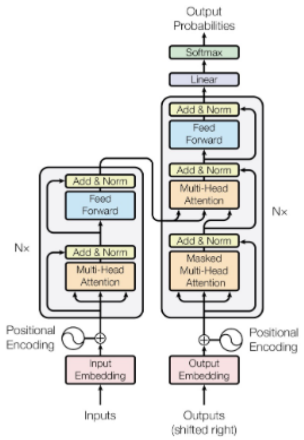
Knowledge-augmented encoder:$p(y|x, z)$

y= pounds



x: we paid 20 __ at the Buckingham Palace gift shop

y= pounds

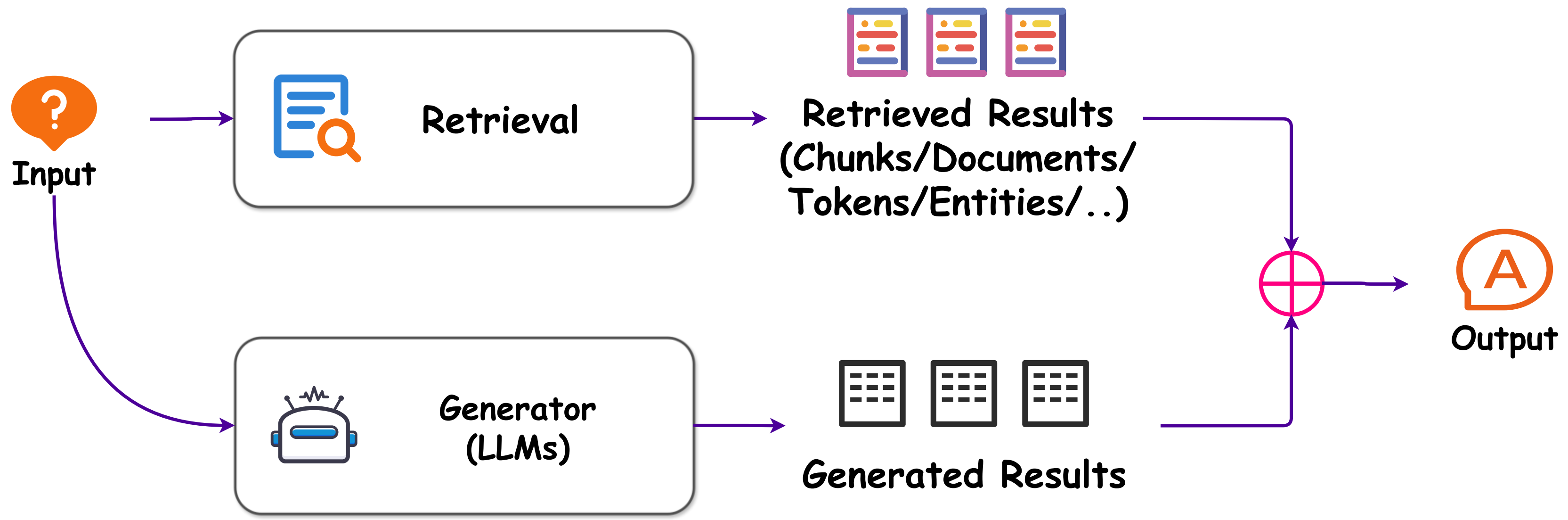


x: we paid 20 __ at the Buckingham Palace gift shop

z: Buckingham Palace is home to the British monarchy

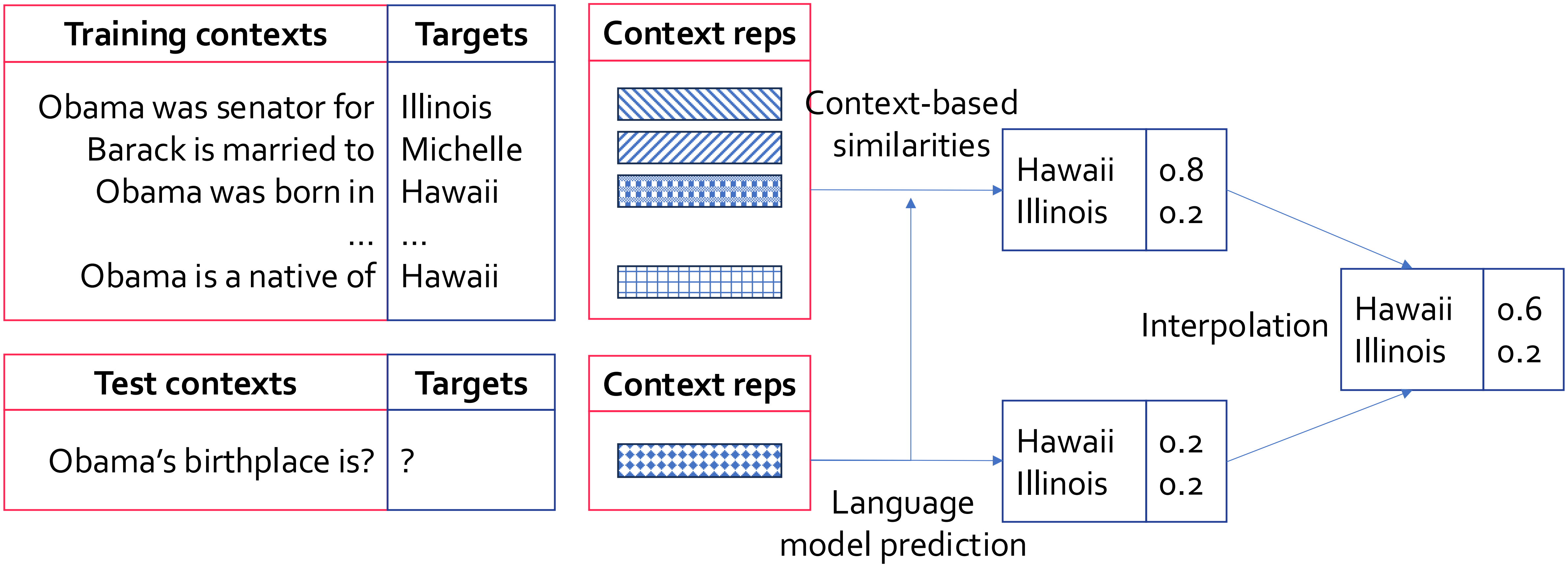
explicit knowledge

Retrieved Results Integration: Output-layer integration



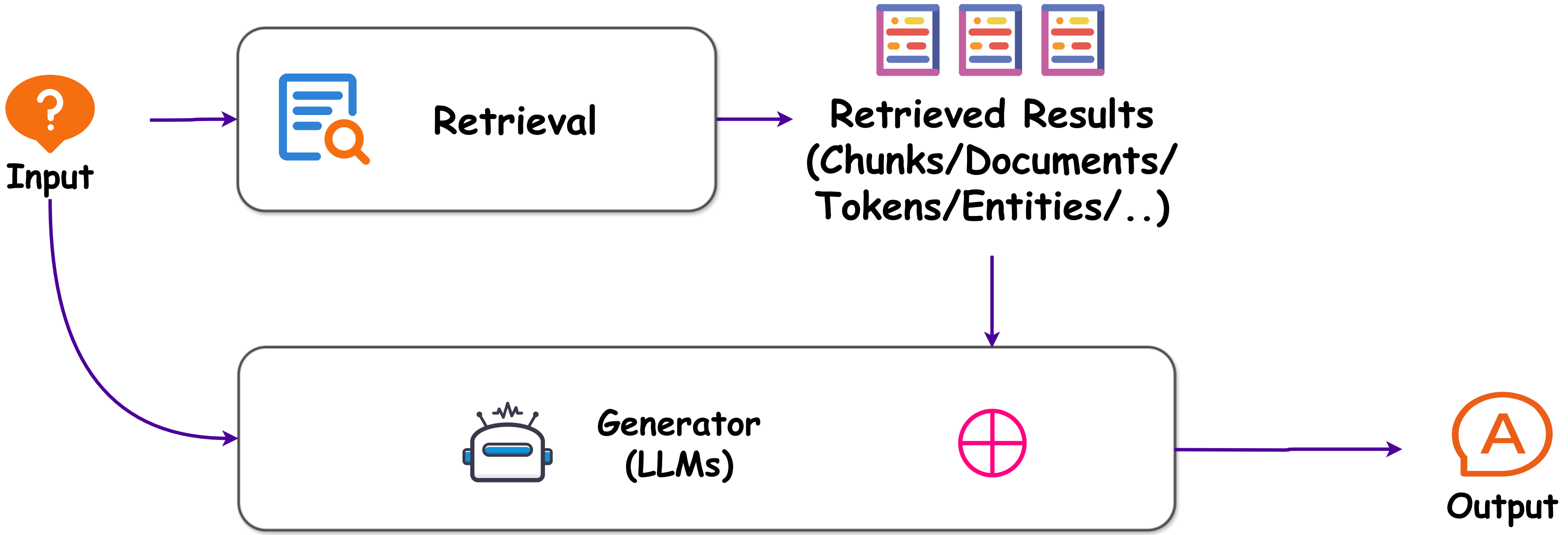
RA-LLM Architecture: Output-layer Integration

- **kNN-LM:** Combining retrieved probabilities and predicted ones in generation



Khandelwal el al. 2019. "Generalization through Memorization: Nearest Neighbor Language Models"

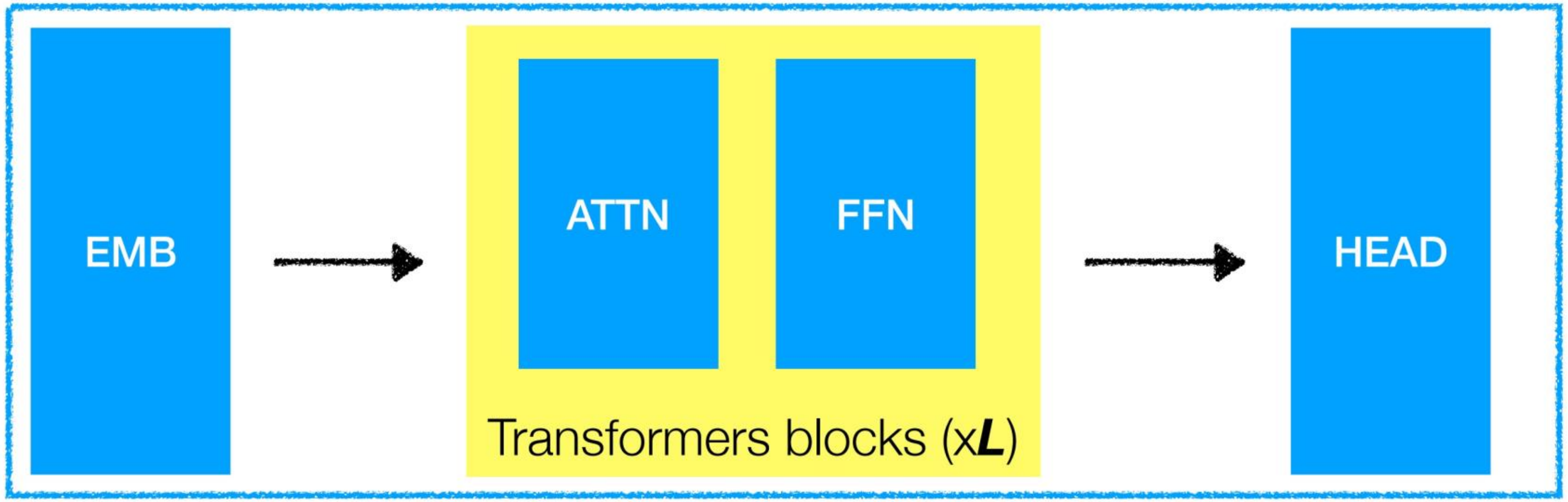
Retrieved Results Integration: Intermediate-layer Integration



Borgeaud et al. 2022. Improving Language Models by Retrieving from Trillions of Tokens

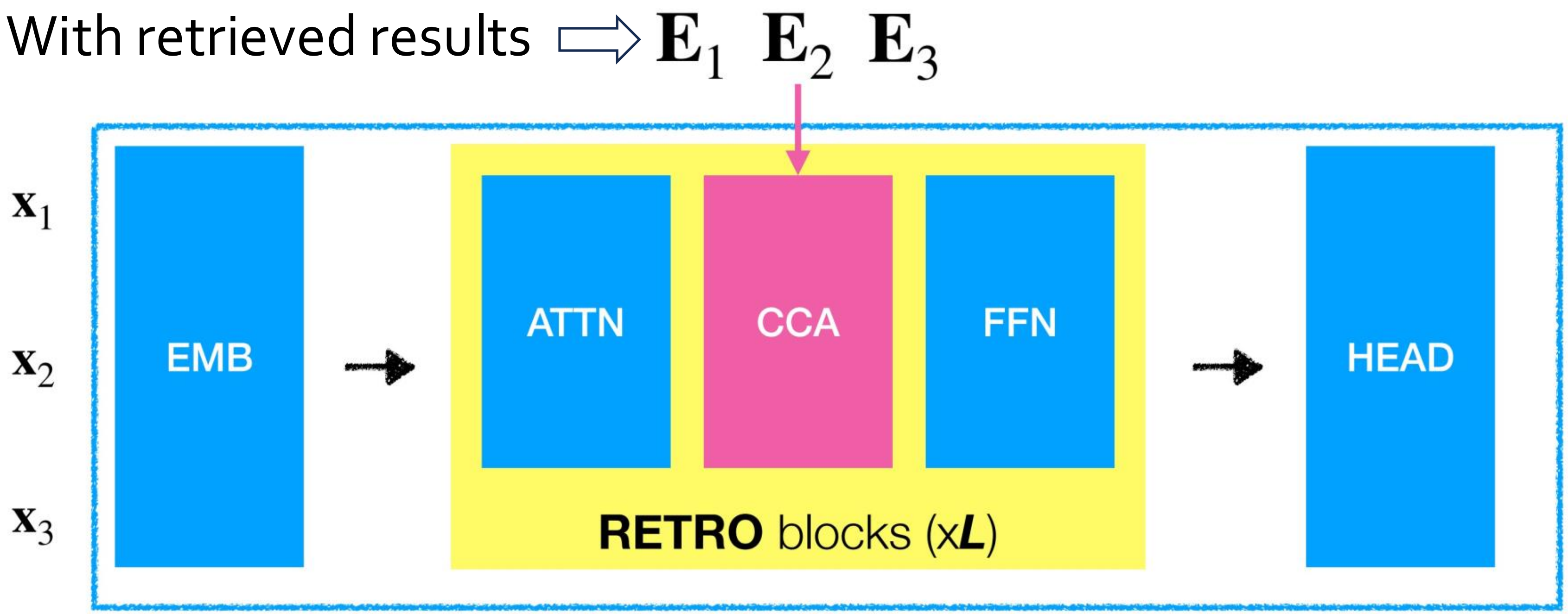
Retrieved Results Integration: Intermediate-layer Integration

Regular Decoder



Retrieved Results Integration: Intermediate-layer Integration

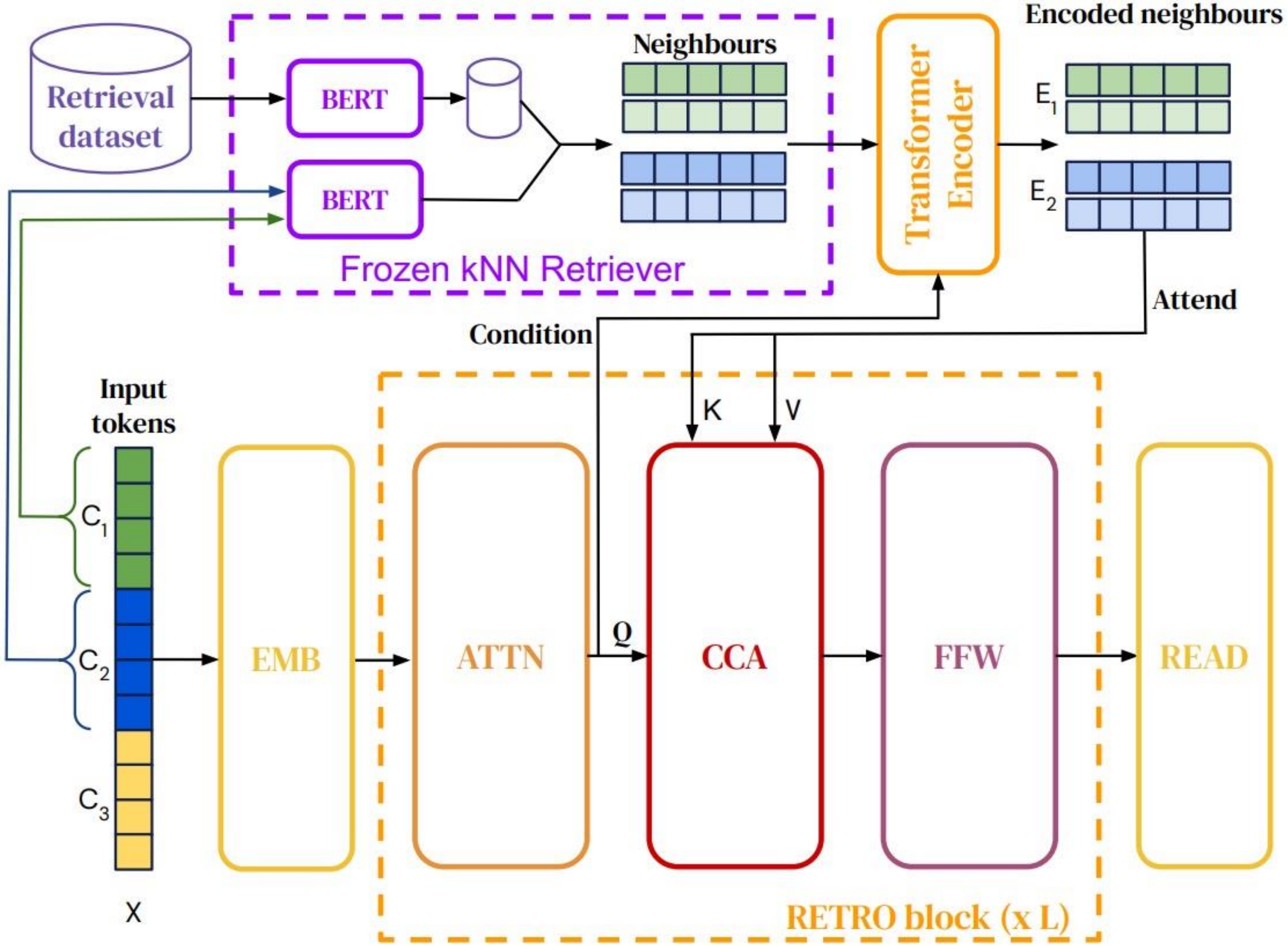
Decoder to incorporate retrieved results
(RETRO)



Chunked Cross Attention (CCA)

Borgeaud et al. 2022. Improving Language Models by Retrieving from Trillions of Tokens

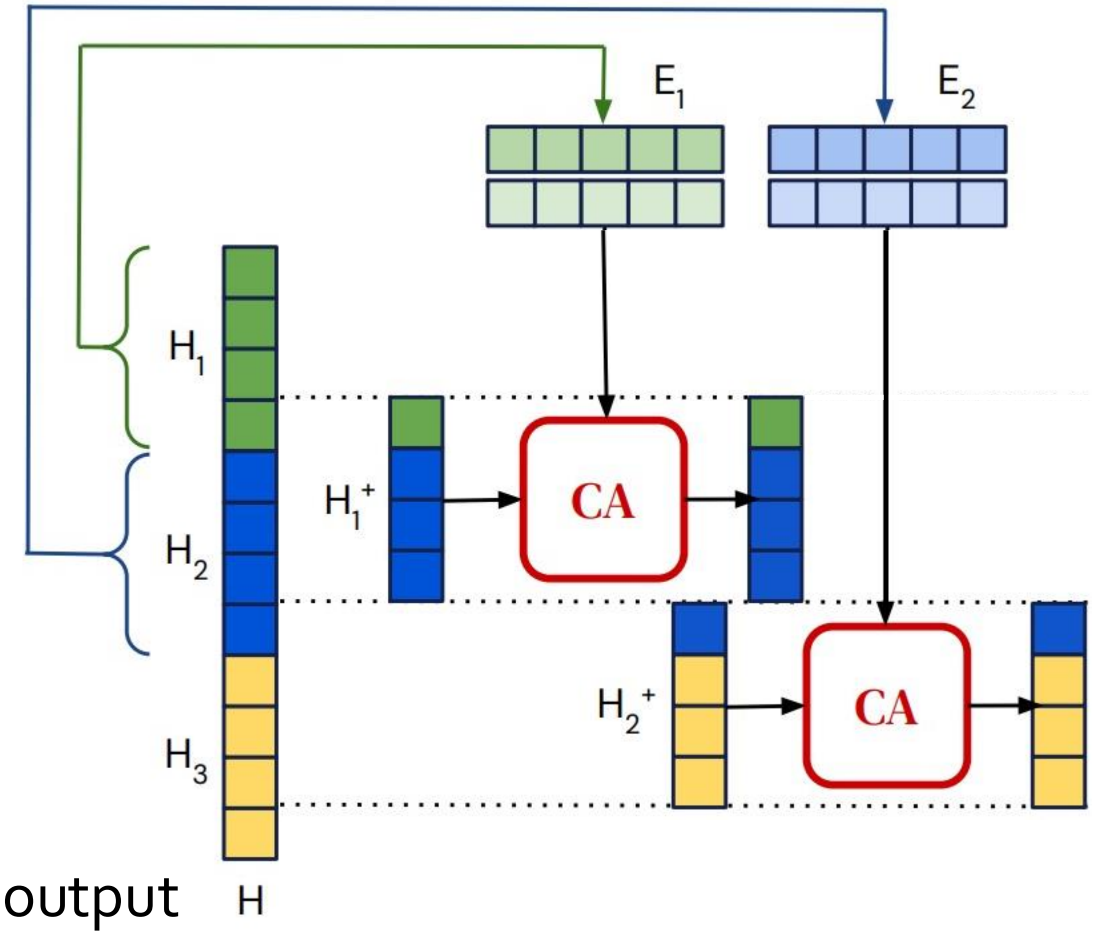
Retrieved Results Integration: Intermediate-layer Integration



Borgeaud et al. 2022. Improving Language Models by Retrieving from Trillions of Tokens

Retrieved Results Integration: Intermediate-layer Integration

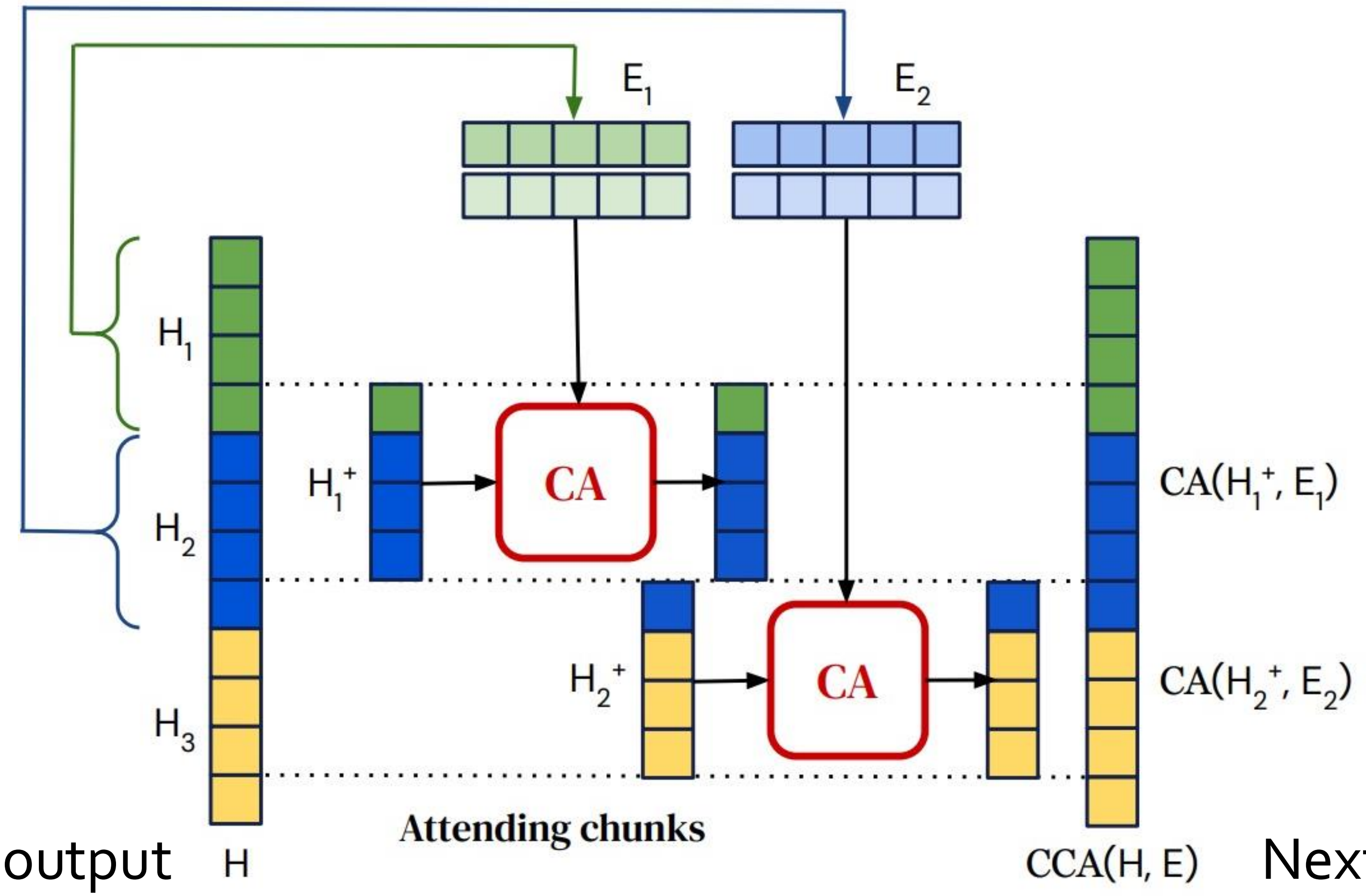
Encoded neighbors



Previous layer output

Retrieved Results Integration: Intermediate-layer Integration

Encoded neighbors



Previous layer output

Next layer input

PART 2: Architecture of RA-LLMs and Main Modules

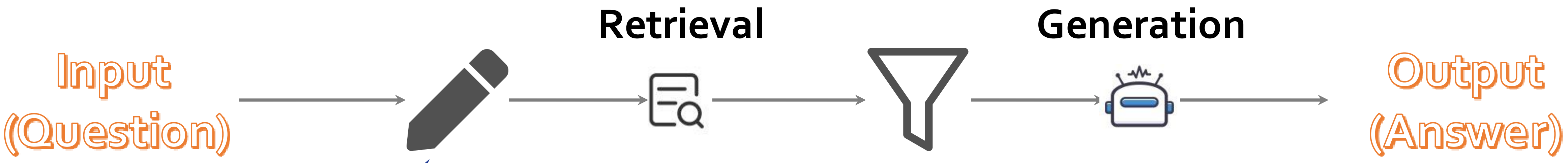
Slides

Website

Website of this tutorial

- RA-LLM architecture overview
- Retriever in RA-LLMs
- Retrieval results integration
- Pre/Post-retrieval techniques
- Special RA-LLM paradigms

Pre/Post-Retrieval Techniques



Pre-retrieval process: to improve the adaptation and effectiveness of the query

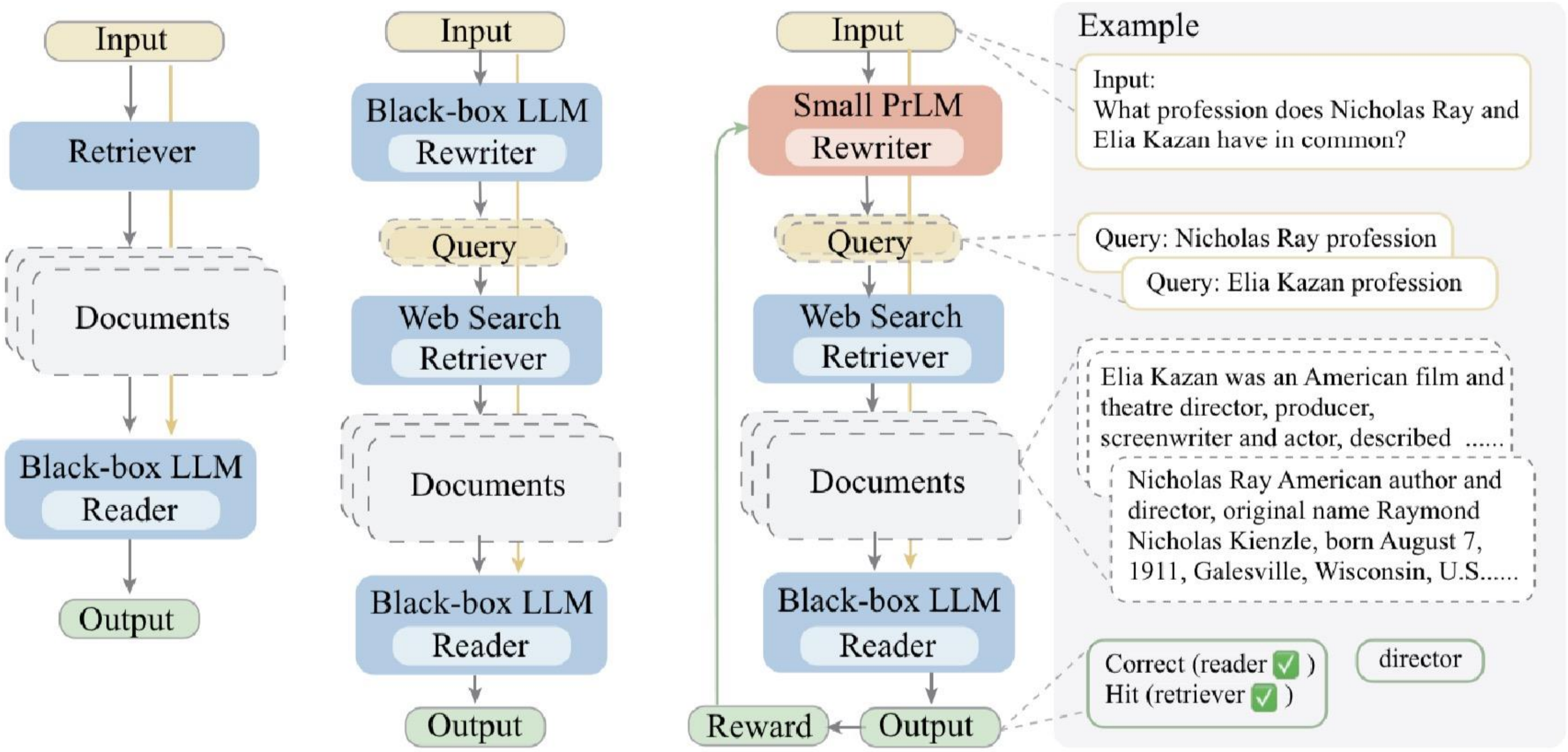
- Query rewriting
- Query decomposition
- Query expansion

Post-retrieval process: to select better results, merge multiple ones, etc

- Re-ranking
- Compression
- Correction

Pre-Retrieval Techniques

❑ **Query Rewriting**: to improve the adaptation of the query



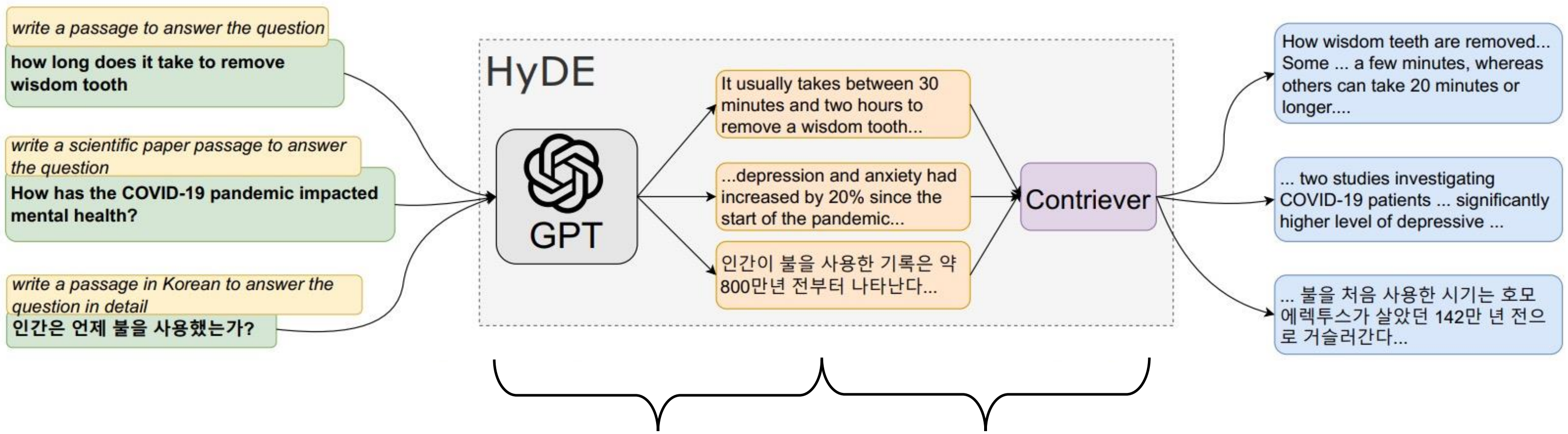
Model	EM	F_1
HotpotQA		
Direct	32.36	43.05
Retrieve-then-read	30.47	41.34
LLM rewriter	32.80	43.85
Trainable rewriter	34.38	45.97
AmbigNQ		
Direct	42.10	53.05
Retrieve-then-read	45.80	58.50
LLM rewriter	46.40	58.74
Trainable rewriter	47.80	60.71
PopQA		
Direct	41.94	44.61
Retrieve-then-read	43.20	47.53
LLM rewriter	46.00	49.74
Trainable rewriter	45.72	49.51

Works on different QA settings

Wang et al. 2023. “Query rewriting for retrieval-augmented large language models”

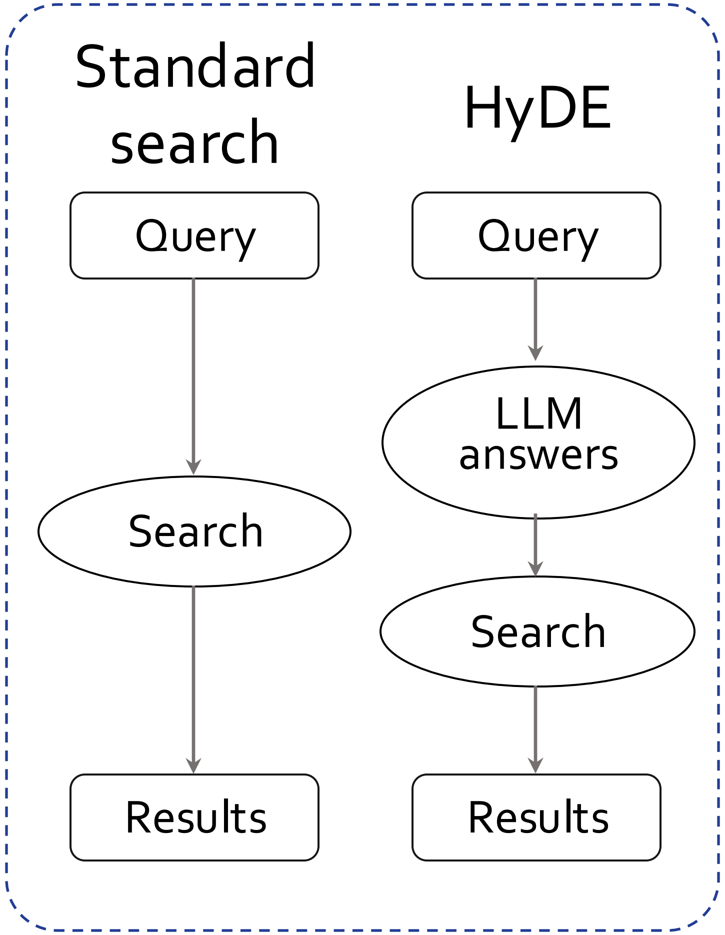
Pre-Retrieval Techniques

- **HyDE**: Hypothetical Document Embeddings



1. Generative task

2. Document-document similarity task



Gao et al. 2022. "Precise zero-shot dense retrieval without relevance labels"

Pre-Retrieval Techniques

❑ Query Expansion

LLM Prompts

Write a passage that answers the given query:

Query: what state is this zip code 85282
Passage: Welcome to TEMPE, AZ 85282. 85282 is a rural zip code in Tempe, Arizona. The population is primarily white...
...
Query: when was pokemon green released
Passage:

Method	Fine-tuning	MS MARCO dev MRR@10	MS MARCO dev R@50	MS MARCO dev R@1k	TREC DL 19 nDCG@10
Sparse retrieval					
BM25	✗	18.4	58.5	85.7	51.2*
+ query2doc	✗	$21.4^{+3.0}$	$65.3^{+6.8}$	$91.8^{+6.1}$	**66.2**$^{+15.0}$
BM25 + RM3	✗	15.8	56.7	86.4	52.2
docT5query (Nogueira and Lin)	✓	**27.7**	**75.6**	**94.7**	64.2
Dense retrieval w/o distillation					
ANCE (Xiong et al., 2021)	✓	33.0	-	95.9	64.5
HyDE (Gao et al., 2022)	✗	-	-	-	61.3
$DPR_{bert\text{-}base}$ (our impl.)	✓	33.7	80.5	95.9	64.7
+ query2doc	✓	**35.1**$^{+1.4}$	**82.6**$^{+2.1}$	**97.2**$^{+1.3}$	**68.7**$^{+4.0}$

New query = original query + generated documents

$$q^+ = \text{concat}(q,\ [\text{SEP}],\ d')$$

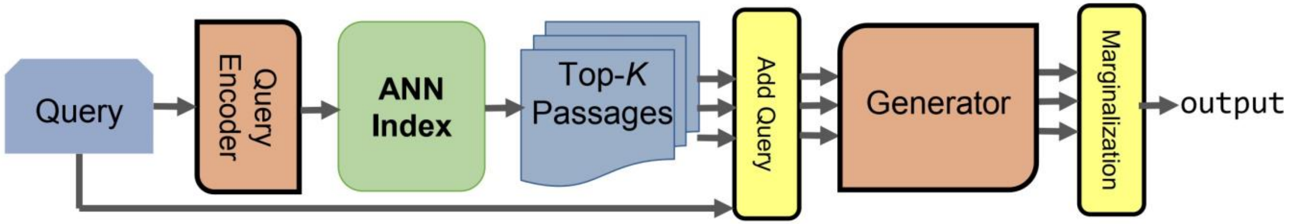
Works for both sparse and dense retrievers

Wang et al. 2023. "Query2doc: Query Expansion with Large Language Models"

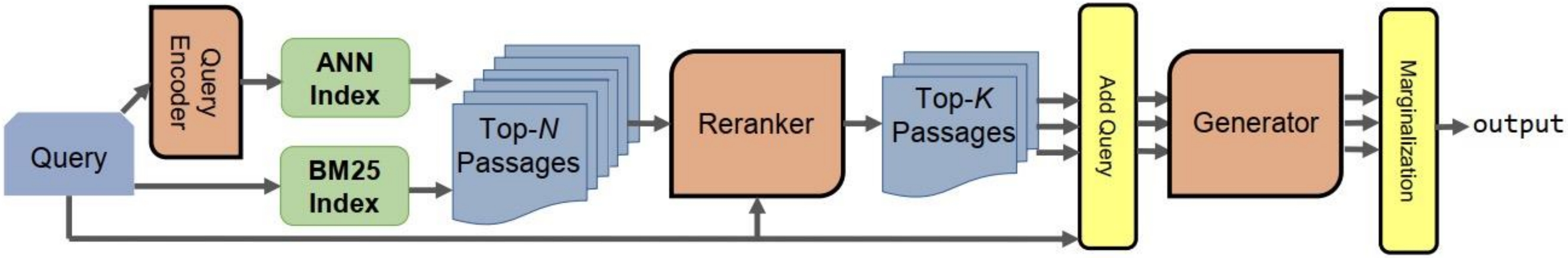
Post-Retrieval Techniques

❑ Retrieved Result Rerank (Re2G)

- ❖ Results from initial retrieval can be greatly improved through the use of a reranker
- ❖ Reranker allows merging retrieval results from sources with incomparable scores, e.g., BM25 and neura initial retrieval



Regular RAG pipeline

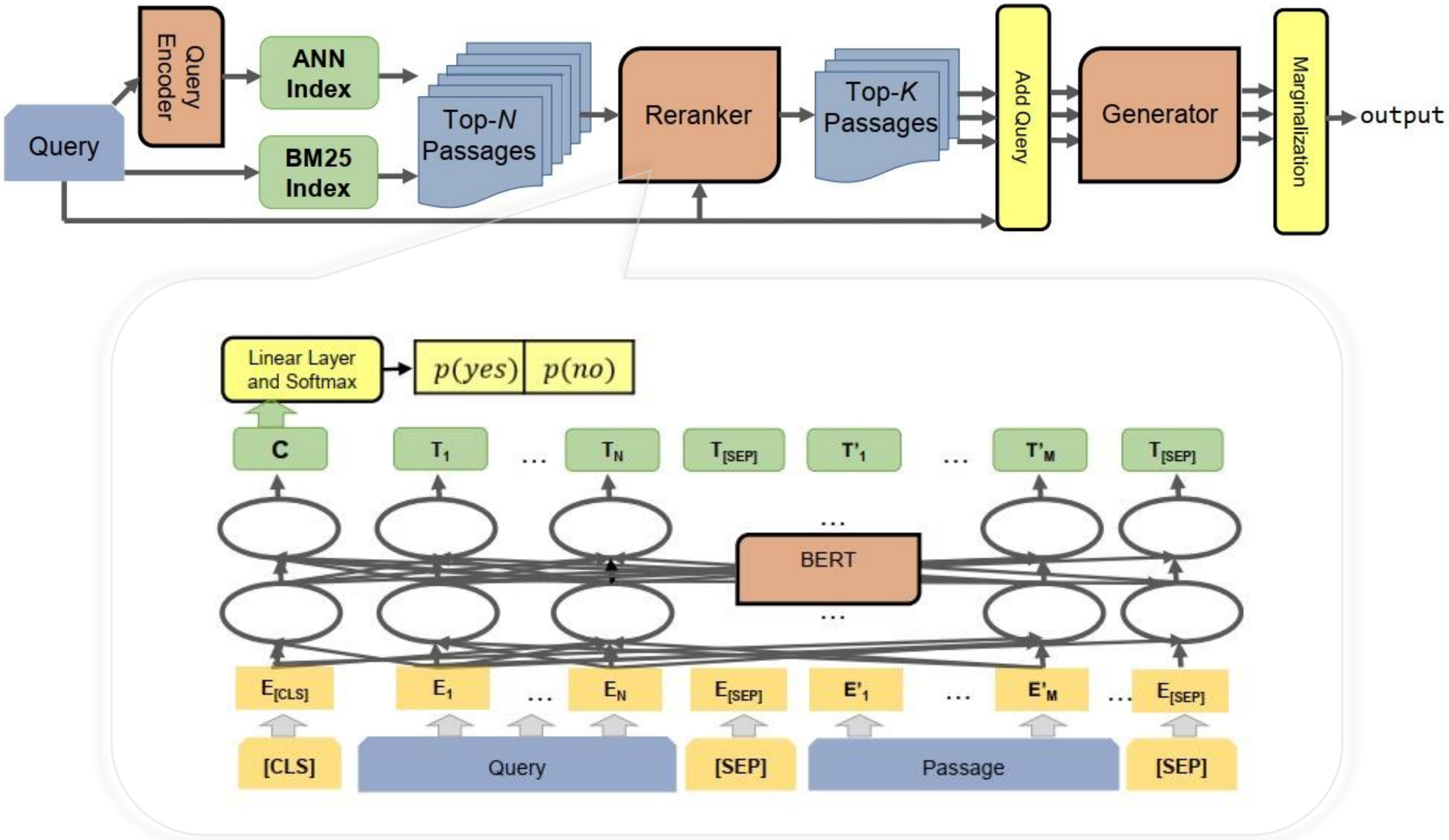


RAG with Reranker

Glass et al. 2022. "Re2G: Retrieve, Rerank, Generate"

Retrieved Result Rerank (Re2G) Model

- **Reranker: interaction model based on the sequence-pair classification**



Nogueira and Cho, 2019, Passage Re-ranking with BERT

Retrieved Result Rerank (Re2G) Performance

	T-REx		NQ		TriviaQA		FEVER		WoW	
	R-Prec	R@5	R-Prec	R@5	R-Prec	R@5	R-Prec	R@5	R-Prec	R@5
BM25	46.88	69.59	24.99	42.57	26.48	45.57	42.73	70.48	27.44	45.74
DPR Stage 1	49.02	63.34	56.64	64.38	60.12	64.04	75.49	84.66	34.74	60.22
KGI_0 DPR	65.02	75.52	64.65	69.60	60.55	63.65	80.34	86.53	**48.04**	**71.02**
Re^2G DPR	**67.16**	**76.42**	**65.88**	**70.90**	**62.33**	**65.72**	**84.13**	**87.90**	47.09	69.88
KGI_0 DPR+BM25	60.48	80.06	36.91	66.94	40.81	64.79	65.95	90.34	35.63	68.47
Reranker Stage 1	81.22	87.00	70.78	73.05	**71.80**	**71.98**	87.71	92.43	55.50	**74.98**
Re^2G Reranker	**81.24**	**88.58**	**70.92**	**74.79**	60.37	70.61	**90.06**	**92.91**	**57.89**	74.62

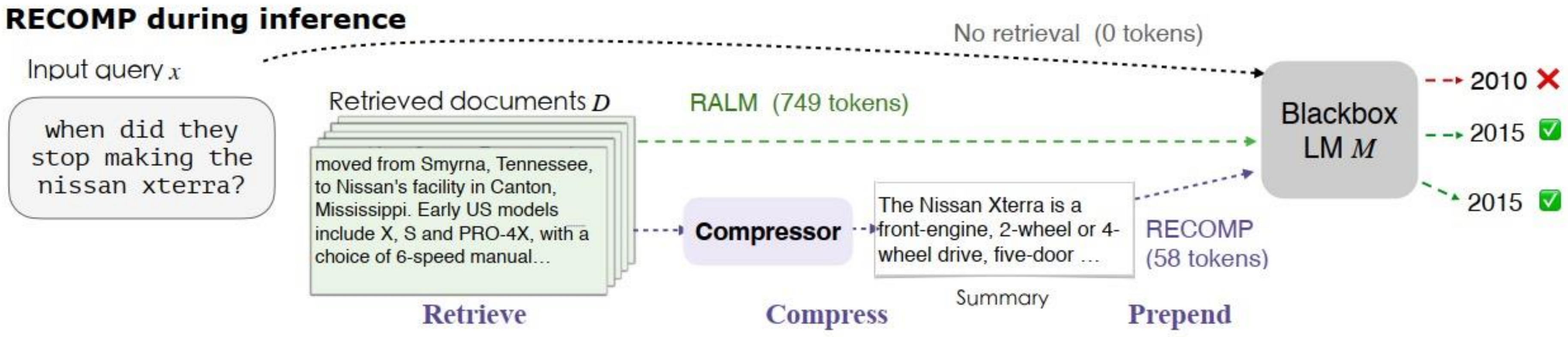
Significantly outperforms pipelines without the *Rerank* stage

Glass et al. 2022. "Re2G: Retrieve, Rerank, Generate"

Post-Retrieval Techniques

❑ Retrieved Result Compression

- ❖ To reduce the computational costs and also relieve the burden of LMs to identify relevant information in long retrieved documents.



❑ Compressor Learning Objectives

- ❖ Concise
- ❖ Effective
- ❖ Faithful

Xu et al. 2023. “RECOMP: Improving retrieval- augmented LMs with context compression and selective augmentation”

Retrieved Result Compression Performance

- QA tasks

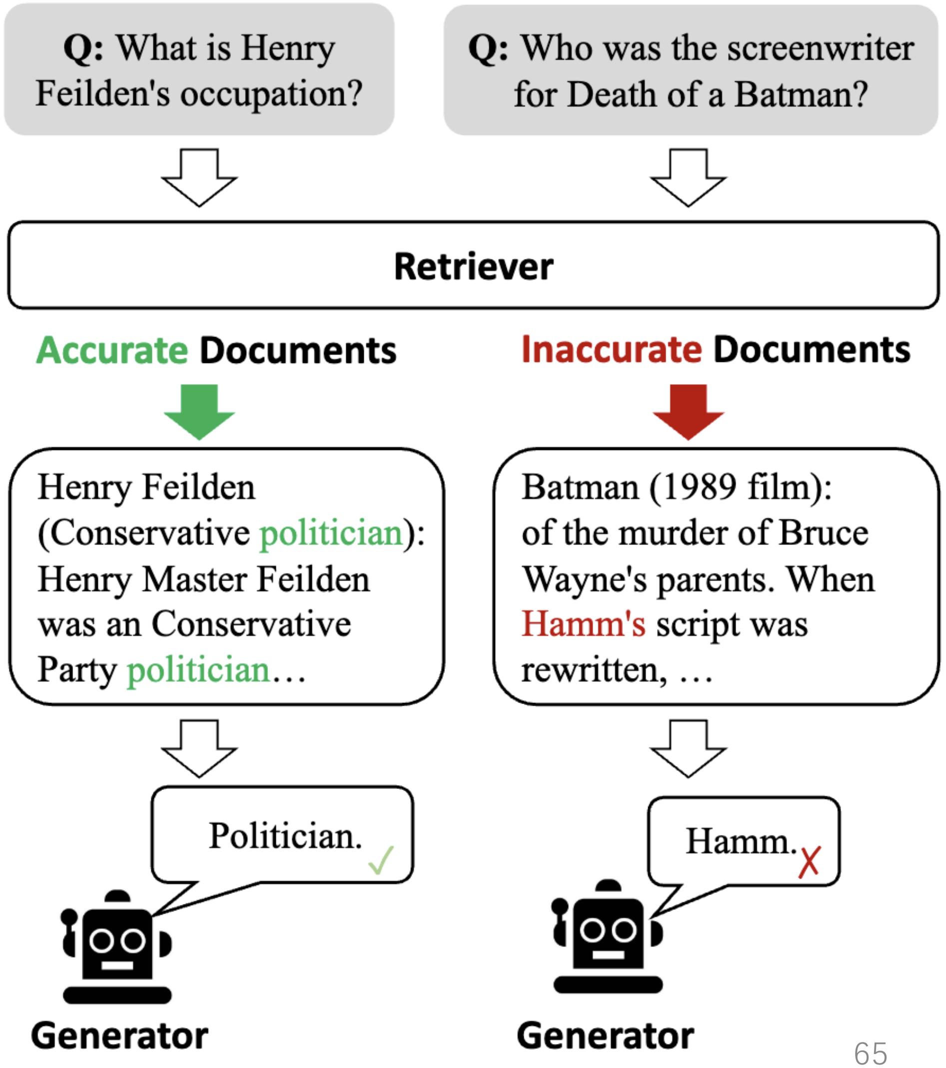
In-Context evidence	NQ # tok	NQ EM	NQ F1	TQA # tok	TQA EM	TQA F1	HotpotQA # tok	HotpotQA EM	HotpotQA F1
-	0	21.99	29.38	0	49.33	54.85	0	17.80	26.10
RALM without compression									
Top 1 documents	132	33.07	41.45	136	57.84	64.94	138	28.80	40.58
Top 5 documents	660	**39.39**	**48.28**	677	**62.37**	**70.09**	684	**32.80**	**43.90**
Phrase/token level compression									
Top 5 documents (NE)	338	23.60	31.02	128	54.96	61.19	157	22.20	31.89
Top 5 documents (BoW)	450	28.48	36.84	259	58.16	65.15	255	25.60	36.00
Extractive compression of top 5 documents									
Oracle	34	60.22	64.25	32	79.29	82.06	70	41.80	51.07
Random	32	23.27	31.09	31	50.18	56.24	61	21.00	29.86
BM25	36	25.82	33.63	37	54.67	61.19	74	26.80	38.02
DPR	39	34.32	43.38	41	56.58	62.96	78	27.40	38.15
Contriever	36	30.06	31.92	40	53.67	60.01	78	28.60	39.48
Ours	37	36.57	44.22	38	**58.99**	65.26	75	**30.40**	**40.14**

Outperforms representative sparse and dense retrievers

Xu et al. 2023. "RECOMP: Improving retrieval- augmented LMs with context compression and selective augmentation"

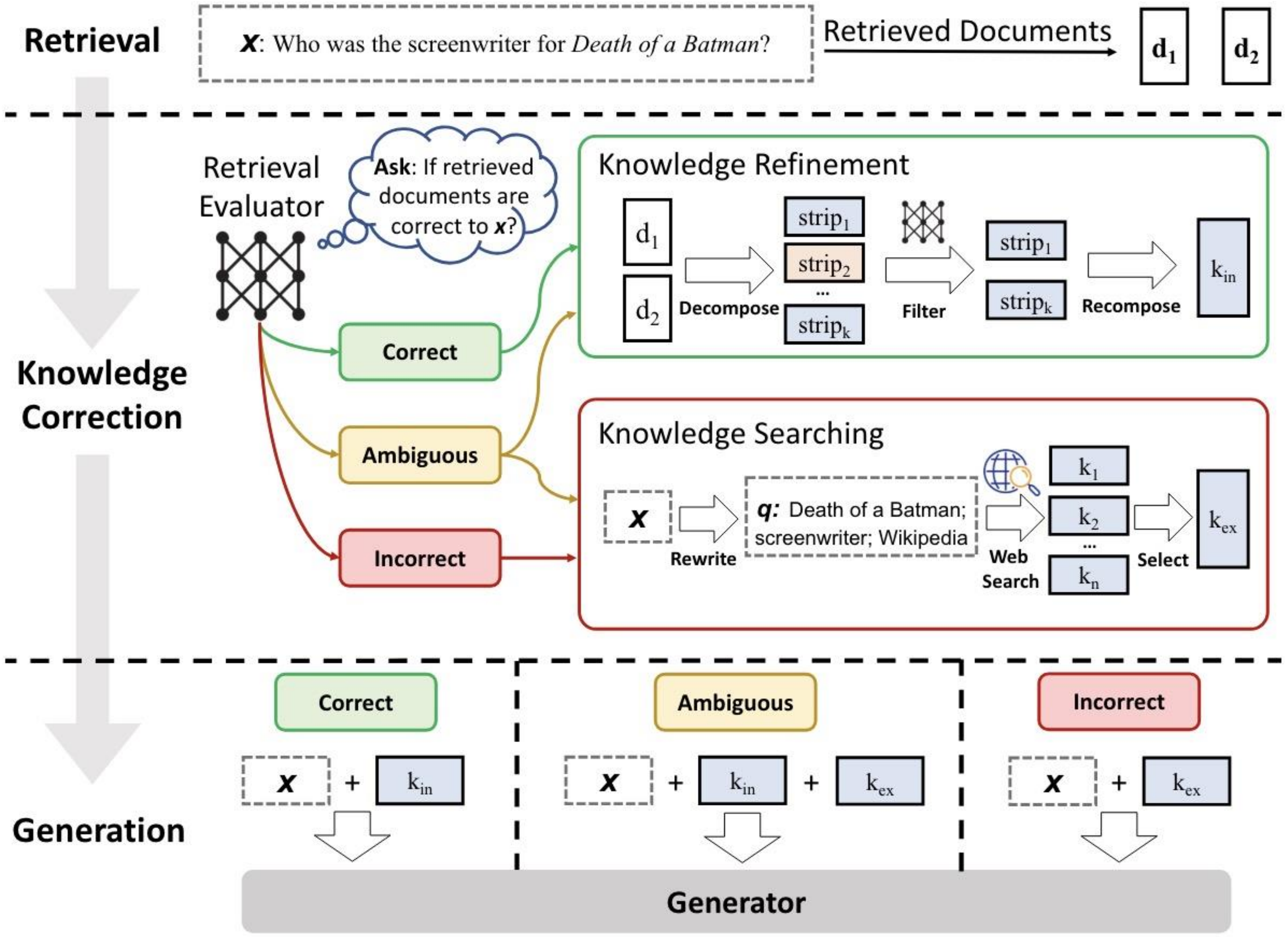
Post-Retrieval Techniques: Corrective RAG

- Corrective Retrieval Augmented Generation (CRAG)
  - Although retrieval-augmented generation (RAG) is a practicable complement to LLMs, it relies heavily on the **relevance of retrieved documents**
  - A lightweight **retrieval evaluator** is designed to assess the overall quality of retrieved documents for a query, returning a **confidence degree** based on which different knowledge retrieval actions can be triggered



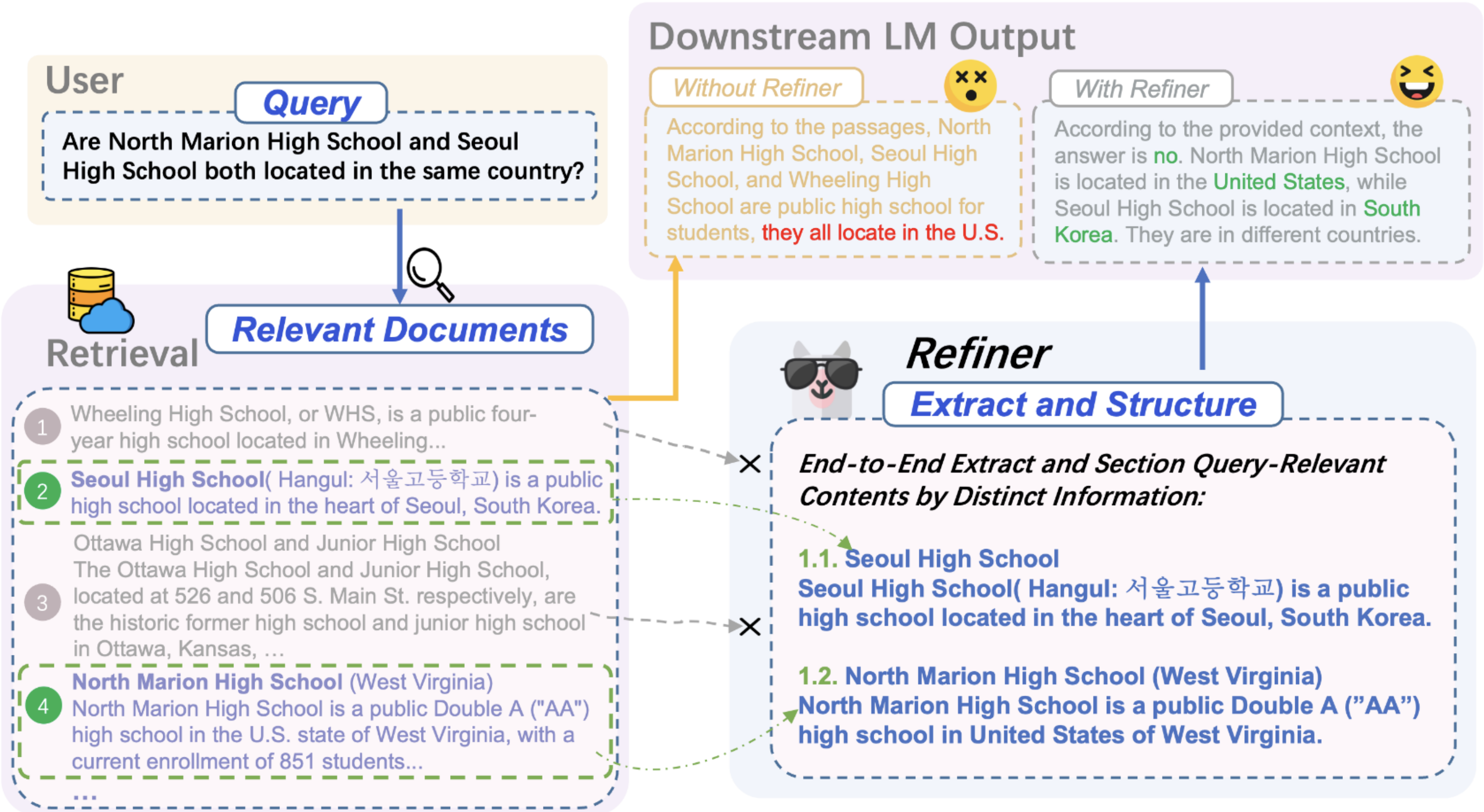
Yan et al., 2024, Corrective Retrieval Augmented Generation

Post-Retrieval Techniques: Corrective RAG



Post-Retrieval Techniques: Refiner

- **Refiner**: leveraging a single decoder-only LLM to adaptively extract query relevant contents verbatim along with the necessary context



Li et al., 2024, Refiner: Restructure Retrieval Content Efficiently to Advance Question-Answering Capabilities

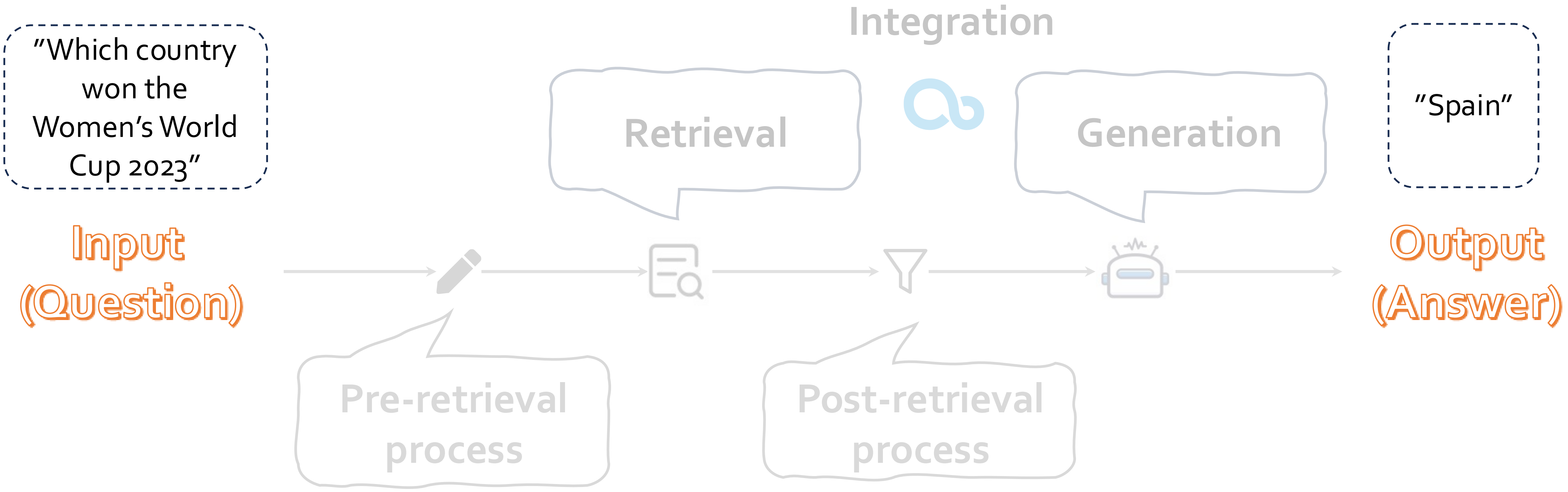
PART 2: Architecture of RA-LLMs and Main Modules

Slides

Website of this tutorial

- RA-LLM architecture overview
- Retriever in RA-LLMs
- Retrieval results integration
- Pre/Post-retrieval techniques
- Special RA-LLM paradigms

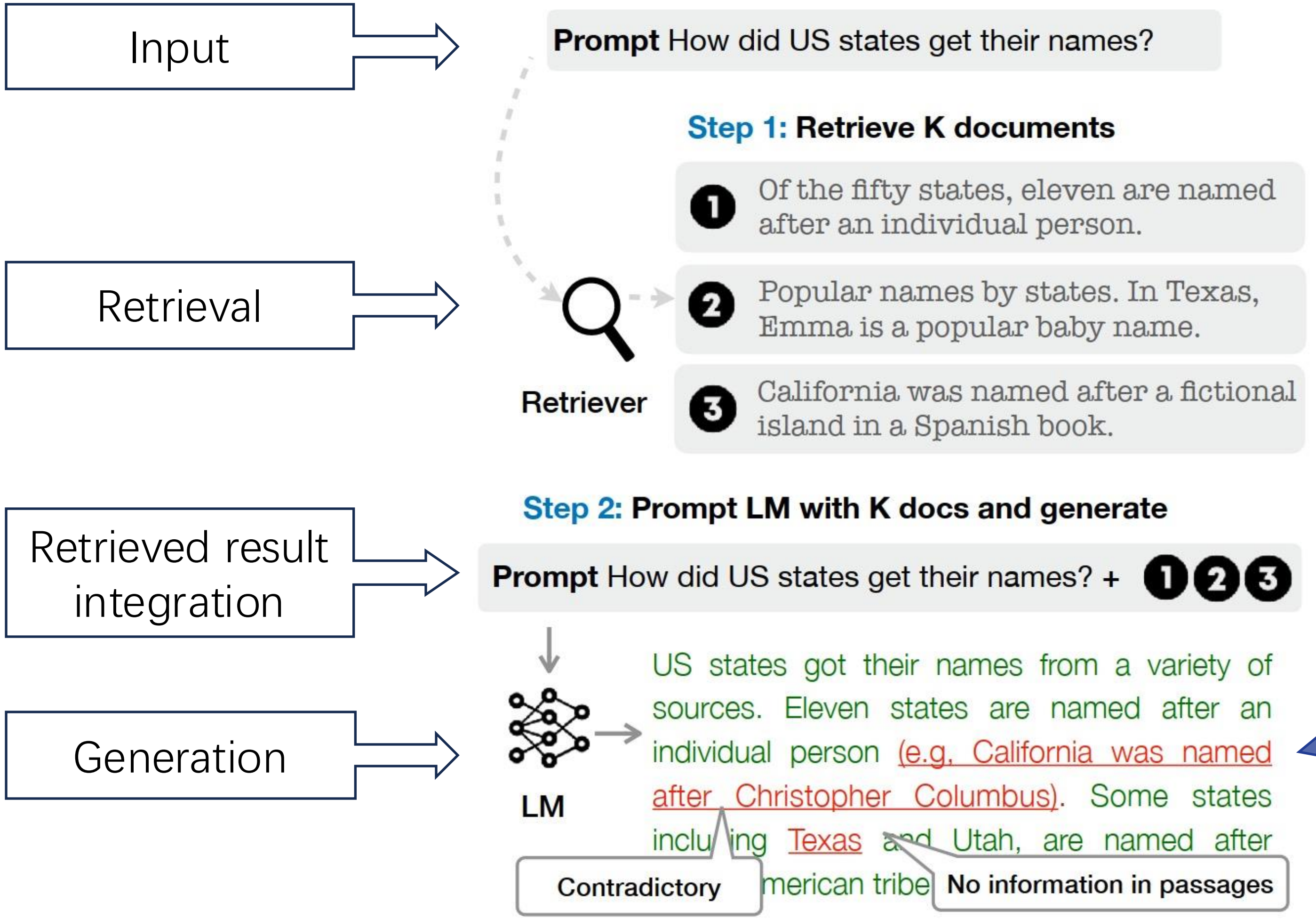
Beyond Standard Pipelines and Components?



More?

Special RAG Pipeline: Self-Reflective RAG (SELF-RAG)

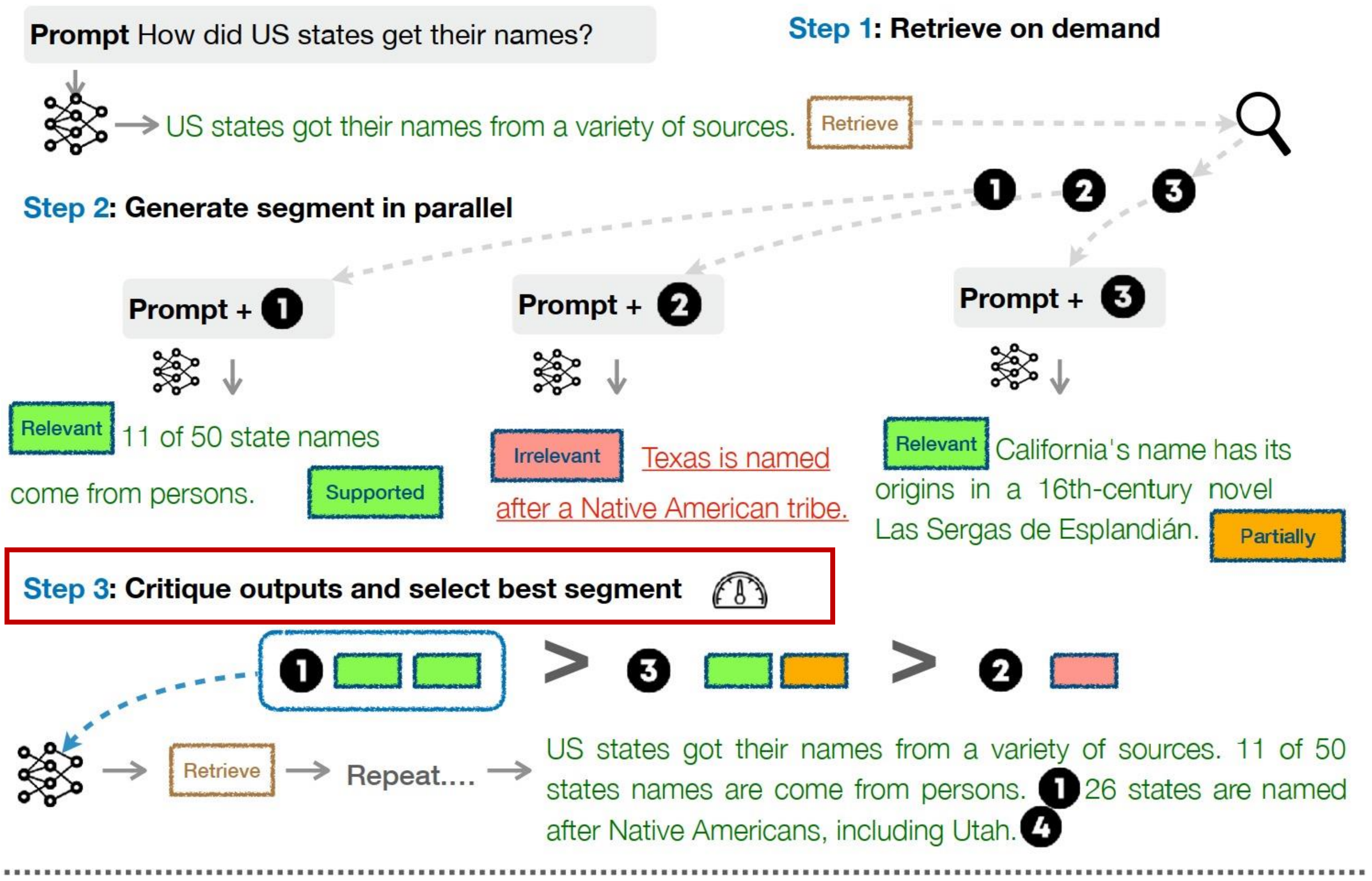
- General Retrieval-Augmented Generation (RAG)



Retrieval-enhanced **generation results** are not necessarily useful or helpful !

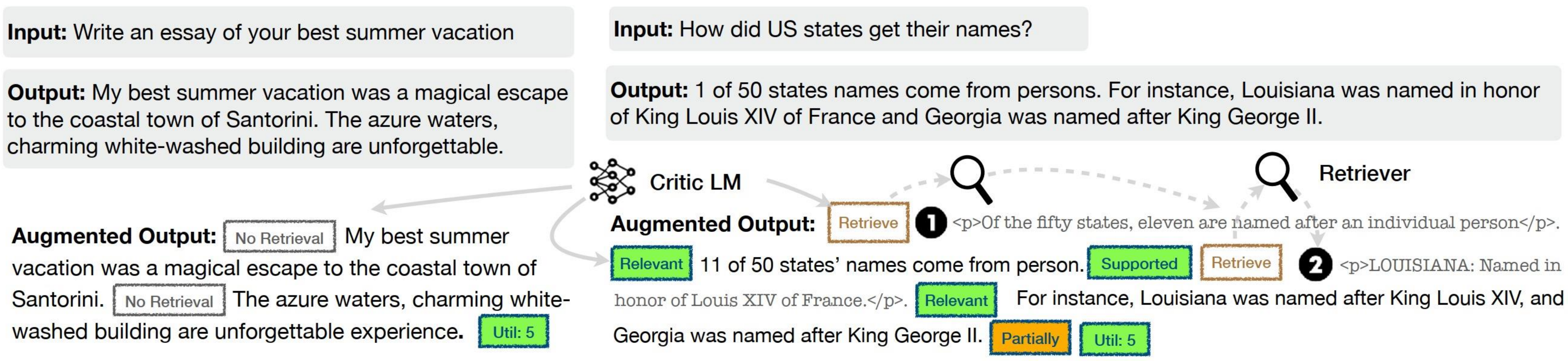
Asai A, et al.2023. "Self-rag: Learning to retrieve, generate, and critique through self-reflection"

SELF-RAG Overview



Key Technical Design in SELF-RAG

❑ Critic Model Training



❑ Four types of reflection tokens used in SELF-RAG

Type	Input	Output	Definitions
Retrieve	x / x, y	{yes, no, continue}	Decides when to retrieve with $\mathcal{R}$
IsRel	x, d	{**relevant**, irrelevant}	d provides useful information to solve x.
IsSup	x, d, y	{**fully supported**, partially supported, no support}	All of the verification-worthy statement in y is supported by d.
IsUse	x, y	{**5**, 4, 3, 2, 1}	y is a useful response to x.

SELF-RAG Algorithm

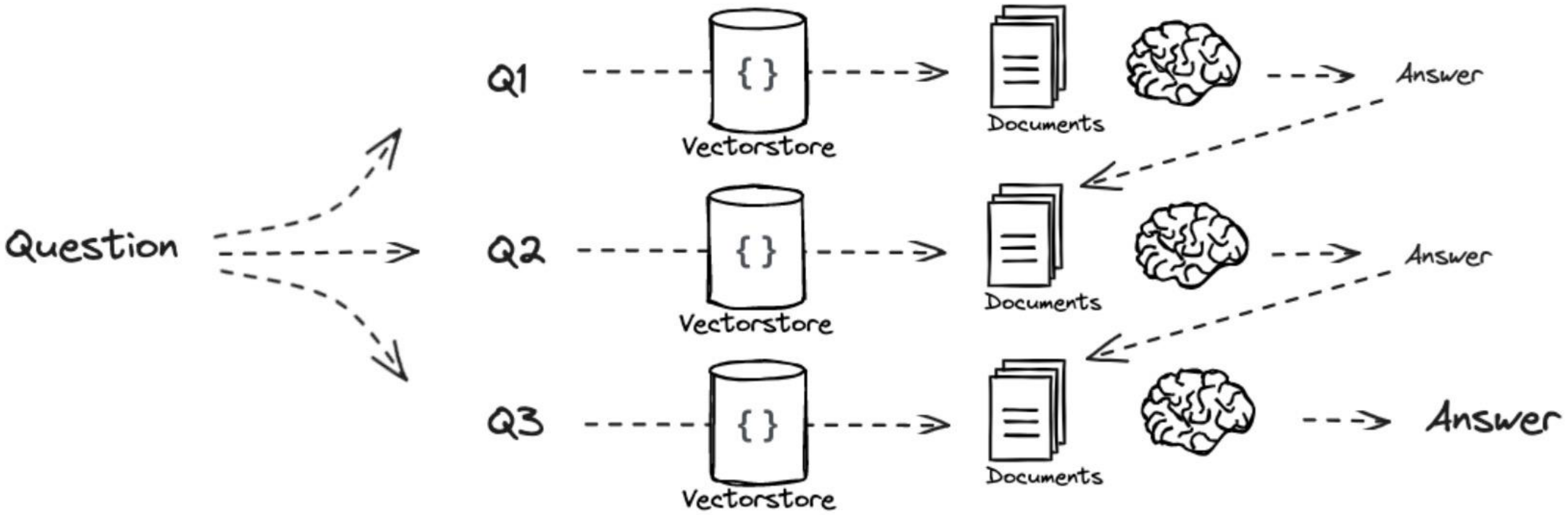
Algorithm 1 SELF-RAG Inference

Require: Generator LM $\mathcal{M}$, Retriever $\mathcal{R}$, Large-scale passage collections $\{d_1, \ldots, d_N\}$

Input: input prompt x and preceding generation $y_{<t}$, **Output:** next output segment y_t
$\mathcal{M}$ predicts [Retrieve] given $(x, y_{<t})$
if [Retrieve] == `Yes` **then**
    Retrieve relevant text passages $\mathbf{D}$ using $\mathcal{R}$ given (x, y_{t-1}) ▷ Retrieve
    $\mathcal{M}$ predicts [IsRel] given x, d and y_t given $x, d, y_{<t}$ for each $d \in \mathbf{D}$ ▷ Generate
    $\mathcal{M}$ predicts [IsSup] and [IsUse] given x, y_t, d for each $d \in \mathbf{D}$ ▷ Critique
    Rank y_t based on [IsRel], [IsSup], [IsUse] ▷ Detailed in Section 3.3
else if [Retrieve] == `No` **then**
    $\mathcal{M}_{gen}$ predicts y_t given x ▷ Generate
    $\mathcal{M}_{gen}$ predicts [IsUse] given x, y_t ▷ Critique

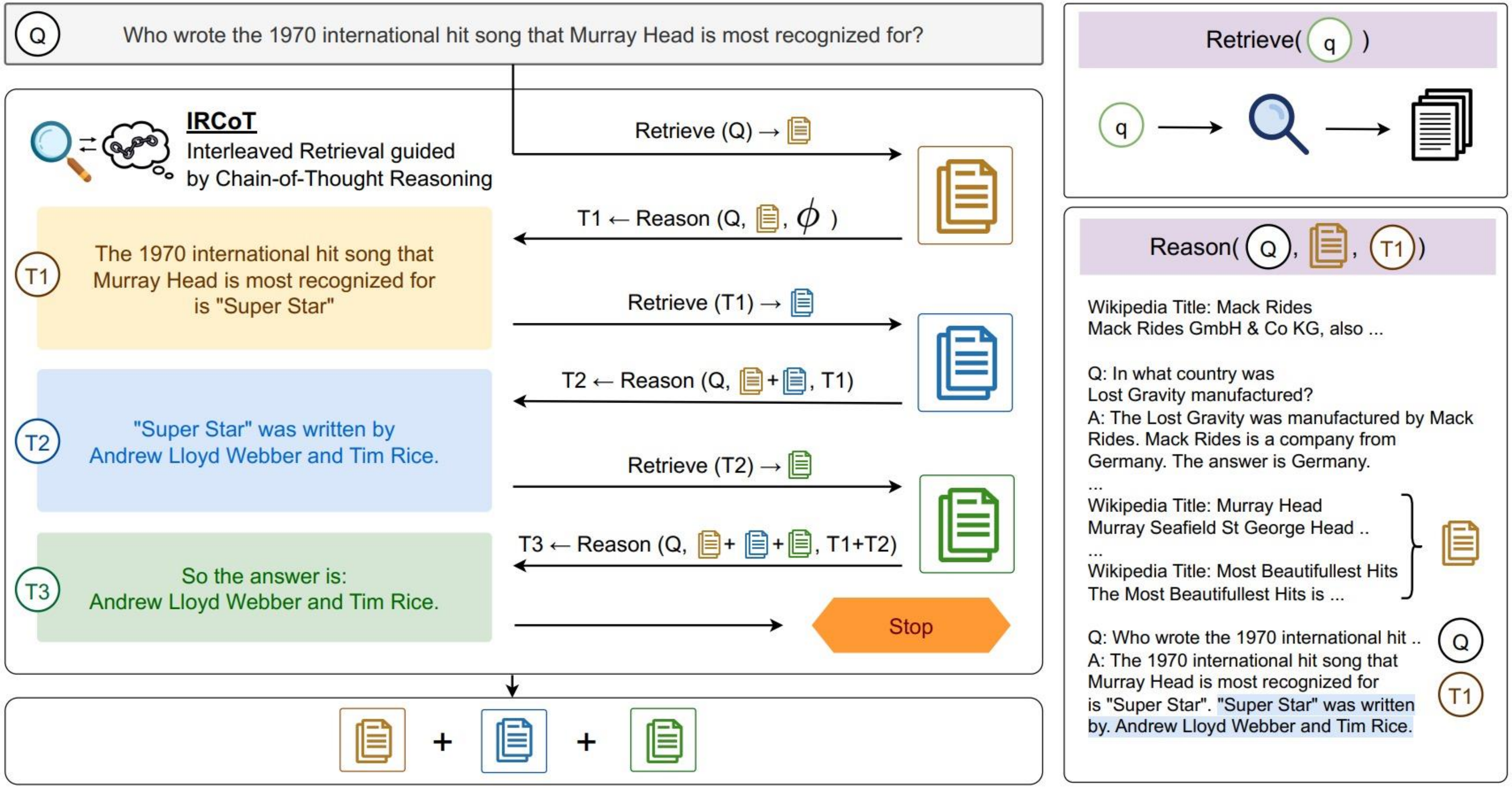
Special RAG Pipeline: Recursively Answer

- Chain-of-Thought + RAG
  - One-step retrieve-and-read approach is insufficient for multi-step QA
  - What to retrieve depends on what has already been derived, which in tern may depend on what was previously retrieved



Zhou, et al.2023. "Least-to-most prompting enables complex reasoning in large language models"
Trivedi, et al. 2023. "Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions"

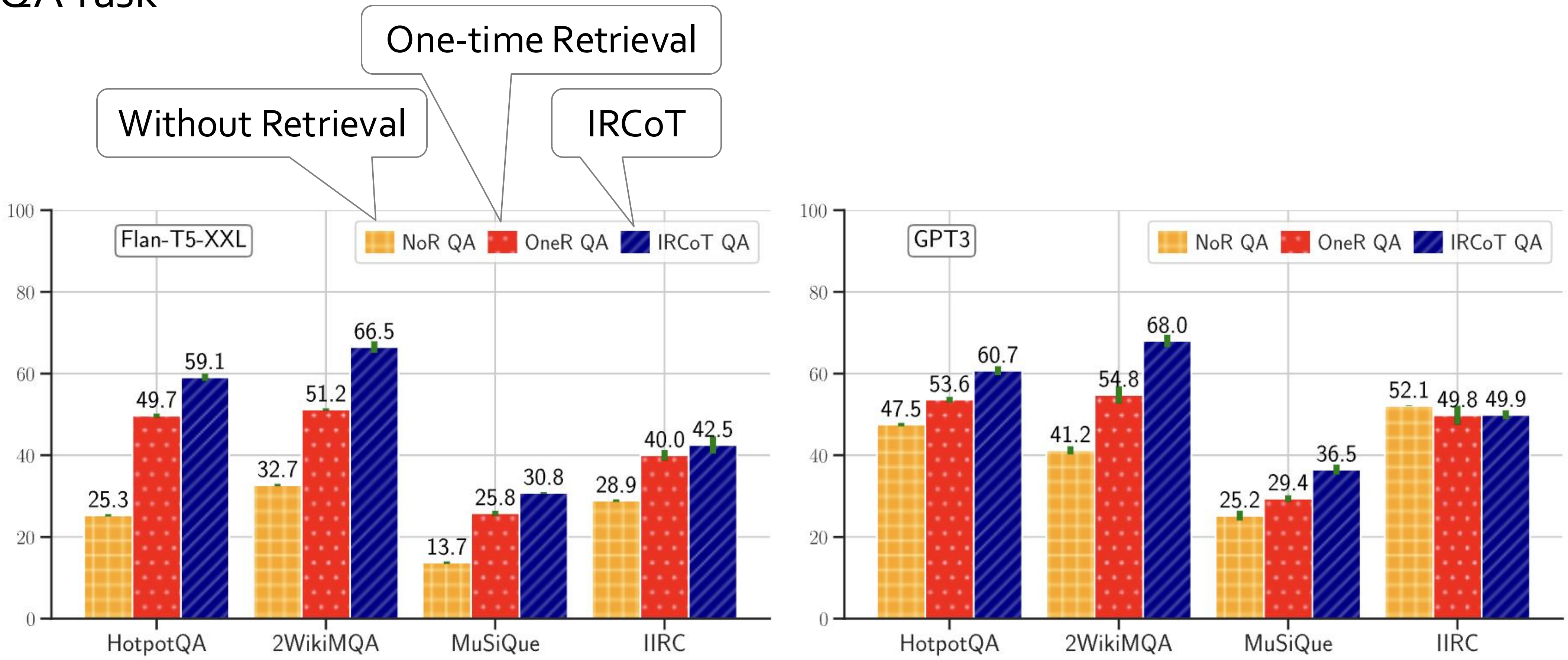
Interleaved Retrieval guided by Chain-of-Thought (IRCoT)



Trivedi, et al. 2023. “Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions”

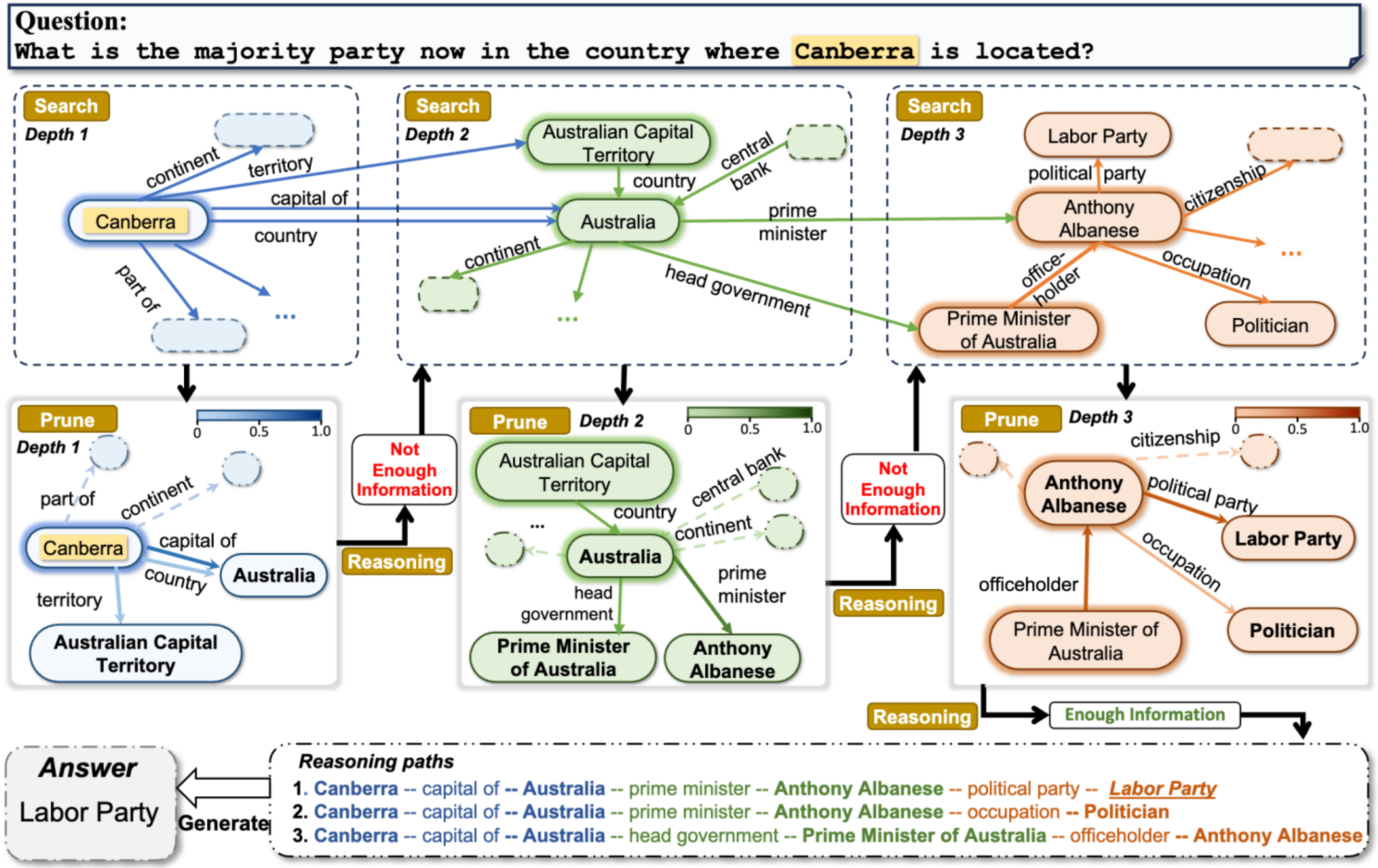
IRCoT Performance

- QA Task



Iterative Retrieval with KG

❑ Think-on-Graph



Sun J, et al., 2024 Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph.

Tutorial Outline

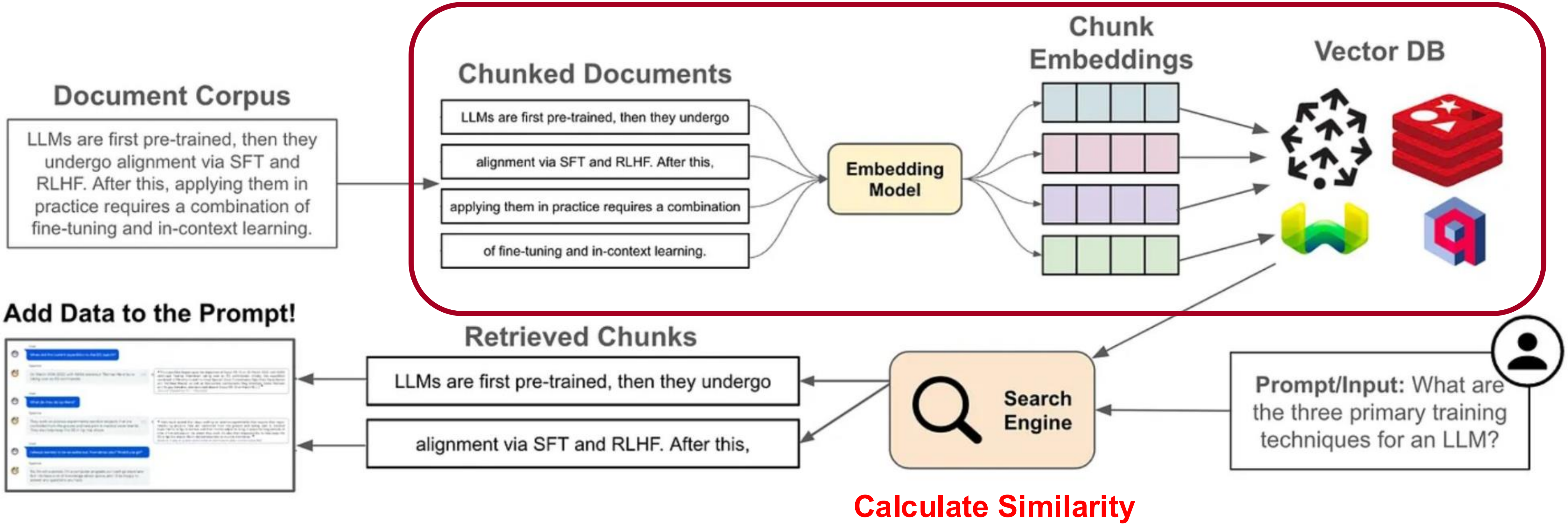
- **Part 1: Introduction** of Retrieval Augmented Large Language Models (RA-LLMs) (Dr. Yujuan Ding)
- **Part 2: Architecture** of RA-LLMs and **Main Modules** (Dr. Yujuan Ding)
- **Part 3: Data Management for RA-LLMs (Pangjing Wu)**
- **Part 4: Learning** Approach of RA-LLMs (Liangbo Ning)
- **Part 5: Applications** of RA-LLMs (Shijie Wang)
- **Part 6: Challenges and Future Directions** of RA-LLMs (Liangbo Ning)

Website of this tutorial
Check out the slides and more information!

Website

Part 3: RA-LLM Data Management

- **Pipeline of RAG with Vector DB**



https://x.com/cwolferesearch/status/1754558231802769857

Part 3: RA-LLM Data Management

Presenter
Pangjing Wu
HK PolyU

- Vector Generation
- Indexing
- Query Processing
- RAG-Oriented VDB Pipeline

Part 3: RA-LLM Data Management

Website

Website of this tutorial

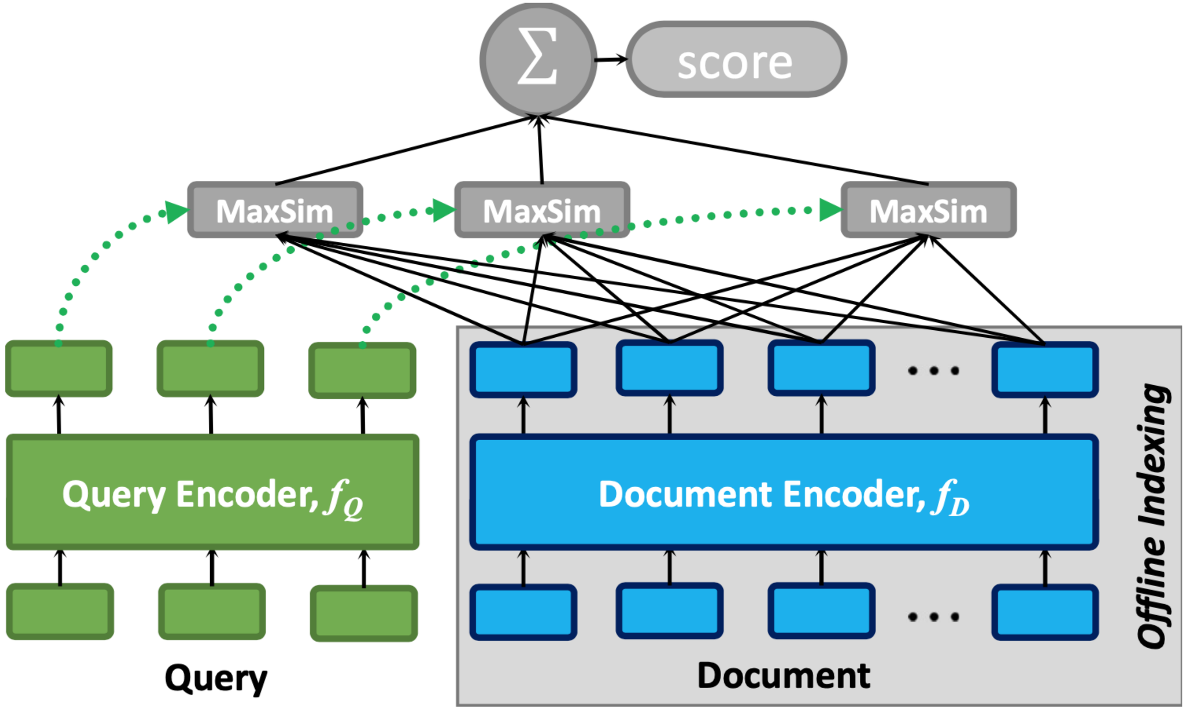
- **Vector Generation**
- Indexing
- Query Processing
- RAG-Oriented VDB Pipeline

RA-LLM Learning: Vector Generation

- Vector generation transforms **documents** into **vectors**.
  - **Dense embedding:** using **low-dimensional dense vectors** capturing semantics.
  - **Sparse embedding:** using **high-dimensional vectors with mostly zero entries**.

RA-LLM Learning: Vector Generation

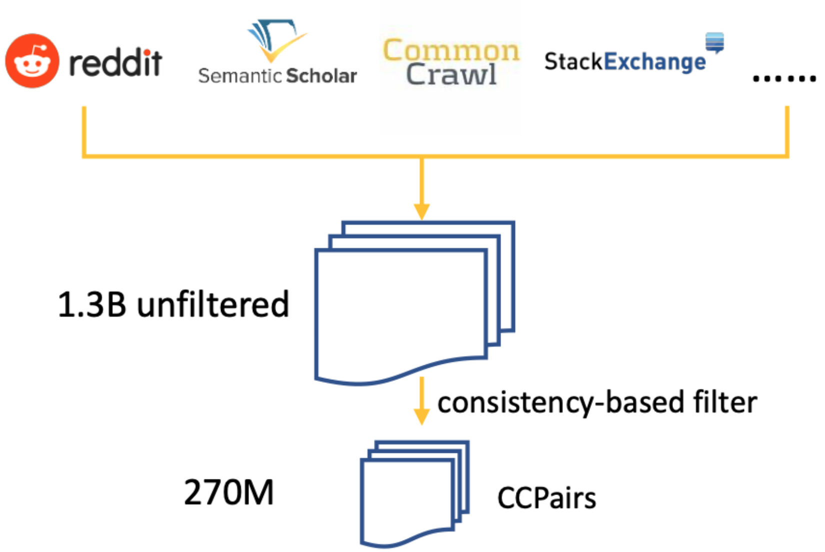
- **ColBERT**
  - Generates query and document vectors separately using BERT.

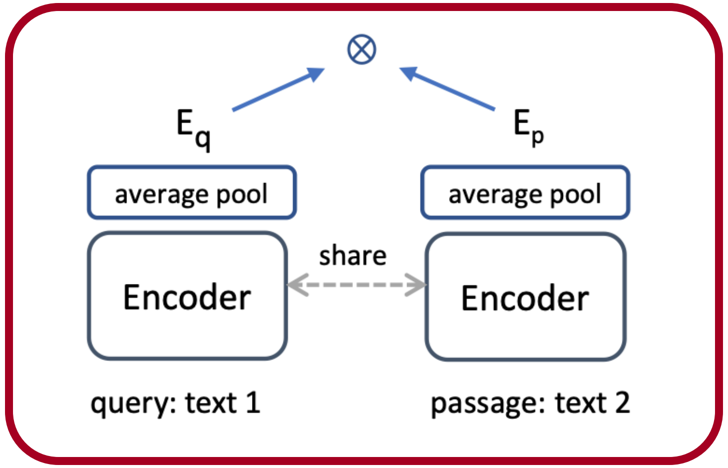


Khattab et al. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT, SIGIR, 2020.

RA-LLM Learning: Vector Generation

- **E5**
  - **Uses a shared encoder with contrastive loss to generate consistent text embeddings.**



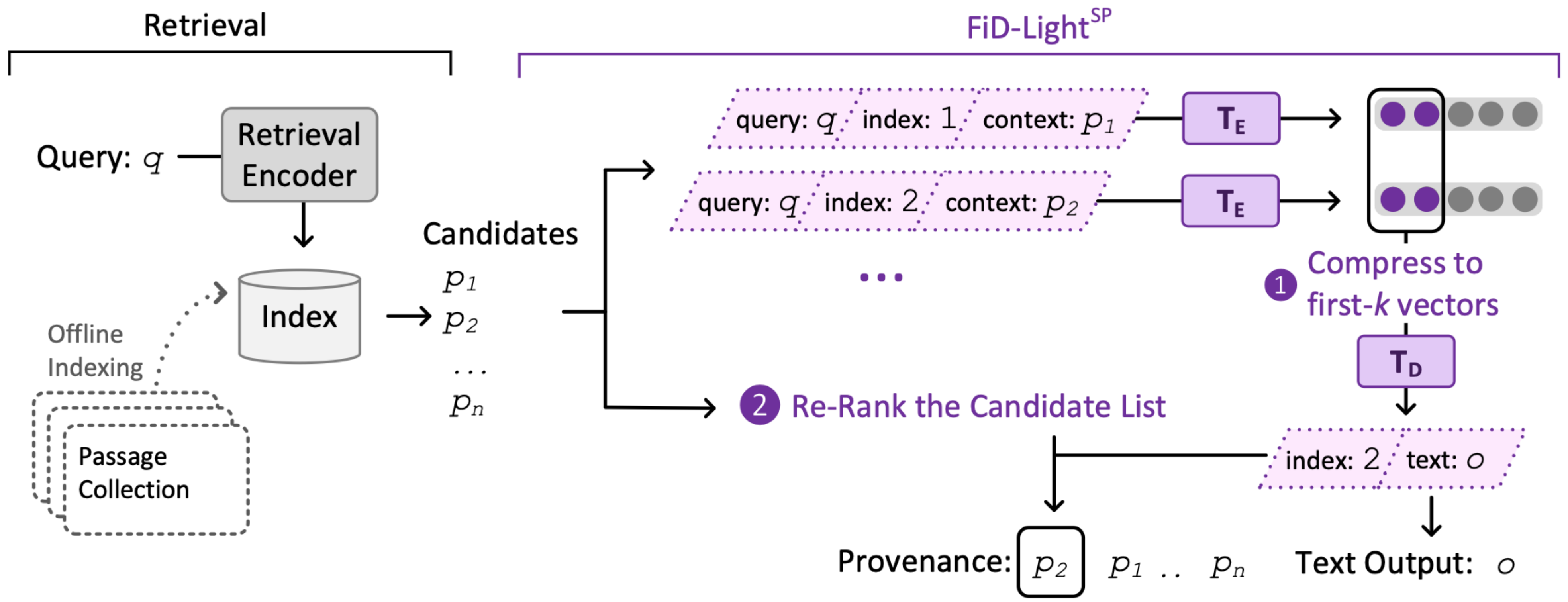


$$\min \quad L_{\text{cont}} = -\frac{1}{n}\sum_i \log \frac{e^{s_\theta(q_i,p_i)}}{e^{s_\theta(q_i,p_i)} + \sum_j e^{s_\theta(q_i,p_{ij}^-)}}$$

Wang et al. Text Embeddings by Weakly-Supervised Contrastive Pre-training, Preprint in ArXiv, 2022.

RA-LLM Learning: Vector Generation

- **FiD-Light**
  - **Compresses encoder outputs into the first-k vectors for efficient ranking.**

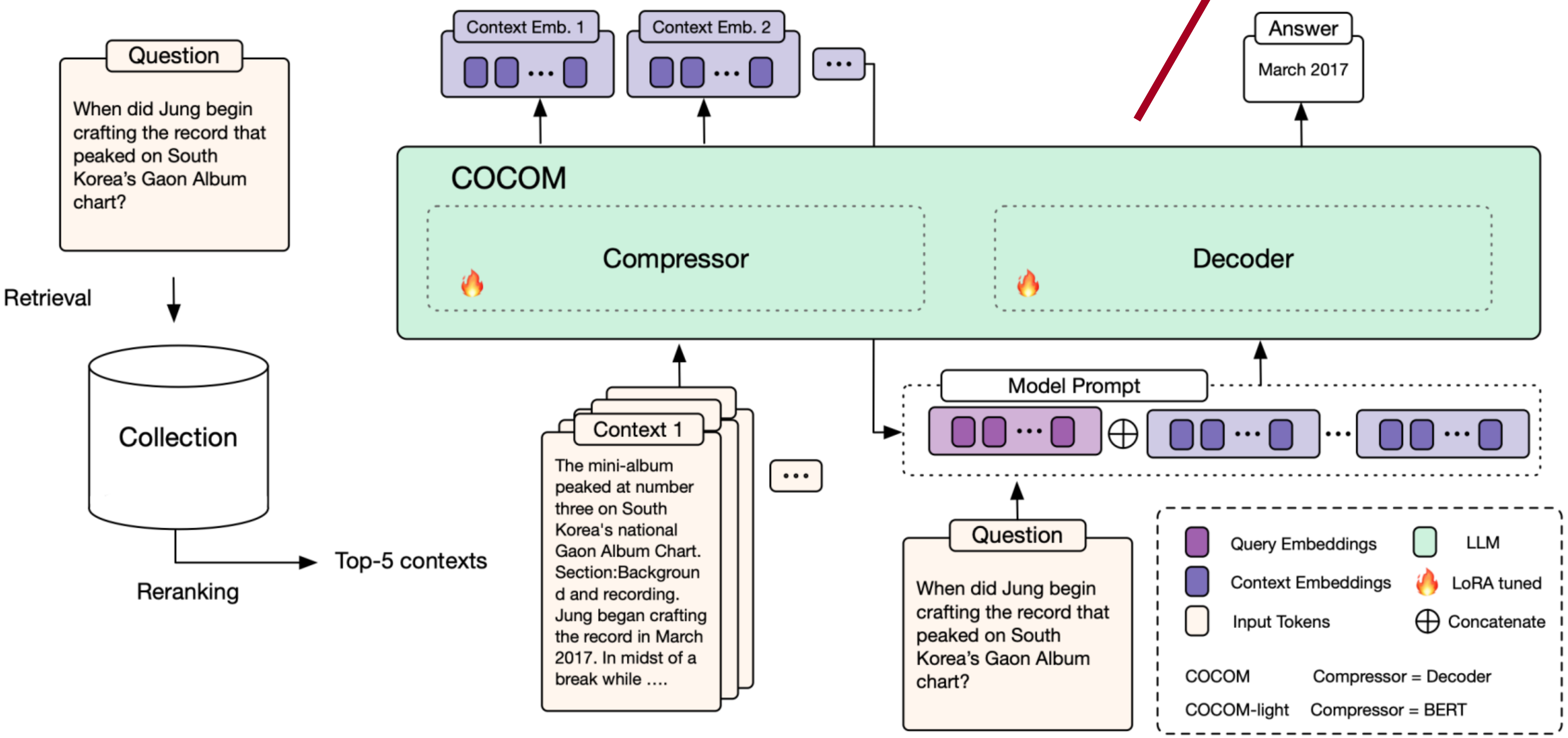


Hofstätter et al. Fid-light: Efficient and effective retrieval-augmented text generation, SIGIR, 2023.

RA-LLM Learning: Vector Generation

- **COCOM**
  - **Compresses multi-context embeddings into a compact form.**

$$\mathcal{L}(\theta_{LLM}, \phi_{comp}) = -\sum_{x_t \in \mathcal{X}_B} \log P_{\theta_{LLM}}(x_t \mid \phi_{comp}(\mathcal{X}_A), x_1, \ldots, x_{t-1})$$



Rau et al. Context Embeddings for Efficient Answer Generation in RAG, WSDM, 2024.

RA-LLM Learning: Vector Generation

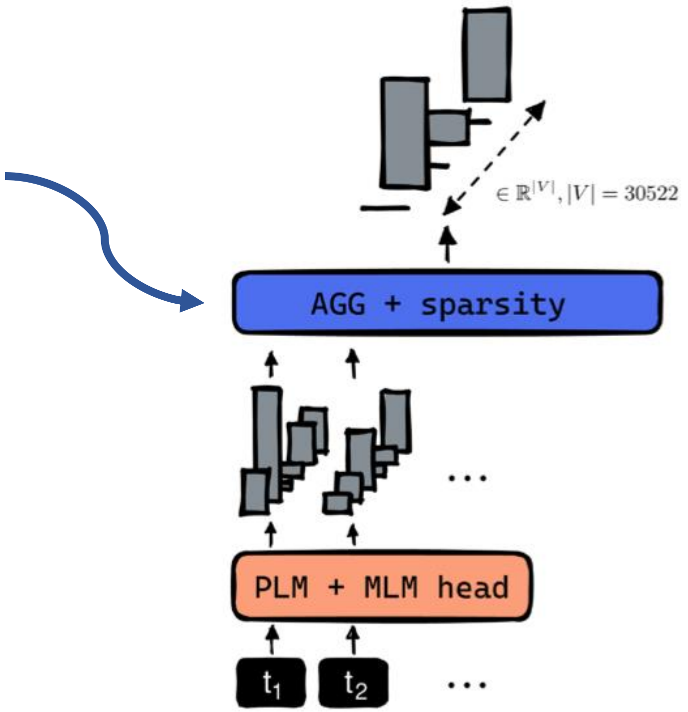
- Vector generation transforms **documents** into **vectors**.
  - **Dense embedding:** using **low-dimensional dense vectors** capturing semantics.
  - **Sparse embedding:** using **high-dimensional vectors with mostly zero entries**.

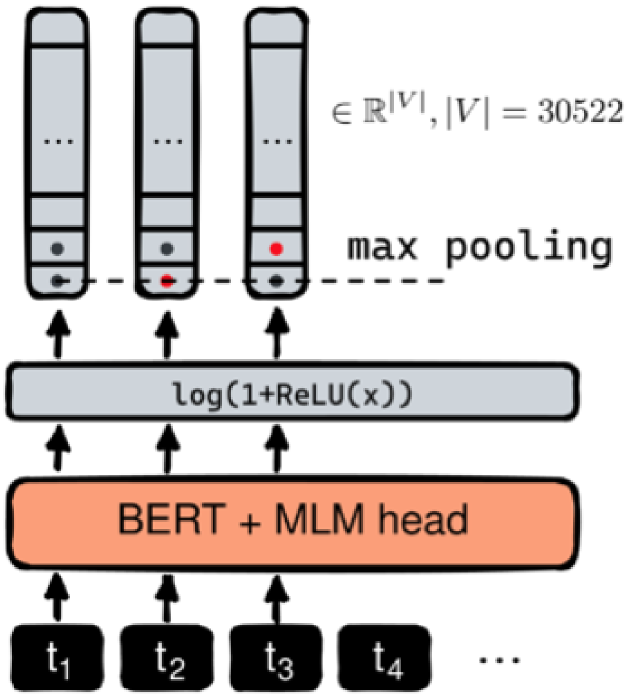
RA-LLM Learning: Vector Generation

- **Efficient-SPLADE**
  - Uses log-saturated term weighting with FLOPS regularization.

$$\tilde{F} = \sum_{v \in V} \left(\frac{1}{b} \sum_{i=1}^{b} |w_v^{(i)}| \right)^2$$

FLOPS

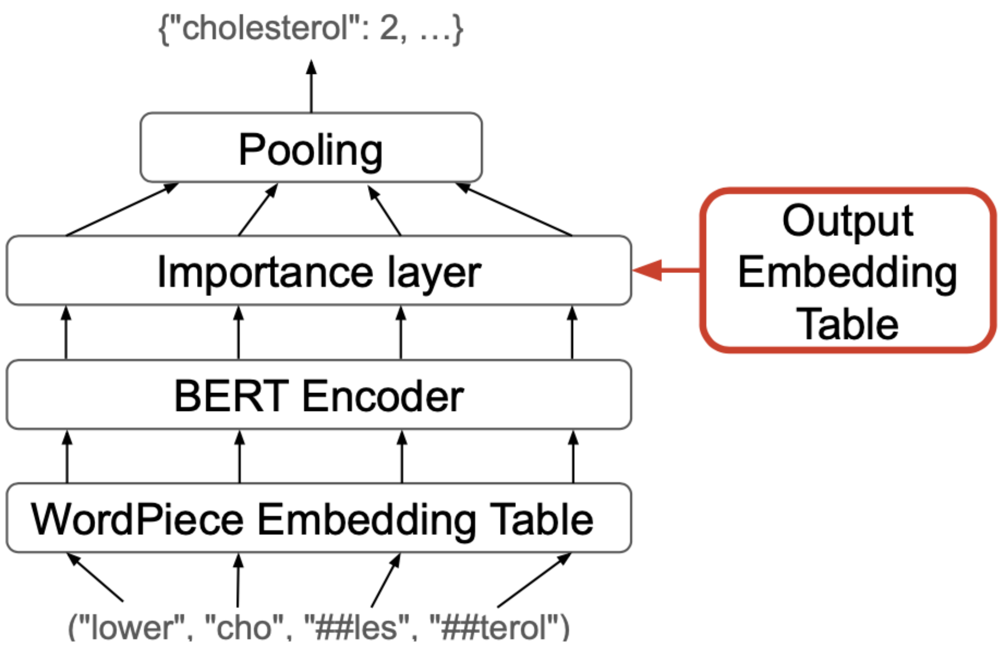




Formal et al. Toward Effective and Efficient Sparse Neural Information Retrieval, TOIS, 2024.

RA-LLM Learning: Vector Generation

- **Expanded-SPLADE**
  - Expands vocabularies by reprojection.



Algorithm 2: Expanded-SPLADE

Inputs: Text t, BERT WordPiece vocab V, Output vocab U
Output: $w^{(t)} \in \mathbb{R}^U$

$(t_1, t_2, \ldots, t_N) \leftarrow$ tokenize t in V
$(h_1, h_2, \ldots, h_N) \leftarrow \mathrm{BERT}(t_1, t_2, \ldots, t_N)$
foreach $1 \le i \le N, j \in U$ **do**
  $w_{ij}^{(t)} = \mathrm{transform}(h_i)^T E'_j + b'_j$
  /* $E'_j \in \mathbb{R}^H$ is a new learned embedding for j */
foreach $j \in U$ **do**
  $w_j^{(t)} = \max_{1 \le i \le N} \log(1 + \mathrm{ReLU}(w_{ij}^{(t)}))$
return $\langle w_j^{(t)} : j \in U \rangle$

Dudek et al. Learning Sparse Lexical Representations Over Specified Vocabularies for Retrieval, CIKM, 2023.

RA-LLM Learning: Vector Generation

- **Dense embedding**

  ✅ **Strong performance in semantic search**

  ❌ **May retrieve irrelevant documents**

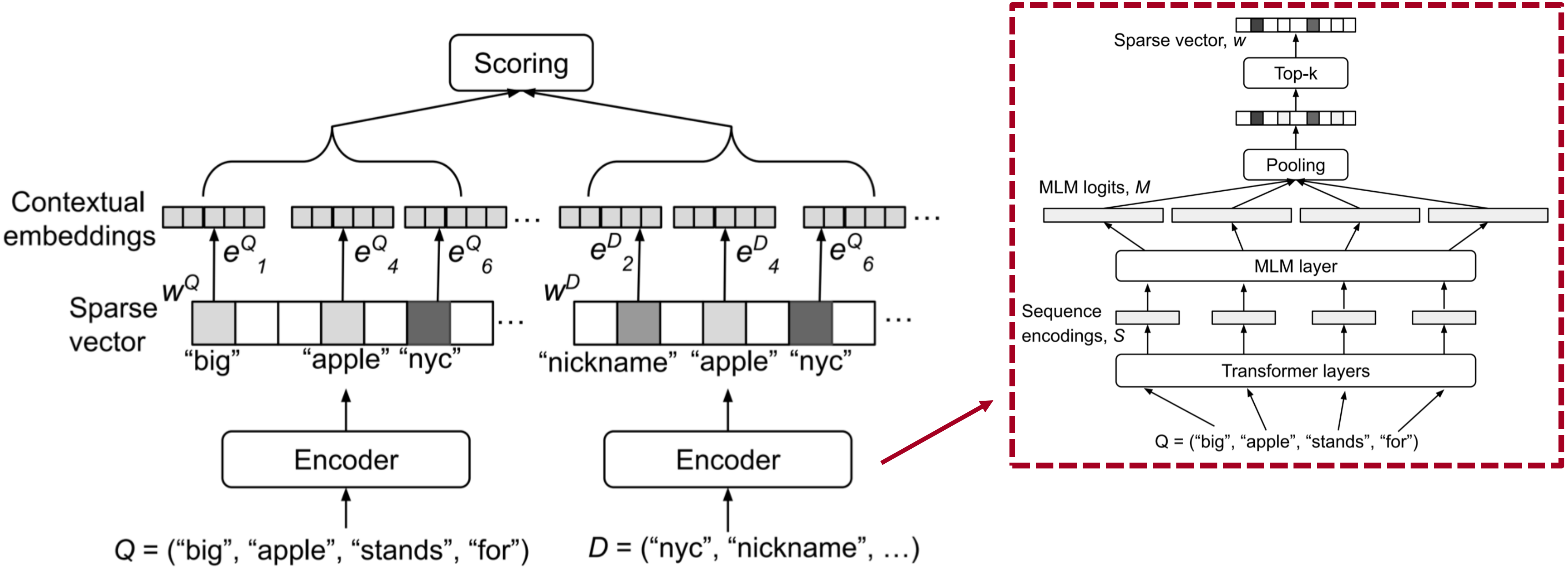
- **Sparse embedding**

  ✅ **Effective for exact match retrieval**

  ❌ **High-dimensional**

RA-LLM Learning: Vector Generation

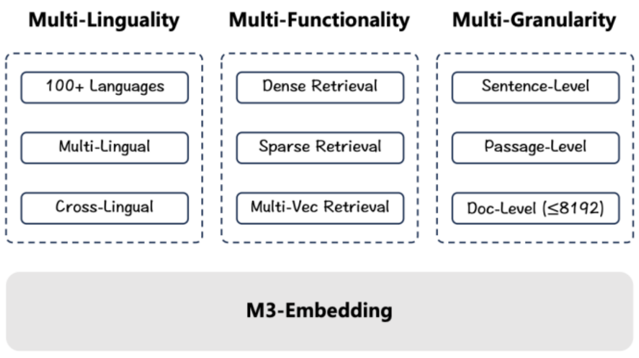
- **SparseEmbed**
  - Using top-k MLM logits and lightweight contextual attention pooling.



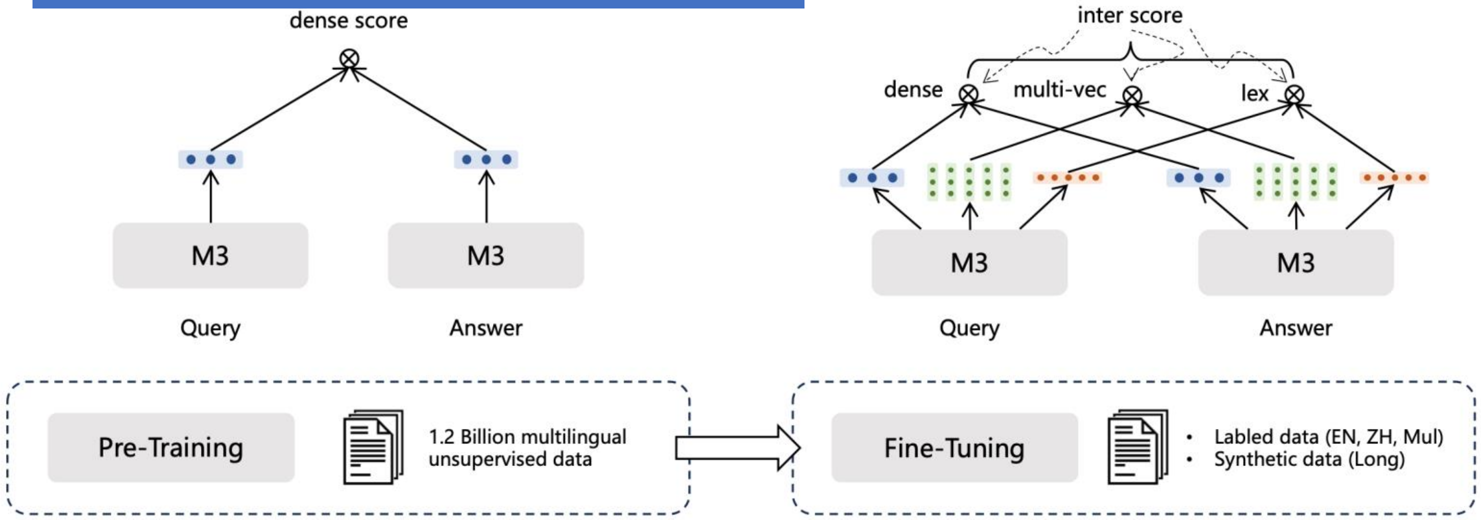
Kong et al. Sparseembed: Learning sparse lexical representations with contextual embeddings for retrieval, SIGIR, 2023.

RA-LLM Learning: Vector Generation

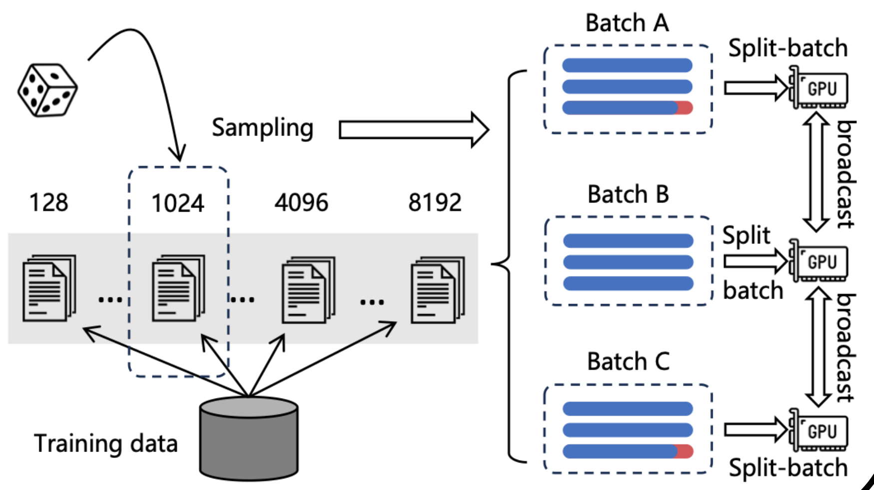
- **M3-Embedding**
  - Simultaneously accomplish the three common retrievals:
    - dense retrieval,
    - multi-vector retrieval,
    - and sparse retrieval.



Multi-stage Training



Efficient Batching



Chen et al. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2024.

Part 3: RA-LLM Data Management

Website of this tutorial

- Vector Generation
- **Indexing**
- Query Processing
- RAG-Oriented VDB Pipeline

Part 3: RA-LLM Data Management

Website of this tutorial

- Vector Generation
- **Indexing**
  - **partitioning**
- Query Processing
- RAG-Oriented VDB Pipeline

RA-LLM Learning: Indexing

- Vector indexes **speed up** queries by **minimizing the number of comparisons**. This is achieved by **partitioning** data so that only a small subset is compared.

  - **Tables** divide data into buckets containing similar vectors.
  - **Trees** are a nesting of tables.
  - **Graphs** connect similar vectors with virtual edges that can then be traversed.

RA-LLM Learning: Indexing

- **SPANN**
  - Learning-based **hash indexing (table)** for fast search.
  - Vectors X are hashed using $h(x)$, assigned to N posting lists $\{X_1, \dots, X_N\}$.

Balanced Posting List Sizes

$$\min_{\mathbf{H},\mathbf{C}} ||\mathbf{X} - \mathbf{HC}||_F^2 + \lambda \sum_{i=1}^{N} (\sum_{l=1}^{|\mathbf{X}|} h_{li} - |\mathbf{X}|/N)^2, \ \text{s.t.} \sum_{i=1}^{N} h_{li} = 1.$$

(Clustering loss) (Balance penalty)

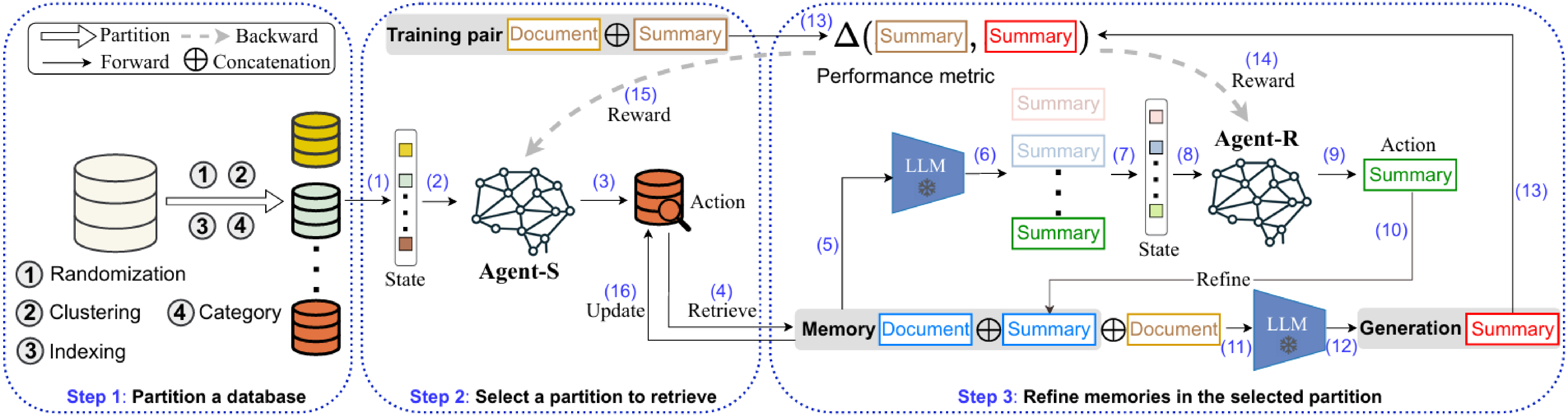
Adaptive Hashing

$$\mathbf{x} \in \mathbf{X}_{ij} \iff \text{Dist}(\mathbf{x}, \mathbf{c}_{ij}) \le (1 + \epsilon_1) \times \text{Dist}(\mathbf{x}, \mathbf{c}_{i1}),$$
$$\text{Dist}(\mathbf{x}, \mathbf{c}_{i1}) \le \text{Dist}(\mathbf{x}, \mathbf{c}_{i2}) \le \cdots \le \text{Dist}(\mathbf{x}, \mathbf{c}_{iK})$$

Chen et al. SPANN: Highly-efficient Billion-scale Approximate Nearest Neighbor Search, NeurIPS 2021

RA-LLM Learning: Indexing

- **M-RAG**
  - **Data partitioning with RL agents to optimize data selection and refinement.**



Wang, et al., 2024, M-RAG: Reinforcing Large Language Model Performance through Retrieval-Augmented Generation with Multiple Partitions.

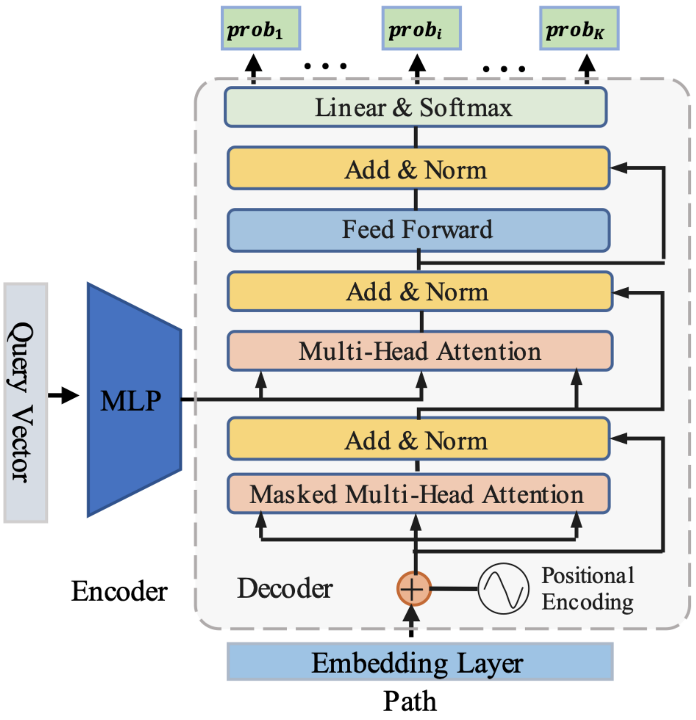
RA-LLM Learning: Indexing

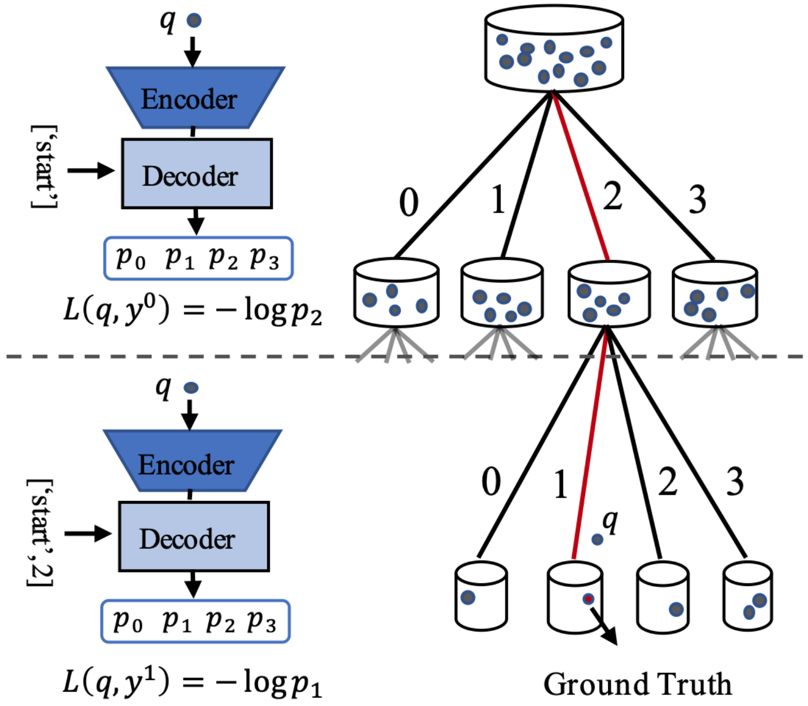
- Vector indexes **speed up** queries by **minimizing the number of comparisons**. This is achieved by **partitioning** data so that only a small subset is compared.

  - **Tables** divide data into buckets containing similar vectors.
  - **Trees** are a nesting of tables.
  - **Graphs** connect similar vectors with virtual edges that can then be traversed.

RA-LLM Learning: Indexing

- **BATLearn**
  - **Utilizes a learnable balanced K-ary tree with sequence-to-sequence routing**



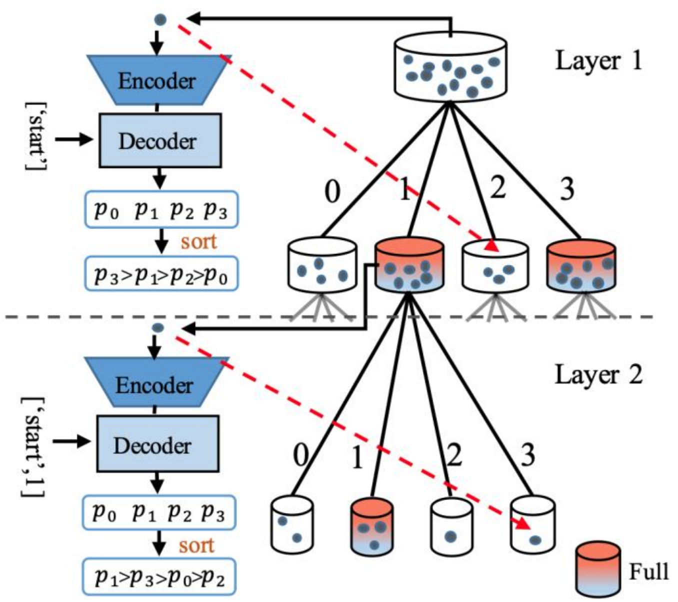


Li, et al., Learning Balanced Tree Indexes for Large-Scale Vector Retrieval, KDD 2023.

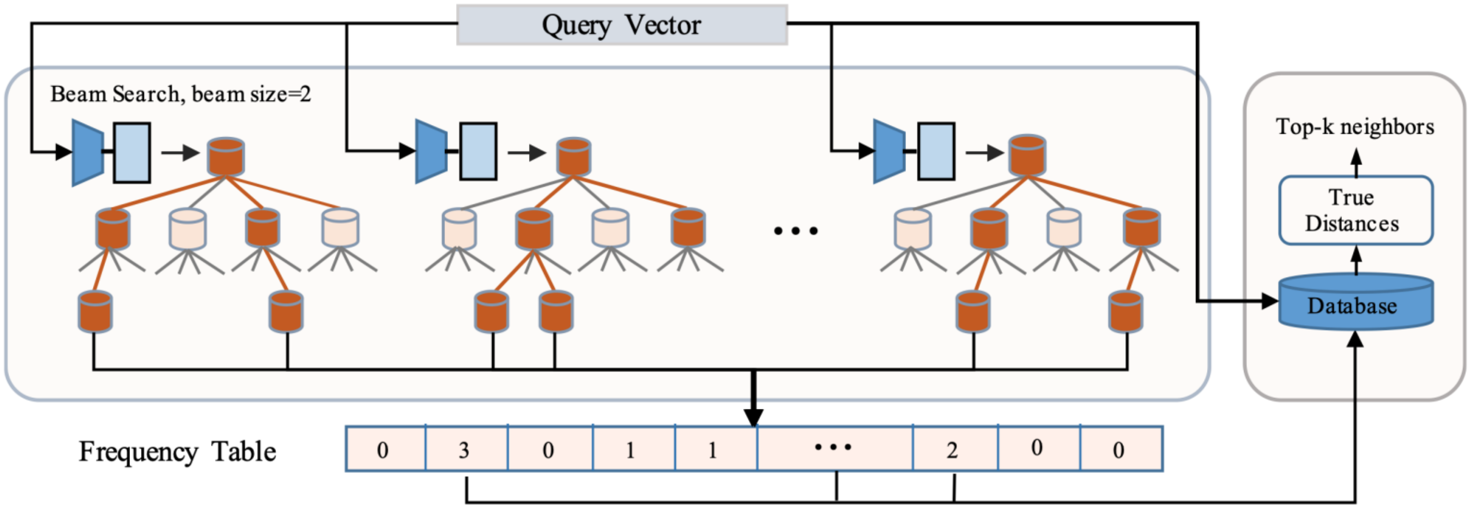
RA-LLM Learning: Indexing

- **BATLearn**

Learning



Inference



Li, et al., Learning Balanced Tree Indexes for Large-Scale Vector Retrieval, KDD 2023.

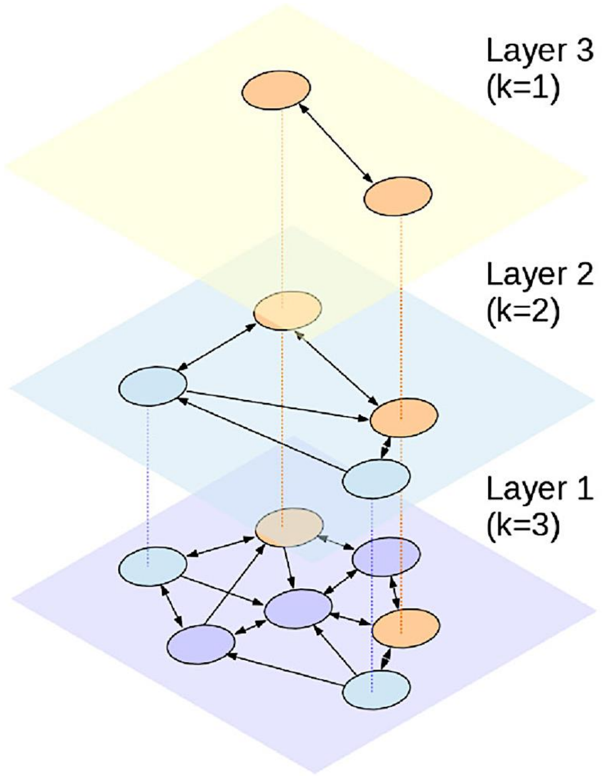
RA-LLM Learning: Indexing

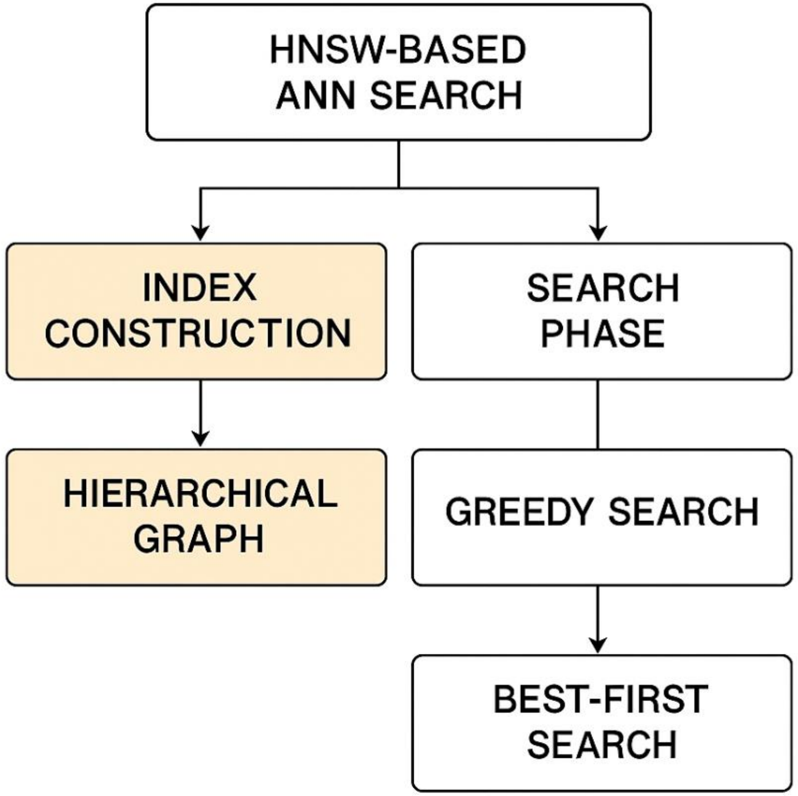
- Vector indexes **speed up** queries by **minimizing the number of comparisons**. This is achieved by **partitioning** data so that only a small subset is compared.

  - **Tables** divide data into buckets containing similar vectors.
  - **Trees** are a nesting of tables.
  - **Graphs** connect similar vectors with virtual edges that can then be traversed.

RA-LLM Learning: Indexing

- **Hierarchical NSW**
  - **A multi-layer proximity graph with probabilistic layer assignment**

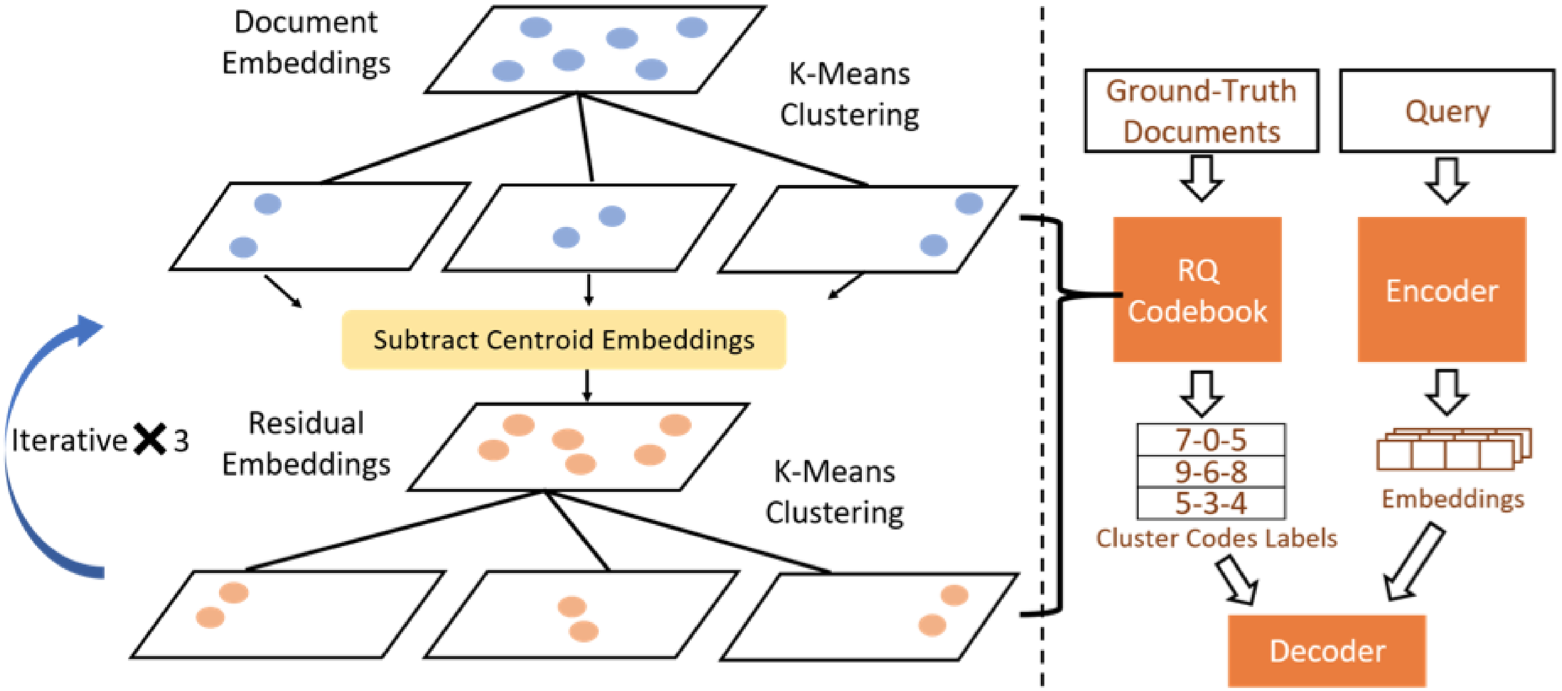




Malkov et al., Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs, TPAMI, 2020

RA-LLM Learning: Indexing

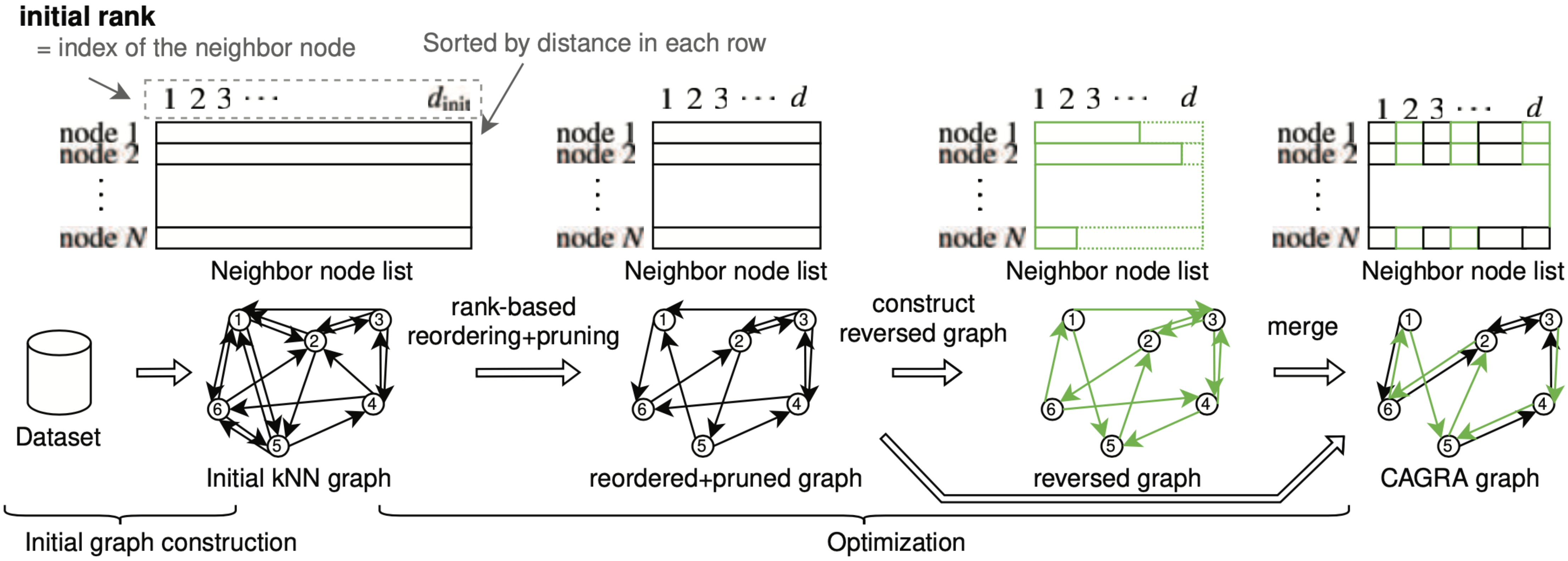
- **MEVI**
  - **Residual quantization with hierarchical clustering**



Zhang et al., Model-enhanced Vector Index, NeurIPS, 2023.

RA-LLM Learning: Indexing

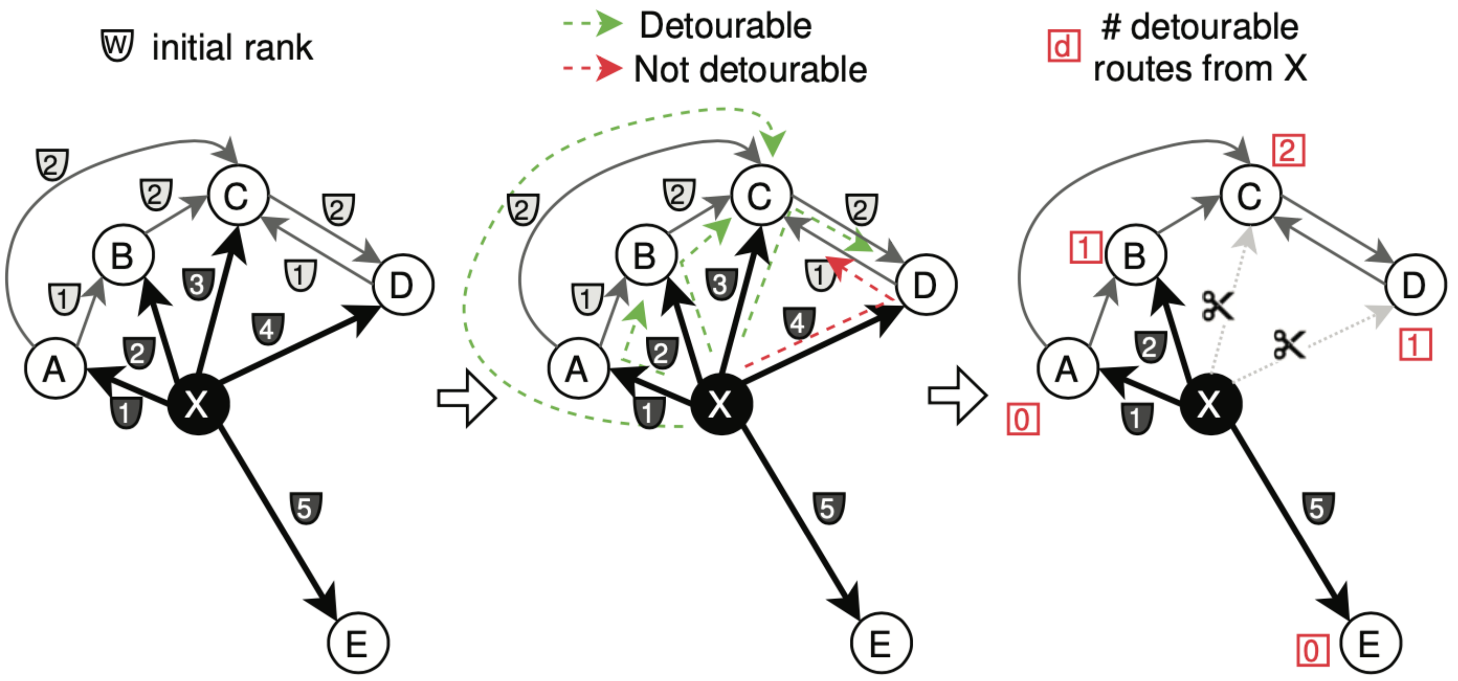
- CAGRA
  - Rank-based edge reordering and reversed edge optimization



Ootomo et al., CAGRA: Highly Parallel Graph Construction and Approximate Nearest Neighbor Search for GPUs, ICDE, 2024.

RA-LLM Learning: Indexing

- **CAGRA**



$$(e_{X \to Z}, e_{Z \to Y}) \text{ s.t. } \max(w_{X \to Z}, w_{Z \to Y}) < w_{X \to Y},$$

Ootomo et al., CAGRA: Highly Parallel Graph Construction and Approximate Nearest Neighbor Search for GPUs, ICDE, 2024.

Part 3: RA-LLM Data Management

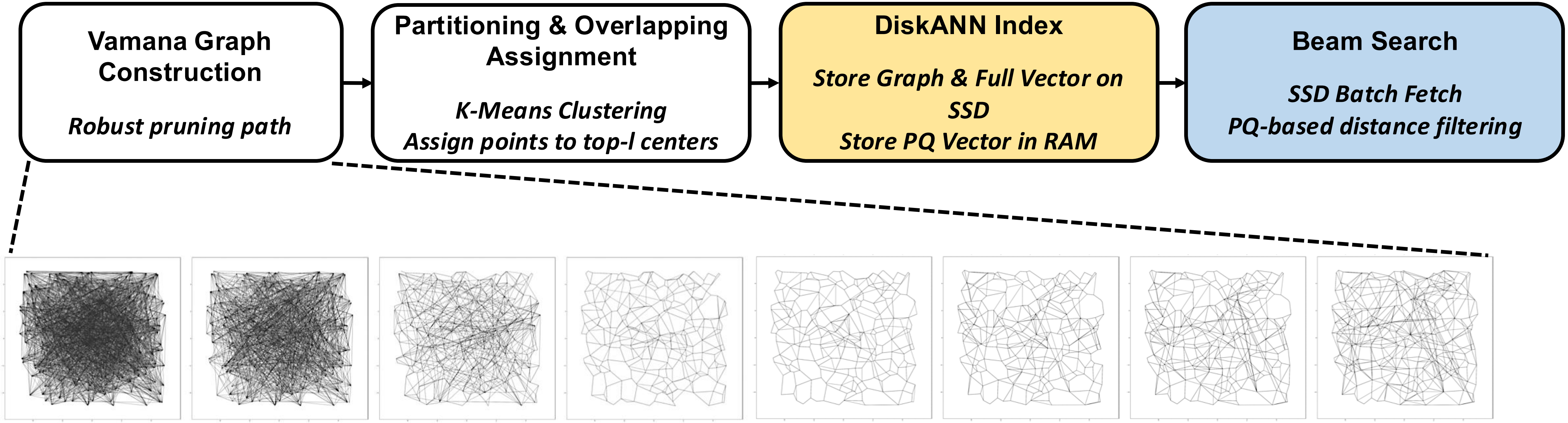
Website

Website of this tutorial

- Vector Generation
- **Indexing**
  - **Storage**
- Query Processing
- RAG-Oriented VDB Pipeline

RA-LLM Learning: Storage

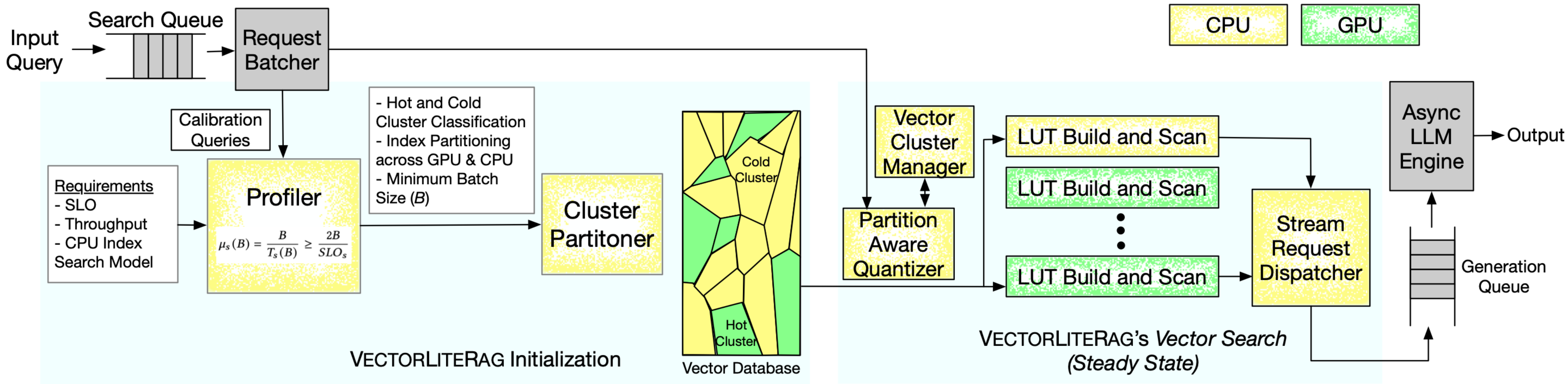
- **DiskANN**
  - Uses a Disk-Memory mixed structure to accelerate query search.



Subramanya et al., DiskANN: Fast Accurate Billion-point Nearest Neighbor Search on a Single Node, NeurIPS, 2019.

RA-LLM Learning: Storage

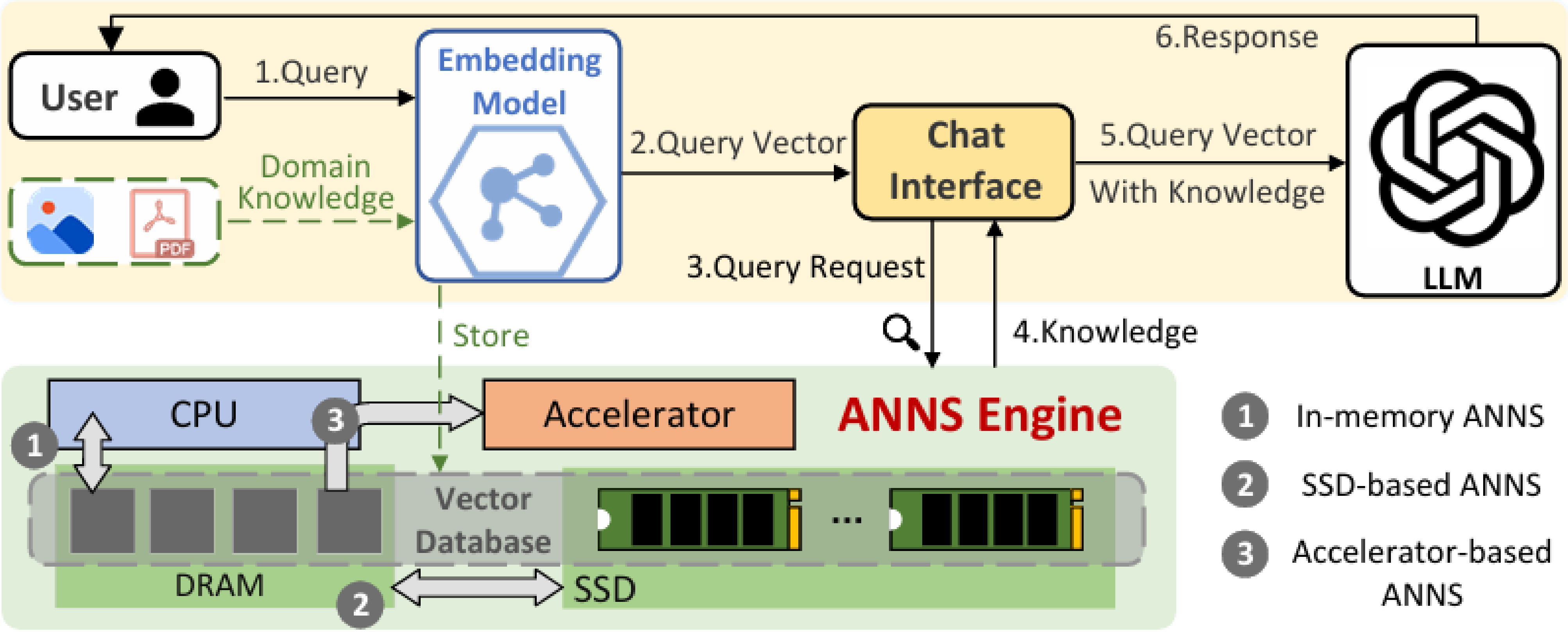
- **VectorLiteRAG**
  - **Uses a GPU-CPU mixed storage structure to accelerate query search.**



Kim, et al., An Adaptive Vector Index Partitioning Scheme for Low-Latency RAG Pipeline, ICDE, 2024.

RA-LLM Learning: Storage

- **Fusion-ANNS**
  - **A GPU-CPU-SSD optimized structure to accelerate query search.**



Tian, et al., Towards High-throughput and Low-latency Billion-scale Vector Search via CPU/GPU Collaborative Filtering and Re-ranking, USENIX, 2025.

Part 3: RA-LLM Data Management

Website

Website of this tutorial

- Vector Generation
- Indexing
- **Query Processing**
- RAG-Oriented VDB Pipeline

Part 3: RA-LLM Data Management

Website

Website of this tutorial

- Vector Generation
- Indexing
- **Query Processing**
  - **Similarity Calculation**
- RAG-Oriented VDB Pipeline

RA-LLM Learning: Query Processing

- **General Similarity Scores**
  - **The query processor mainly deals with how to specify the query criteria and how to execute search queries.**

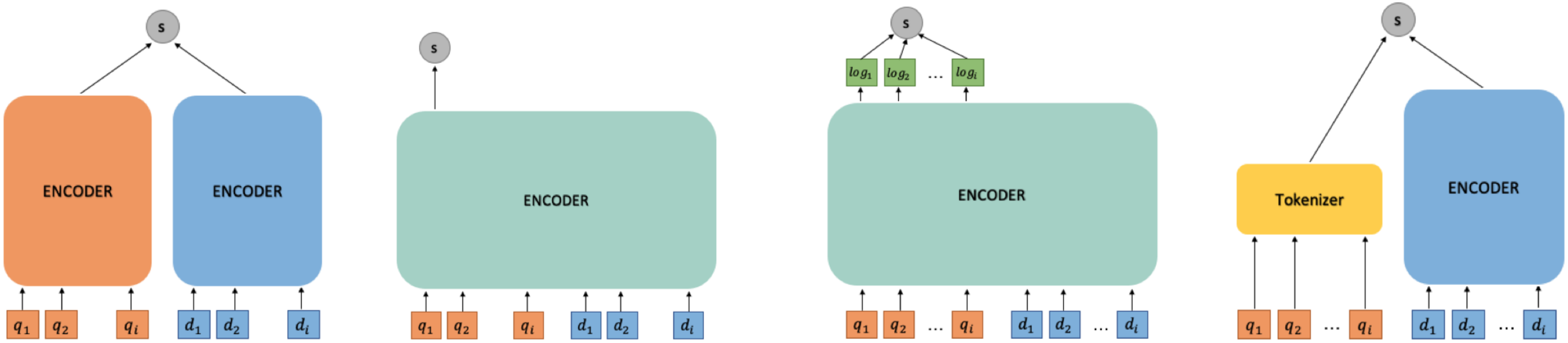
Table 1 Common similarity scores

Type	Score	Metric	Complexity	Range
Sim	Inner Prod	✗	$O(D)$	$\mathbb{R}$
	Cosine	✗	$O(D)$	$[-1, 1]$
Dist	Minkowski	✓	$O(D)$	$\mathbb{R}^+$
	Hamming	✓	$O(D)$	$\mathbb{N}$

Pan et al. Survey of vector database management system, VLDBJ, 2024.

RA-LLM Learning: Query Processing

- **Existing Query Processing Paradigm**

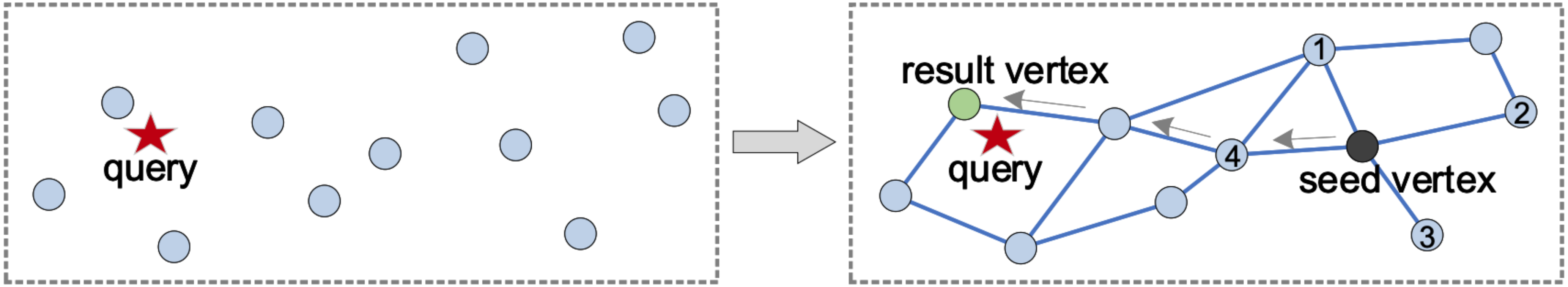


	Representation-based	**Deep LMs re-rankers**	**Deep query likelihood models**	**TILDE**
Effectiveness	★★	★★	★★★	★★
Efficiency	★★	★	★	★★★

Zhuang et al. TILDE: Term Independent Likelihood moDEl for Passage Re-ranking, SIGIR, 2021.

RA-LLM Learning: Query Processing

- **Query Processing with Graph-Based ANN Search.**
  - Navigates the graph index from a seed vertex until the closest result vertex is identified.



(a) Original dataset (b) Graph index

Wang et al., A Comprehensive Survey and Experimental Comparison of Graph-Based Approximate Nearest Neighbor Search, VLDB 2021.

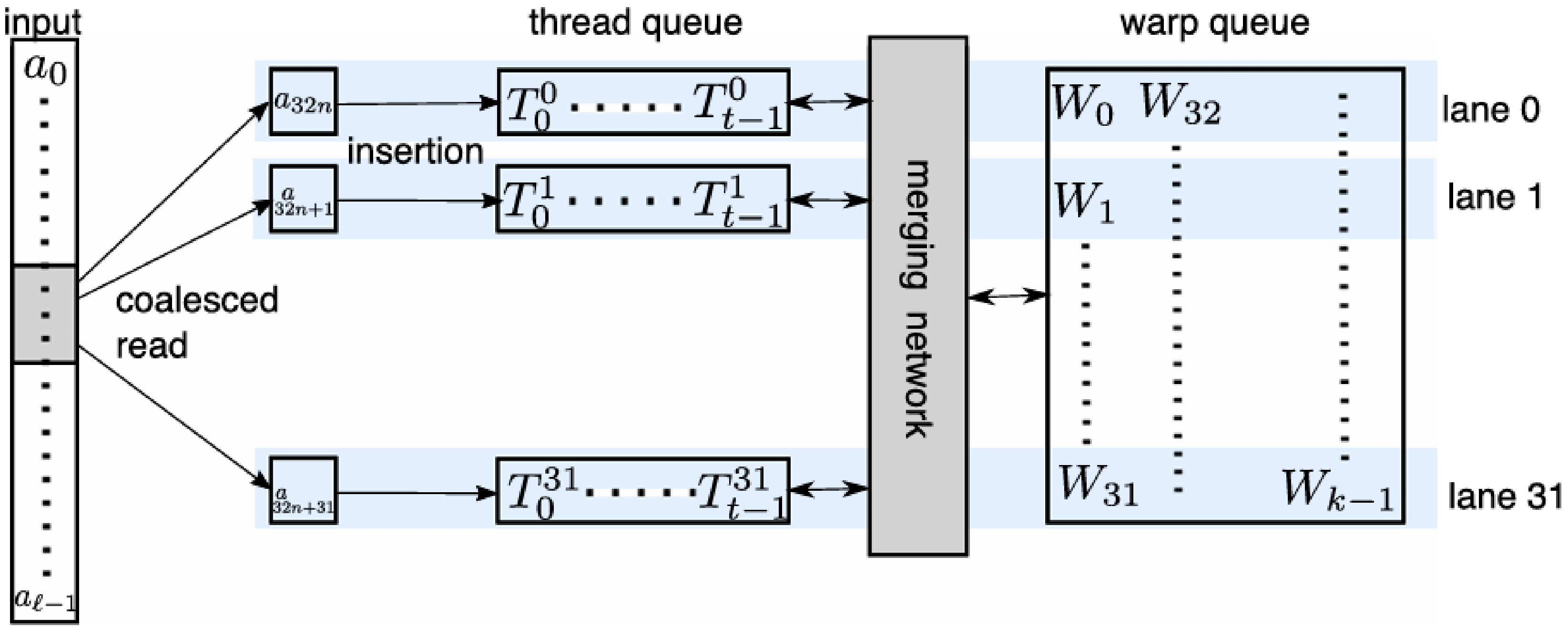
Part 3: RA-LLM Data Management

Website of this tutorial

- Vector Generation
- Indexing
- Query Processing
  - Acceleration
- RAG-Oriented VDB Pipeline

RA-LLM Learning: Query Processing

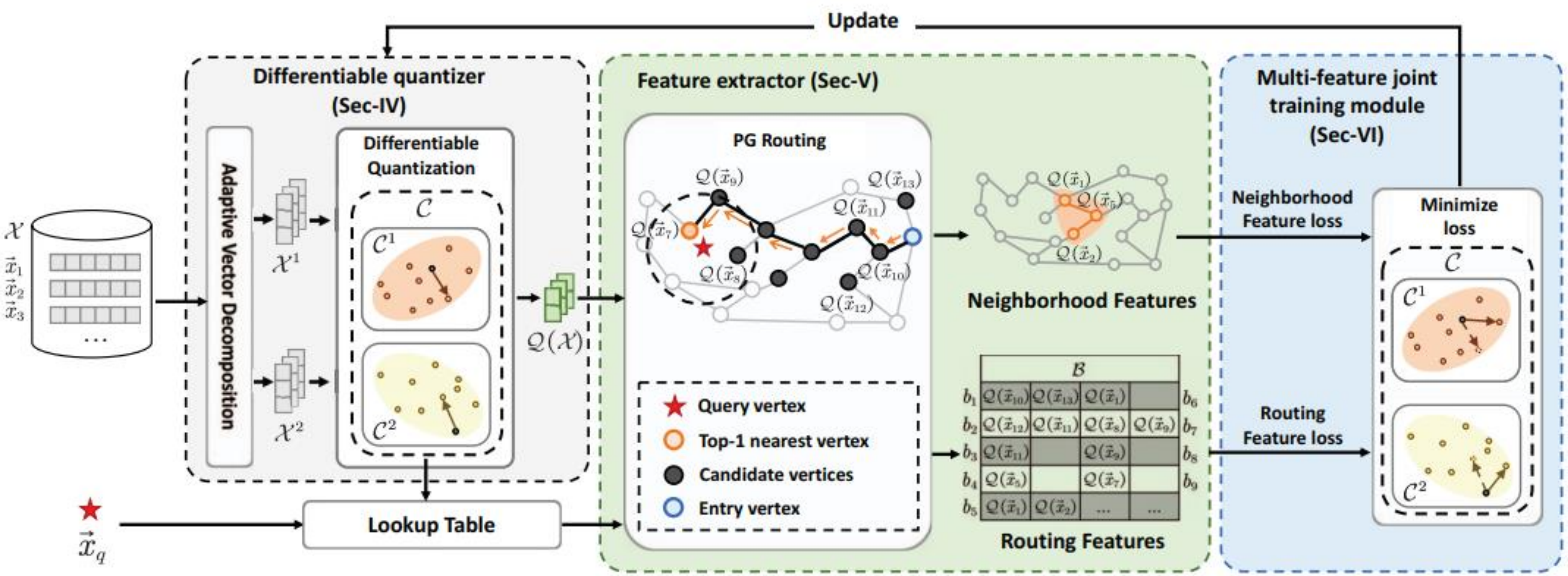
- **WarpSelect**
  - **A GPU-optimized high-performance top-k selection algorithm.**



Johnson et al., Billion-Scale Similarity Search with GPUs, TBD, 2021.

RA-LLM Learning: Query Processing

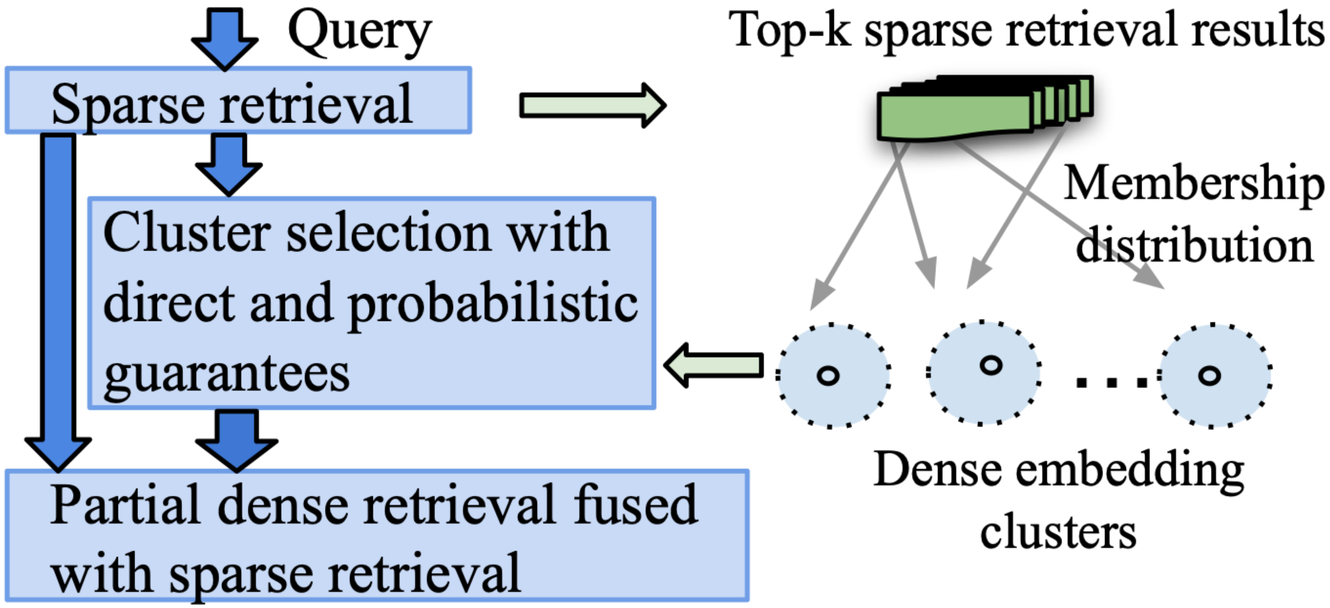
- **Routing-guided learned Product Quantization (RPQ)**
  - **Integrates routing-guided learned product quantization with adaptive vector decomposition.**



Yue et al. Routing-Guided Learned Product Quantization for Graph-Based Approximate Nearest Neighbor Search, ICDE, 2024.

RA-LLM Learning: Query Processing

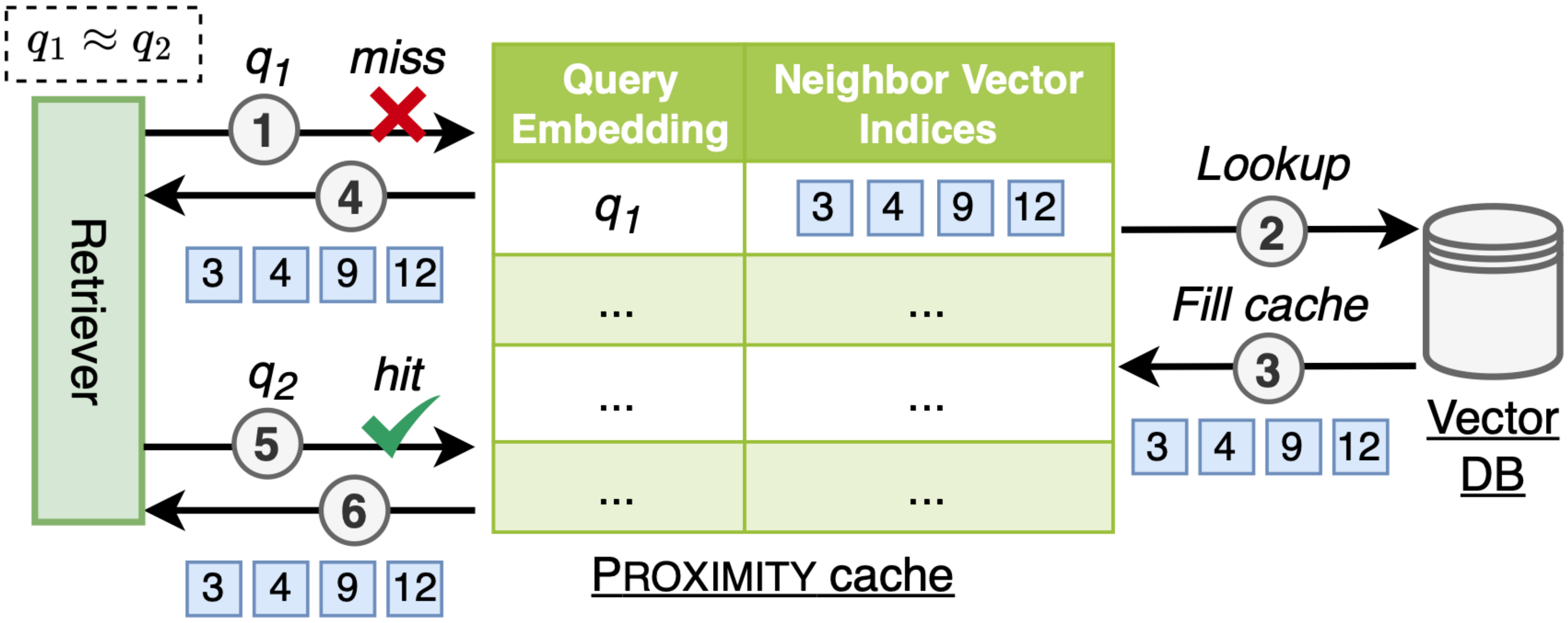
- **CDFS**
  - Combines sparse retrieval with cluster-guided partial dense retrieval



Yang et al. Cluster-based Partial Dense Retrieval Fused with Sparse Text Retrieval, SIGIR, 2024.

RA-LLM Learning: Query Processing

- **Proximity**
  - **Reusing similar query results without repeated database lookups.**



Bergman, et al., Leveraging Approximate Caching for Faster Retrieval-Augmented Generation, EuroMLSys 2025.

Part 3: RA-LLM Data Management

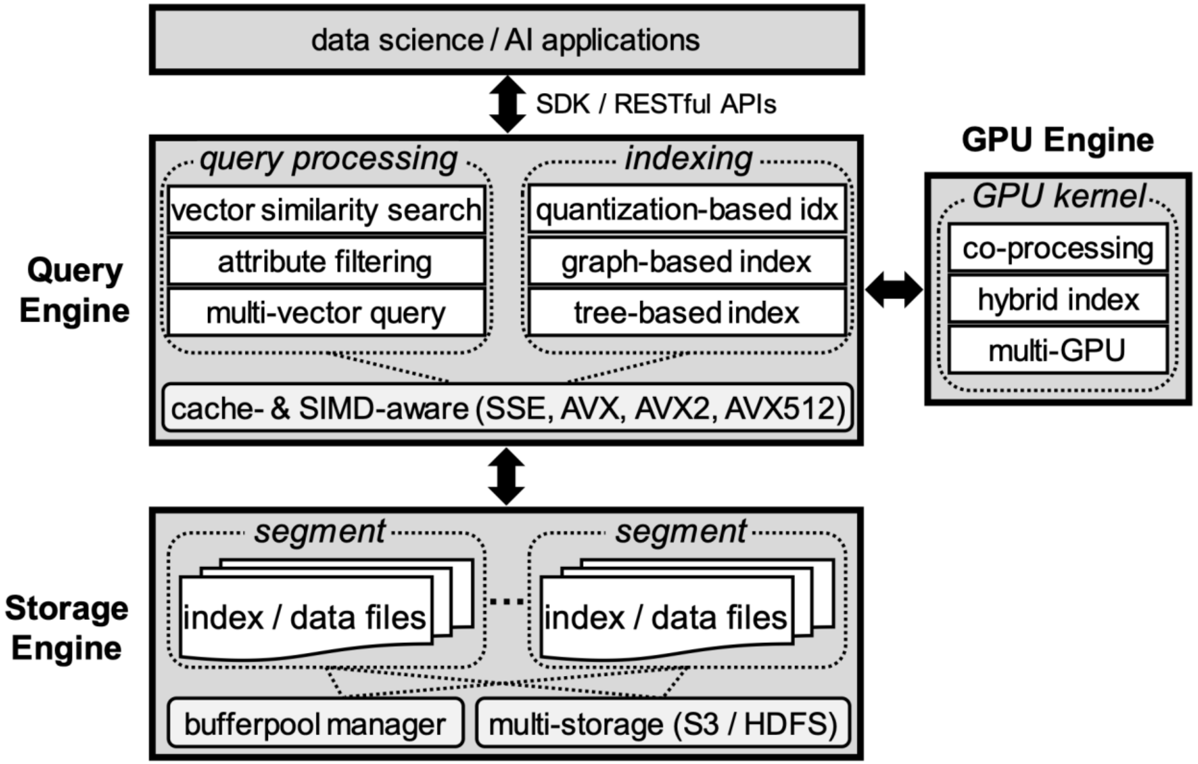
Website

Website of this tutorial

- Vector Generation
- Indexing
- Query Processing
- **RAG-Oriented VDB Pipeline**

RA-LLM Learning: RAG-Oriented VDB Pipeline

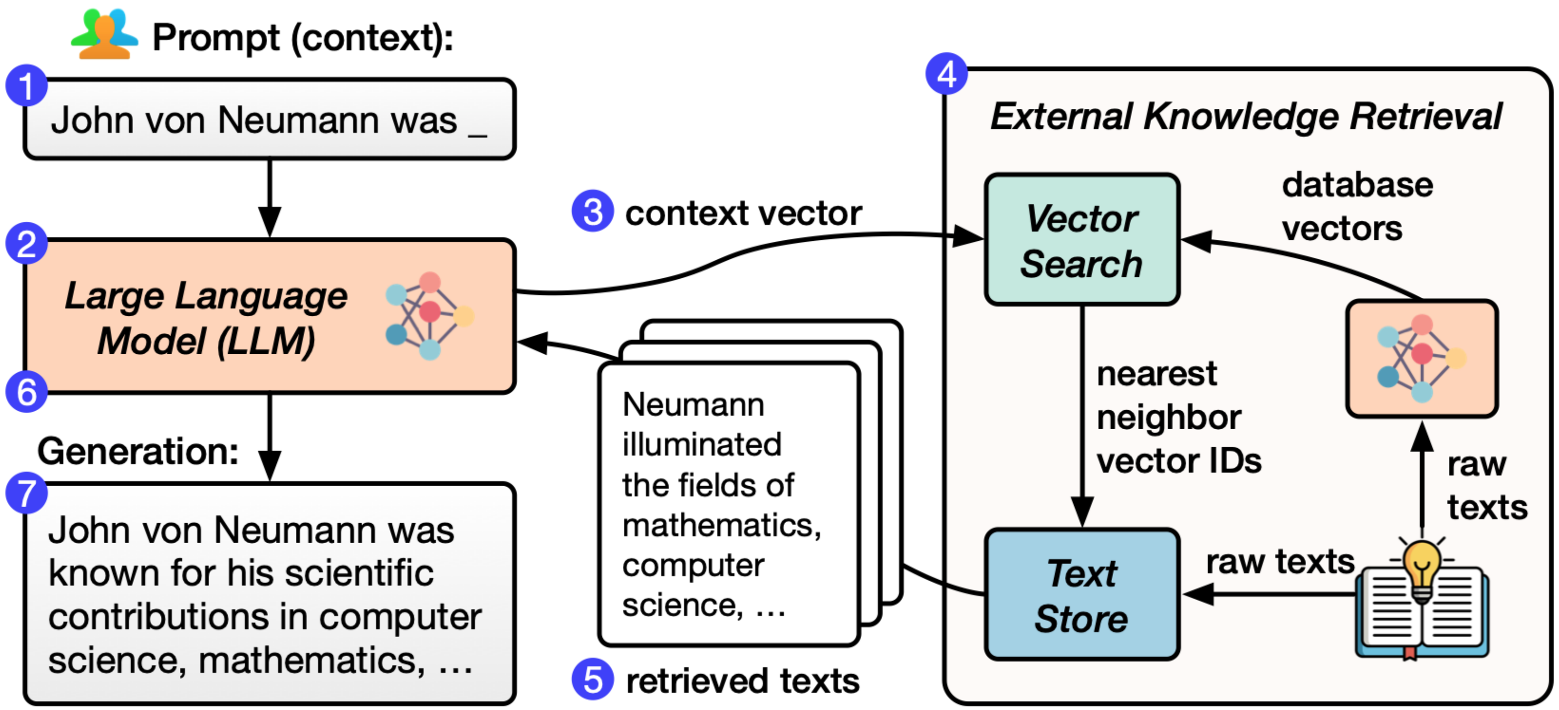
- **Milvus**
  - Leverages a hybrid GPU-CPU engine with SIMD-optimized query processing and multi-storage for scalability.



Wang et al. Milvus: A Purpose-Built Vector Data Management System, SIGMOD, 2021.

RA-LLM Learning: RAG-Oriented VDB Pipeline

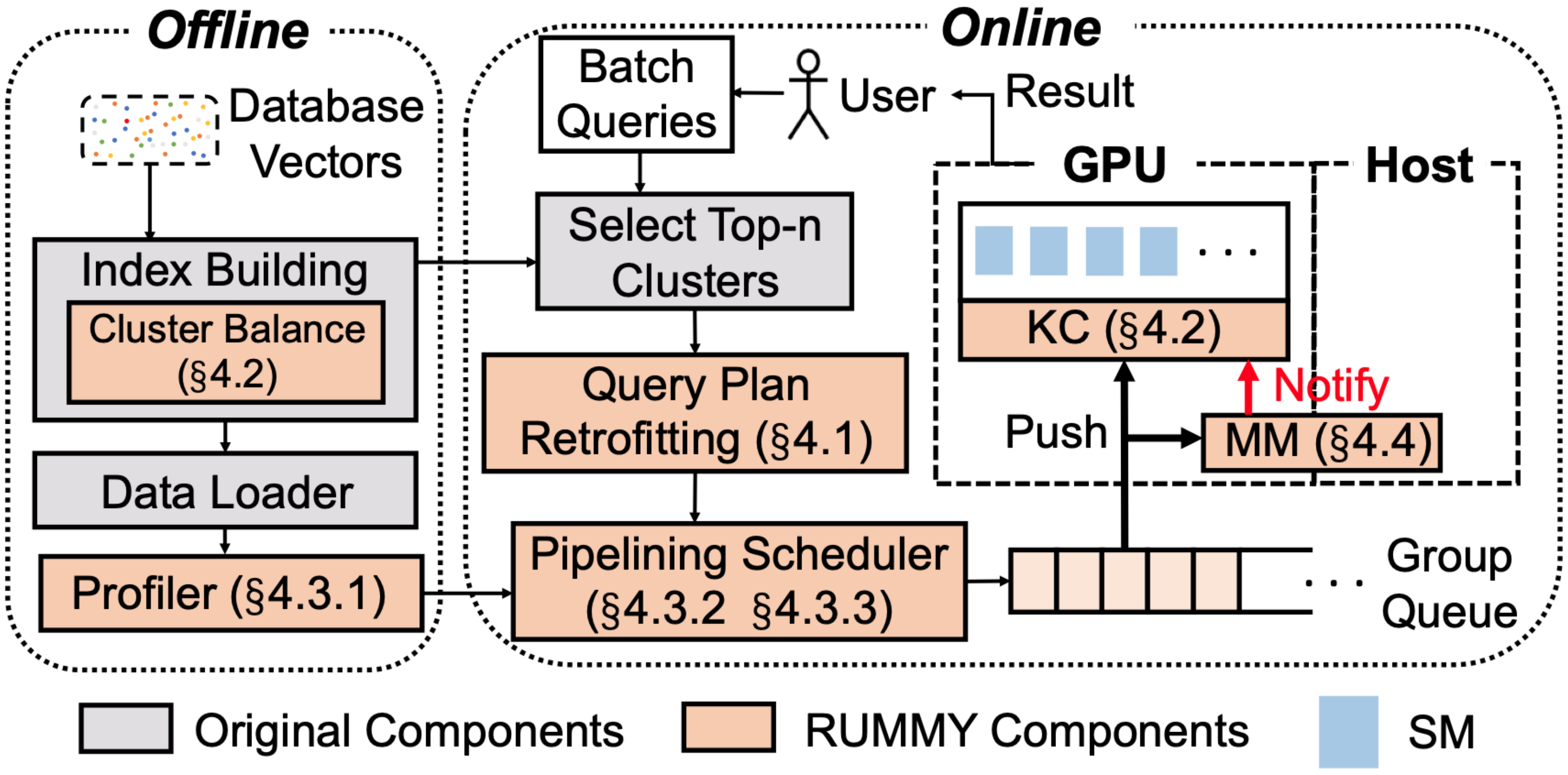
- **Chameleon**
  - **FPGA-based near-memory accelerators for vector search.**
  - **Disaggregated architecture for LLM inference.**



Jiang et al. Chameleon: a Heterogeneous and Disaggregated Accelerator System for Retrieval-Augmented Language Models, VLDB, 2024.

RA-LLM Learning: RAG-Oriented VDB Pipeline

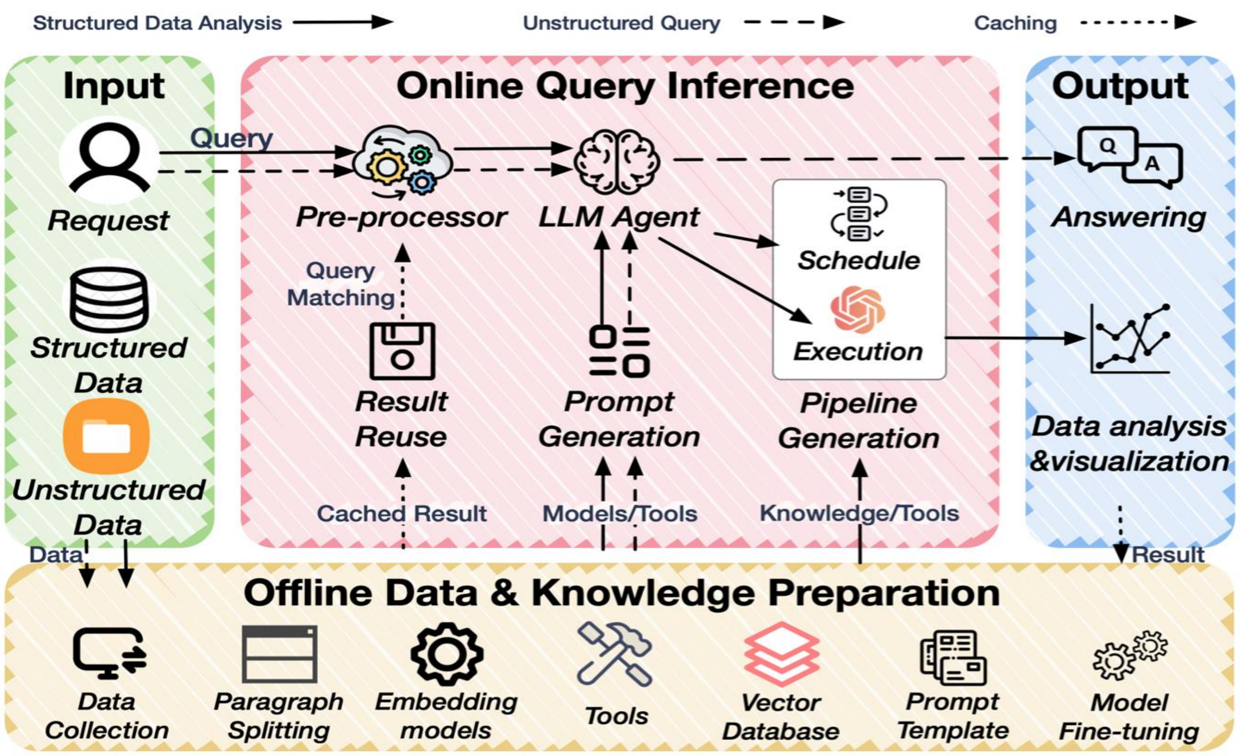
- **Rummy**
  - **Using cluster-balanced indexing, query plan retrofitting, and pipelined scheduling.**



Zhang et al., Fast Vector Query Processing for Large Datasets Beyond GPU Memory with Reordered Pipelining, USENIX 2024.

RA-LLM Learning: RAG-Oriented VDB Pipeline

- **Chat2Data**
  - **A multi-stage pipeline with adaptive query preprocessing, LLM-driven pipeline generation, and caching.**



Zhao, et al., Chat2Data: An Interactive Data Analysis System with RAG, Vector Databases and LLMs, VLDB, 2024.

Tutorial Outline

- **Part 1: Introduction** of Retrieval Augmented Large Language Models (RA-LLMs) (Dr. Yujuan Ding)
- **Part 2: Architecture** of RA-LLMs and **Main Modules** (Dr. Yujuan Ding)
- **Part 3: Data Management** for RA-LLMs (Pangjing Wu)
- **Part 4: Learning Approach of RA-LLMs (Liangbo Ning)**
- **Part 4: Applications** of RA-LLMs (Shijie Wang)
- **Part 5: Challenges and Future Directions** of RA-LLMs (Liangbo Ning)

Website of this tutorial
Check out the slides and more information!