Retrieve-Augmented Large Language model

陈云

上海财经大学

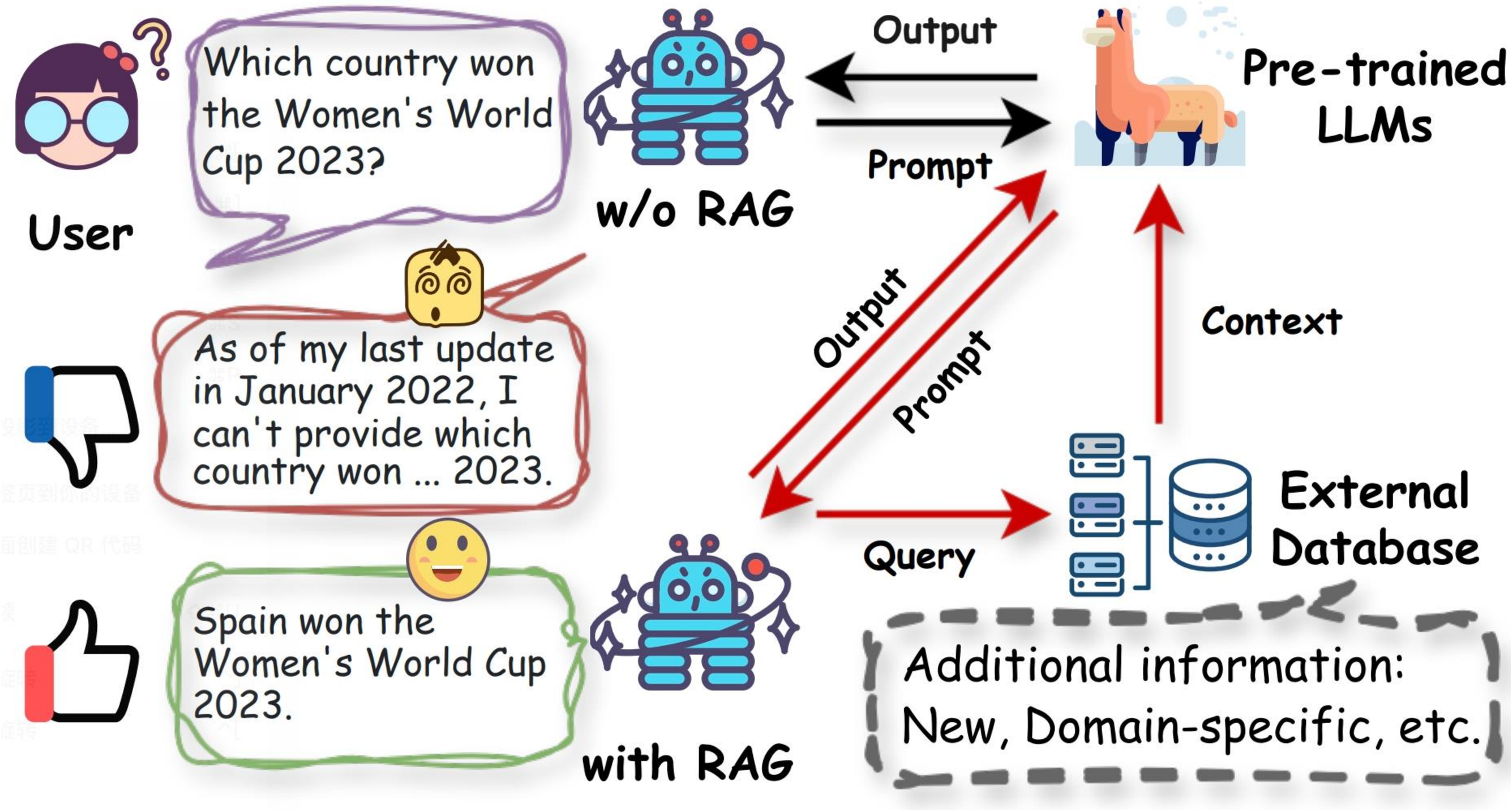
信息管理与工程学院计算机系

Content

- Introduction
- Retrieval-augmented LLM
- Framework of RA-LLM

Introduction

- Retrieve augmented large language model (RA-LLM)
  - Generation with retrieved information



Information Retrieval

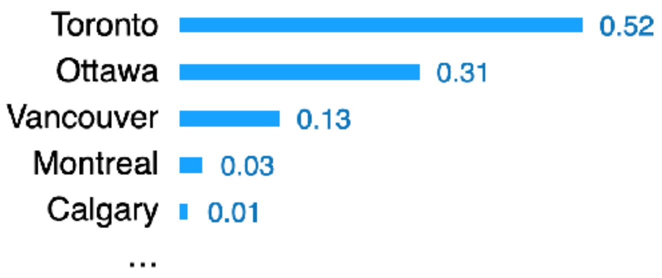
- Information retrieval (IR)
  - Locate and retrieve information that is relevant to a user's query
  - Retrieval and ranking
- Sparse retrieval: TF-IDF, BM25
- Dense retrieval
  - Sentence-bert
  - DPR, ColBERT
  - OpenAI Embeddings

Retrieval-augmented Language Models

- It is a **language model**

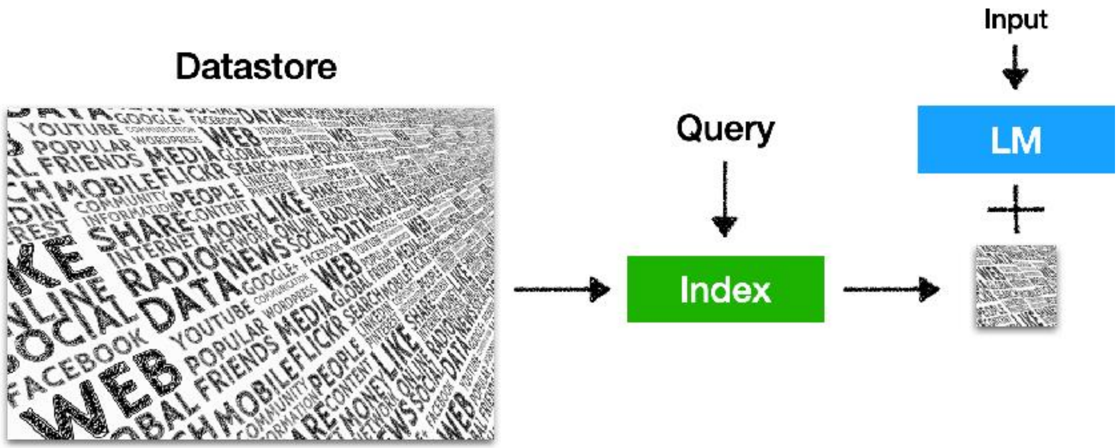
$$P(x_n | x_1, x_2, \cdots, x_{n-1})$$

The capital city of Ontario is _



(can be broadly extended to masked language models or encoder-decoder models)

- It retrieves from an **external datastore** (at least during inference time)



(Also referred to semi-parametric and non-parametric models)

Retrieval for knowledge-intensive NLP tasks

Representative tasks: open-domain QA, fact checking, entity linking, ..

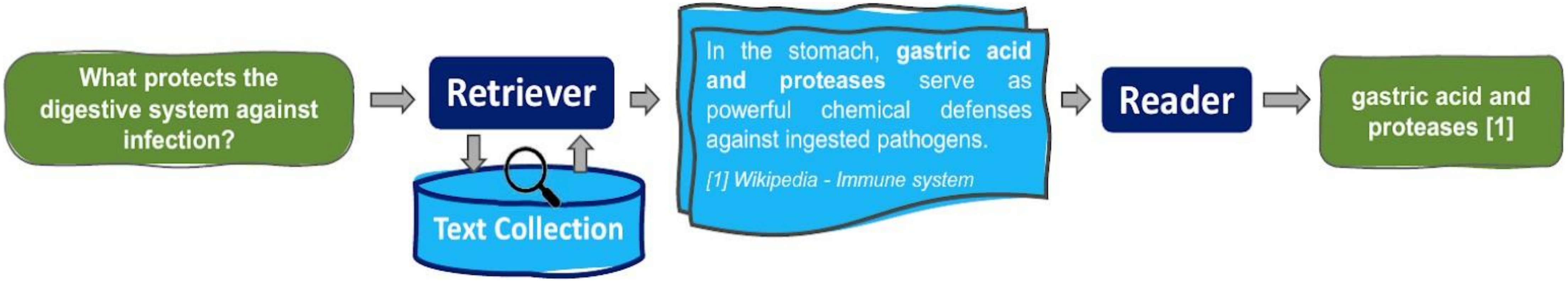


Image: http://ai.stanford.edu/blog/retrieval-based-NLP/

Drives a lot of research on better algorithms for **dense retrieval**, e.g., **DPR, ColBERT**, **Contriever**…

Why retrieval LMs?

Why RA-LLM?

- LLMs can't memorize all (long-tail) knowledge in their parameters

List 5 important papers authored by Geoffrey Hinton

ChatGPT

Geoffrey Hinton is a renowned computer scientist …
Here are five important papers authored by him:

1. "**Learning Internal Representations by Error Propagation**" (with **D. E. Rumelhart** and **R. J. Williams**) - This paper, published in **1986**, ..

2. "**Deep Boltzmann Machines**" (with **R. Salakhutdinov**) - Published in **2009**, ..

…

4. "**Deep Learning**" (with Y. Bengio and A. Courville) - Published as a book in **2016**,…

5. "**Attention Is All You Need**" (with V. Vaswani, N. Shazeer, et al.) - Published in **2017**, this paper introduced the Transformer model,…

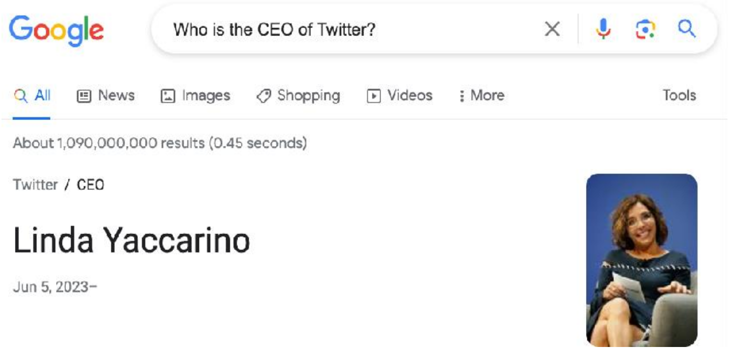
Why RA-LLM?

- LLMs' knowledge is easily outdated and hard to update

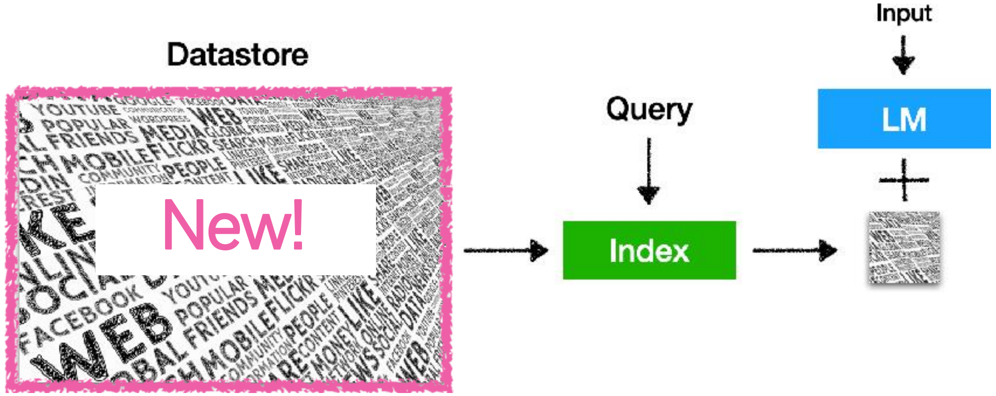
Who is the CEO of Twitter?

ChatGPT

As of my **knowledge cutoff in September 2021**, the CEO of Twitter is **Jack Dorsey….**

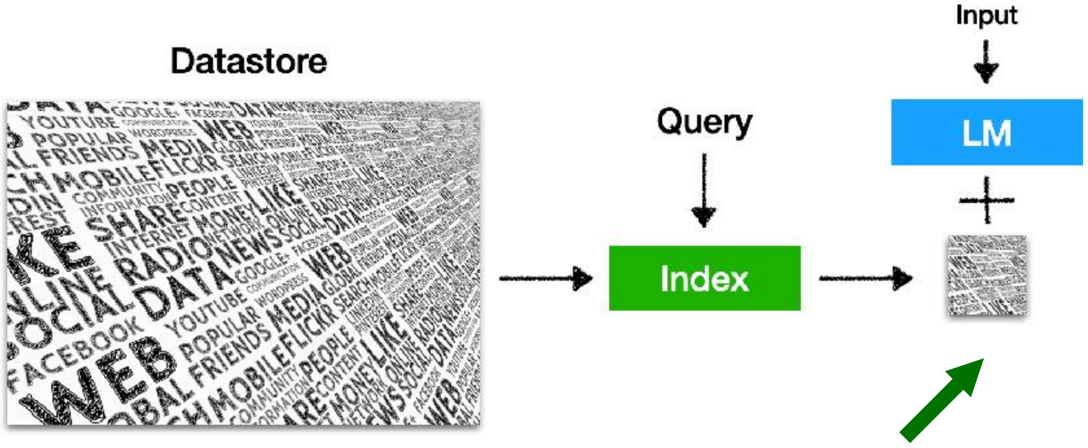


- Existing **knowledge editing** methods are still NOT scalable
- The datastore can be easily **updated** and **expanded** - even without retraining!



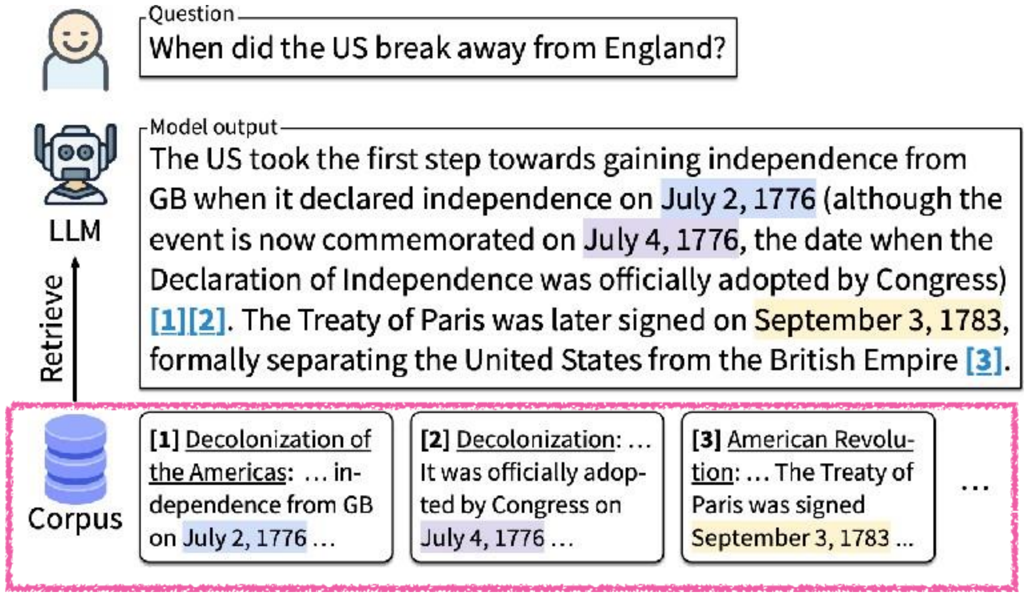
Why RA-LLM?

- LLMs' output is challenging to interpret and verify



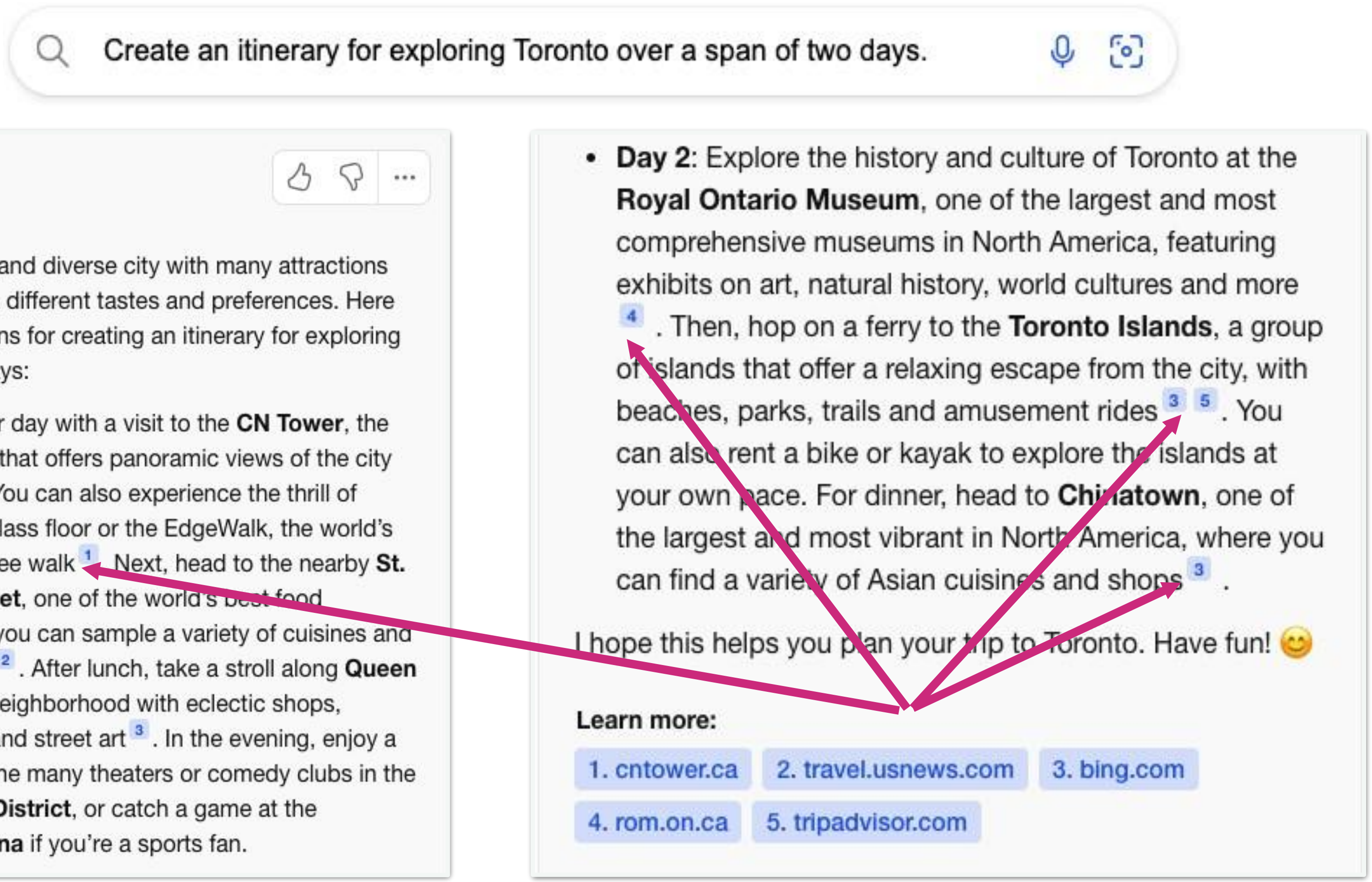
Can trace knowledge source from retrieval results - better interpretability & control

Generating text with citations



Why RA-LLM?

- LLMs' output is challenging to interpret and verify



Google Shares Drop $100 Billion

- An accompanying GIF prompts Bard with the question, "What new discoveries from the James Webb Space Telescope can I tell my 9-year-old about?“
- The chatbot responds with a few bullet points, including the claim that the telescope took the very first pictures of "exoplanets“, or planets outside of earth's solar system.
- But the James Webb Telescope didn't discover exoplanets.

TECHNOLOGY

Google shares drop $100 billion after its new AI chatbot makes a mistake

February 9, 2023 · 10:15 AM ET

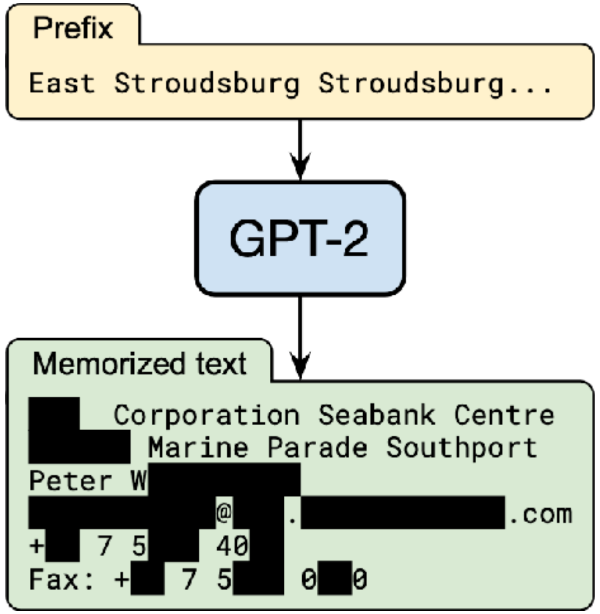
By Emily Olson

Shares for Google's parent company, Alphabet, dropped 9% Wednesday after its AI chatbot, Bard, gave an incorrect answer.

Dan Kitwood/Getty Images

Why RA-LLM?

- LLMs are shown to easily leak private training data

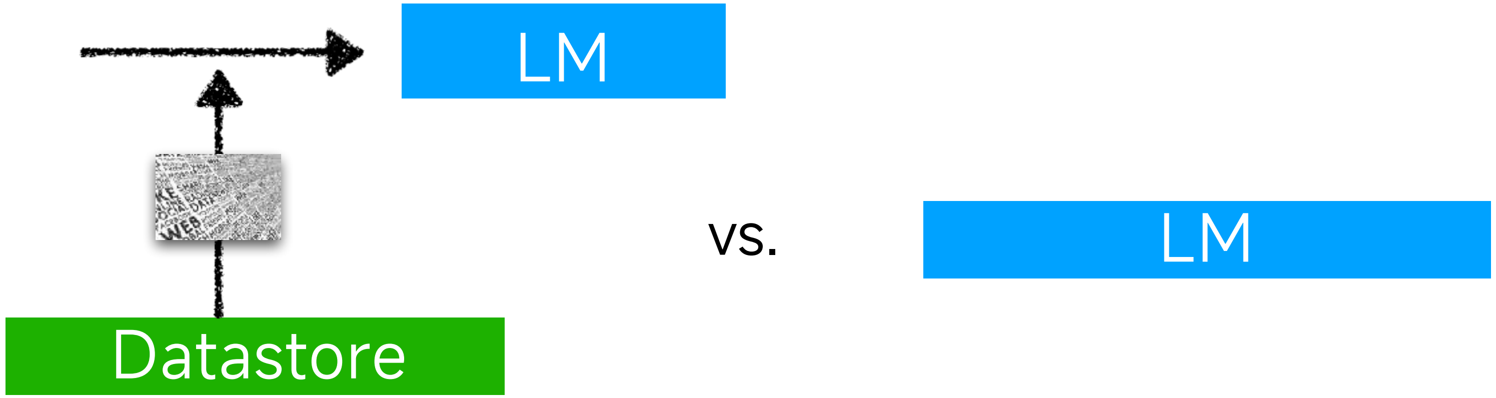


Individualization on private data by storing it in the datastore

Carlini et al. 2021. "Extracting Training Data from Large Language Models"

Why RA-LLM?

- LLMs are large and expensive to train and run



- Long-term goal
  - Can we possibly reduce the training and inference costs, and scale down the size of LLMs?
  - e.g., RETRO (Borgeaud et al., 2021): "obtains comparable performance to GPT-3 on the Pile, despite using 25x fewer parameters"

Why RA-LLM?

- LLMs are large and expensive to train and run

- It seems scaling larger and larger models is the main way of improving the performance
- But with a tremendous increase in training energy cost
  - Additional computations at training and inference time
  - Increased memorization of the training data

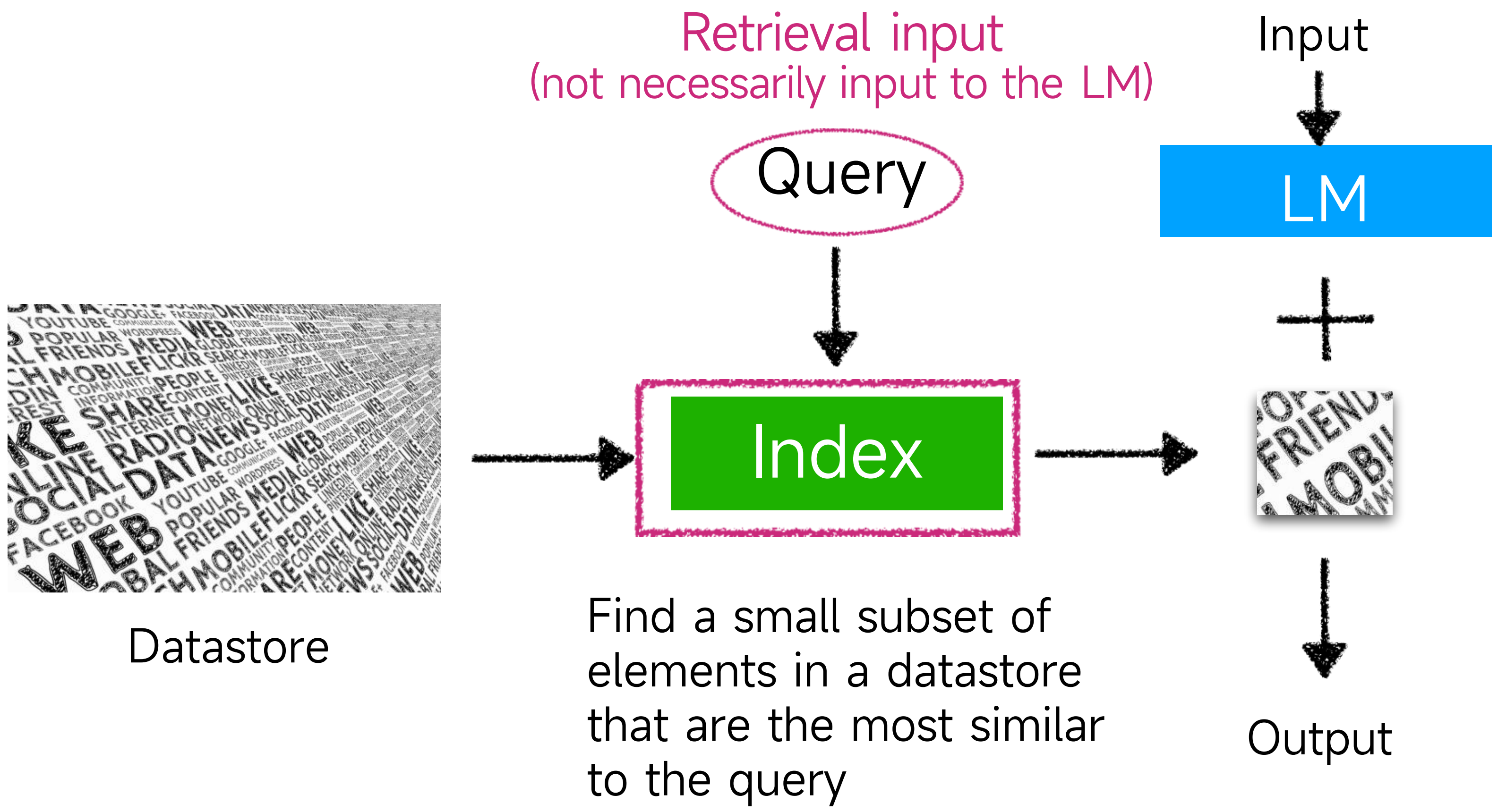
Can we separate language information from world knowledge information?

Why RA-LLM?

- Tackling Inefficiency
  - Retrieval-based models can be much smaller and faster
- Tackling Opaqueness
  - When the model produces an answer, we can read the sources it retrieved and judge their relevance and credibility for ourselves.
- Tackling Static Knowledge
  - The retrieval knowledge store can be efficiently updated or expanded by modifying the text corpus
  - Real-time/Dynamic data

> - When you fine-tune a model, it's like studying for an exam one week away.
> - When you insert knowledge into the prompt (e.g., via retrieval), it's like taking an exam with open notes.

Retrieval-augmented Language Model



Similarity Score

Goal: find a small subset of elements in a datastore that are the most similar to the query

A similarity score between two pieces of text

Example $$\mathrm{sim}(i, j) = \mathrm{tf}_{i,j} \times \log \frac{N}{\mathrm{df}_i}$$

\# of total docs

\# of docs containing

\# of occurrences of in j

Example $$\mathrm{sim}(i, j) = \mathrm{Encoder}(i) \cdot \mathrm{Encoder}(j)$$

Maps the text into an d-dimensional vector

An entire field of study on how to get (or learn) the similarity function better

[Faiss Wiki Documentation — Faiss documentation]()

FAISS

Method	Class name	`index_factory`	Main parameters	Bytes/vector	Exhaustive	Comments
Exact Search for L2	`IndexFlatL2`	`"Flat"`	`d`	`4*d`	yes	brute-force
Exact Search for Inner Product	`IndexFlatIP`	`"Flat"`	`d`	`4*d`	yes	also for cosine (normalize vectors beforehand)
Hierarchical Navigable Small World graph exploration	`IndexHNSWFlat`	`"HNSW,Flat"`	`d`, `M`	`4*d + x * M * 2 * 4`	no	
Inverted file with exact post-verification	`IndexIVFFlat`	`"IVFx,Flat"`	`quantizer`, `d`, `nlists`, `metric`	`4*d + 8`	no	Takes another index to assign vectors to inverted lists. The 8 additional bytes are the vector id that needs to be stored.
Locality-Sensitive Hashing (binary flat index)	`IndexLSH`	-	`d`, `nbits`	`ceil(nbits/8)`	yes	optimized by using random rotation instead of random projections
Scalar quantizer (SQ) in flat mode	`IndexScalarQuantizer`	`"SQ8"`	`d`	`d`	yes	4 and 6 bits per component are also implemented.
Product quantizer (PQ) in flat mode	`IndexPQ`	`"PQx"`, `"PQ"M"x"nbits`	`d`, `M`, `nbits`	`ceil(M * nbits / 8)`	yes	
IVF and scalar quantizer	`IndexIVFScalarQuantizer`	"IVFx,SQ4" "IVFx,SQ8"	`quantizer`, `d`, `nlists`, `qtype`	SQfp16: 2 * `d` + 8, SQ8: `d` + 8 or SQ4: `d/2` + 8	no	Same as the `IndexScalarQuantizer`
IVFADC (coarse quantizer+PQ on residuals)	`IndexIVFPQ`	`"IVFx,PQ"y"x"nbits`	`quantizer`, `d`, `nlists`, `M`, `nbits`	`ceil(M * nbits/8)+8`	no	
IVFADC+R (same as IVFADC with re-ranking based on codes)	`IndexIVFPQR`	`"IVFx,PQy+z"`	`quantizer`, `d`, `nlists`, `M`, `nbits`, `M_refine`, `nbits_refine`	`M+M_refine+8`	no	

Exact Search

Approximate Search
(Relatively easy to scale to ~1B elements)

CPU vs. GPU

https://github.com/facebookresearch/faiss/wiki

Vector Database

- A type of database that indexes and stores vector embeddings for fast retrieval and similarity search
- Advantages over vector indices like Faiss
  - Data management
  - Metadata storage and filtering
  - Scalability
  - Real-time updates
  - Backups and collections
  - Data security and access control

Pinecone

Chroma

Milvus

[Vector stores | Langchain]

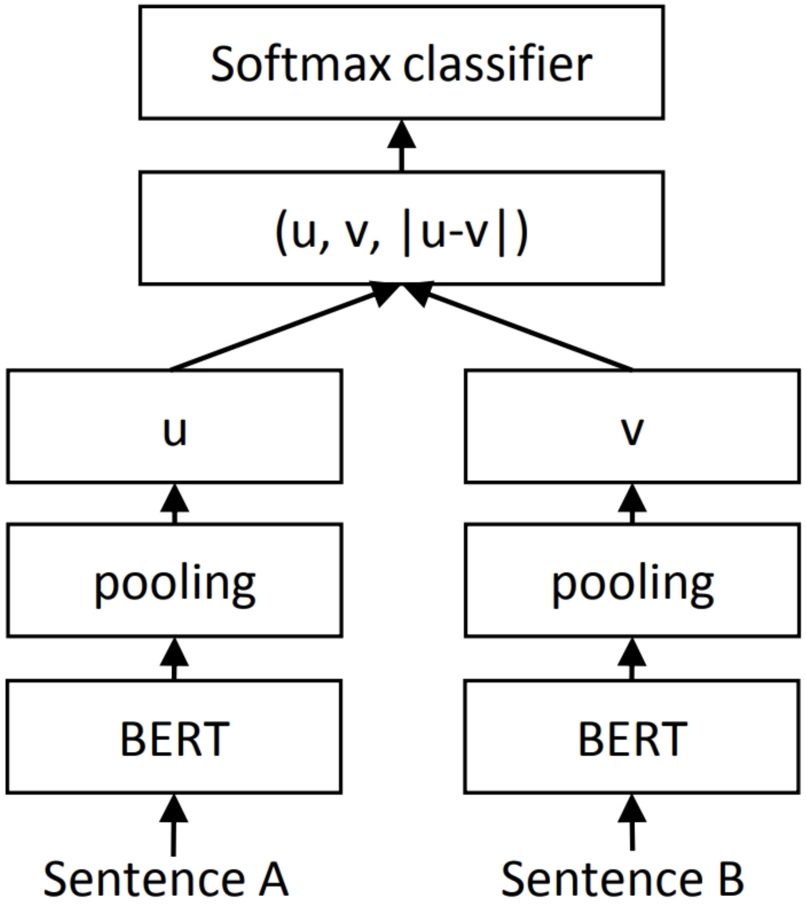
Sentence-BERT

- Training data:
  - Combination of the SNLI and the Multi-Genre NLI
- Default pooling strategy is MEAN.

Classification Objective Function. We concatenate the sentence embeddings u and v with the element-wise difference $|u-v|$ and multiply it with the trainable weight $W_t \in \mathbb{R}^{3n \times k}$:

$$o = \text{softmax}(W_t(u, v, |u-v|))$$

where n is the dimension of the sentence embeddings and k the number of labels. We optimize cross-entropy loss. This structure is depicted in



[1908.10084] Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks (arxiv.org)

Dense Passage Retriever (DPR)

- Retrieve k passages from M documents
  - k=20~100, M=millions~billions
- Passage encoder + query encoder (based on BERT)
  - Take the representation at the [CLS] token as the output
  - Given a question q at run-time, we derive its embedding $v_q = E_Q(q)$ and retrieve the top k passages with embeddings closest to v_q.

are the closest to the question vector. We define the similarity between the question and the passage using the dot product of their vectors:

$$\mathrm{sim}(q, p) = E_Q(q)^\intercal E_P(p). \qquad (1)$$

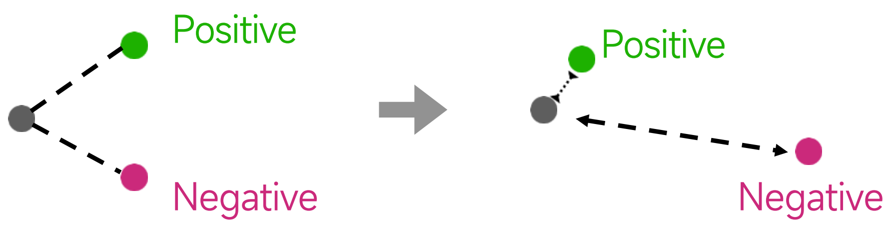
[2004.04906] Dense Passage Retrieval for Open-Domain Question Answering (arxiv.org)

Dense Passage Retriever (DPR)

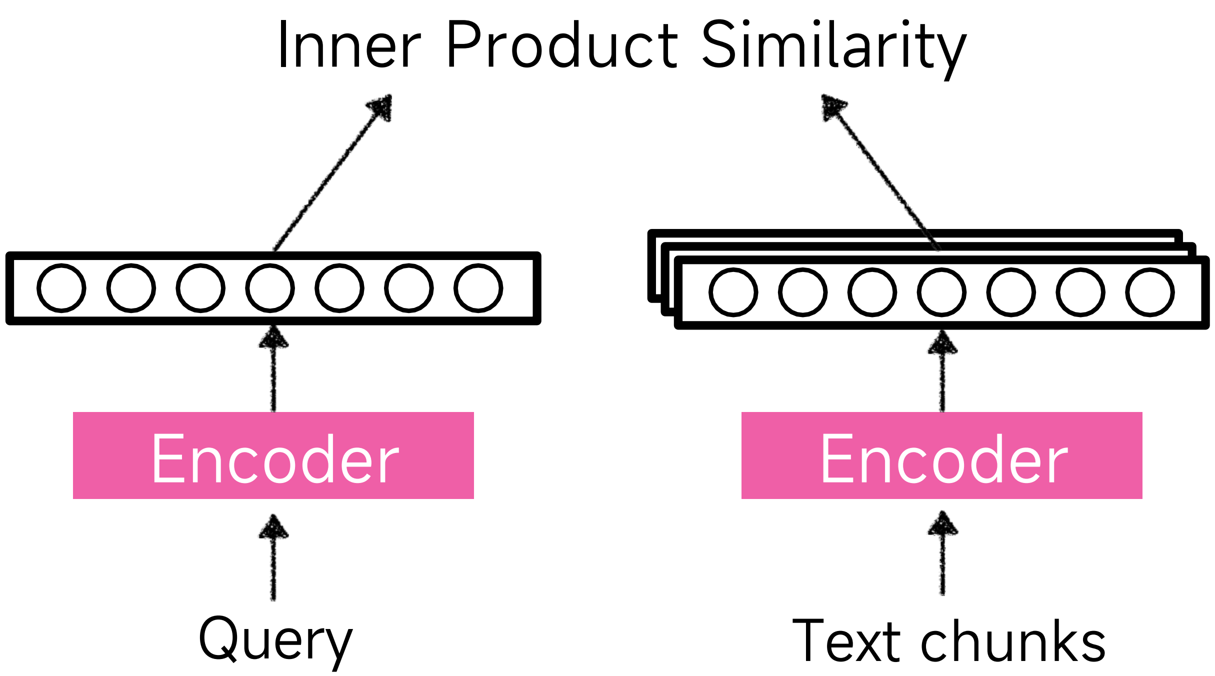
- Training data
  - Question Answering Datasets
  - Each instance contains one question q_i and one relevant (positive) passage p_i^+ , along with n irrelevant (negative) passages $p_{i,j}^-$.
- In-batch negative
  - Re-using gold passages from the same batch as negatives
  - Any (q_i, p_j) pair is a positive example when $i = j$, and negative otherwise

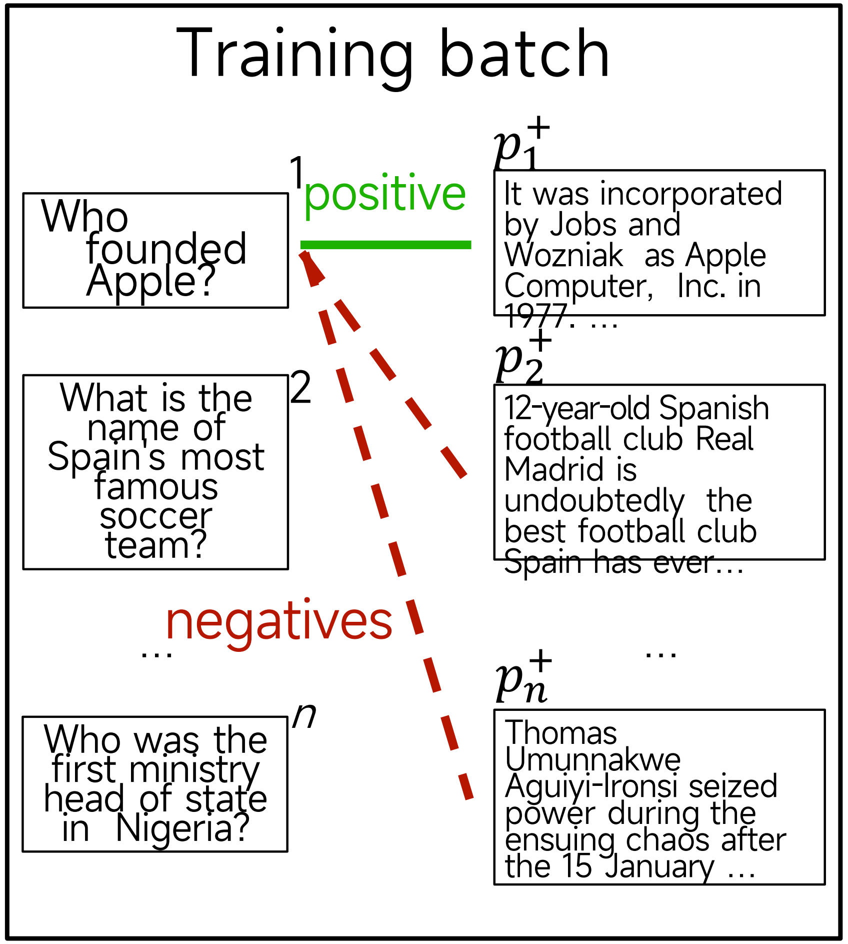
$$L(q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^-) = -\log \frac{e^{\mathrm{sim}(q_i, p_i^+)}}{e^{\mathrm{sim}(q_i, p_i^+)} + \sum_{j=1}^{n} e^{\mathrm{sim}(q_i, p_{i,j}^-)}}$$

Contrastive learning



Dense Passage Retriever (DPR)





[DPR/train_dense_encoder.py at main · facebookresearch/DPR]

ColBERT

- A ranking model based on contextualized late interaction over BERT
- Every query embedding interacts with all document embeddings via a MaxSim operator
- Share a single BERT model among our query and document encoders
  - Prepending a special token [Q] to queries and another token [D] to documents

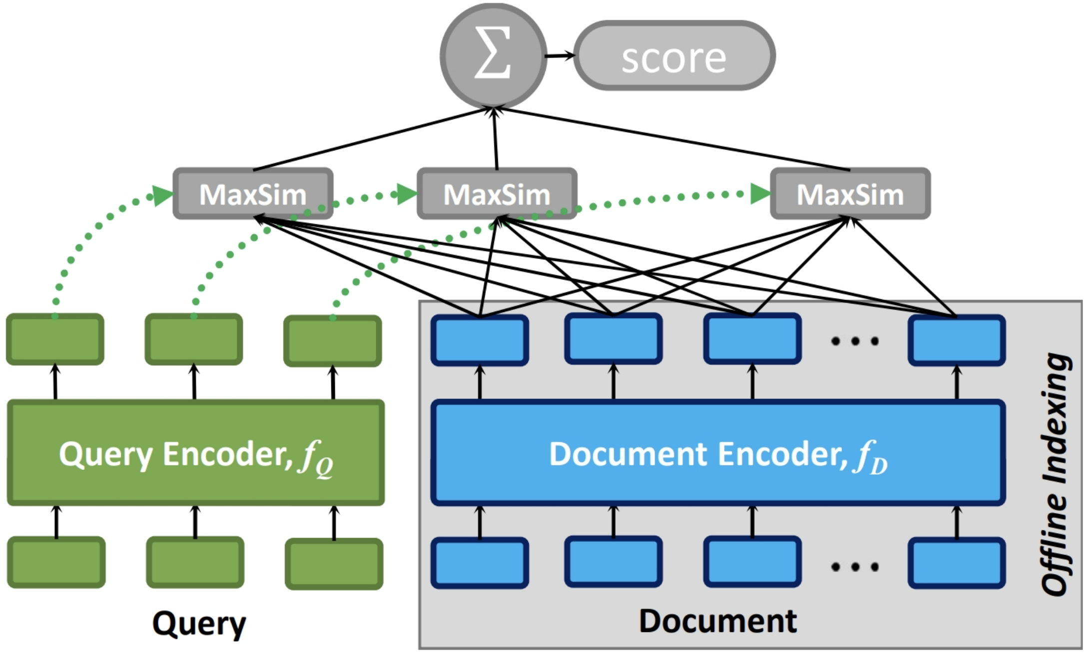


Figure 3: The general architecture of ColBERT given a query q and a document d.

[2004.12832] ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT (arxiv.org)

ColBERT

- A summation of maximum simil

E_q := Normalize(CNN(BERT("$[Q]q_0q_1...q_l$##...#")))

E_d := Filter(Normalize(CNN(BERT("$[D]d_0d_1...d_n$"))))

[ColBERT/colbert/modeling/colbert.py at colbertv1 · stanford-futuredata/ColBERT (github.com)]()

$$S_{q,d} := \sum_{i \in [|E_q|]} \max_{j \in [|E_d|]} E_{q_i} \cdot E_{d_j}^T$$

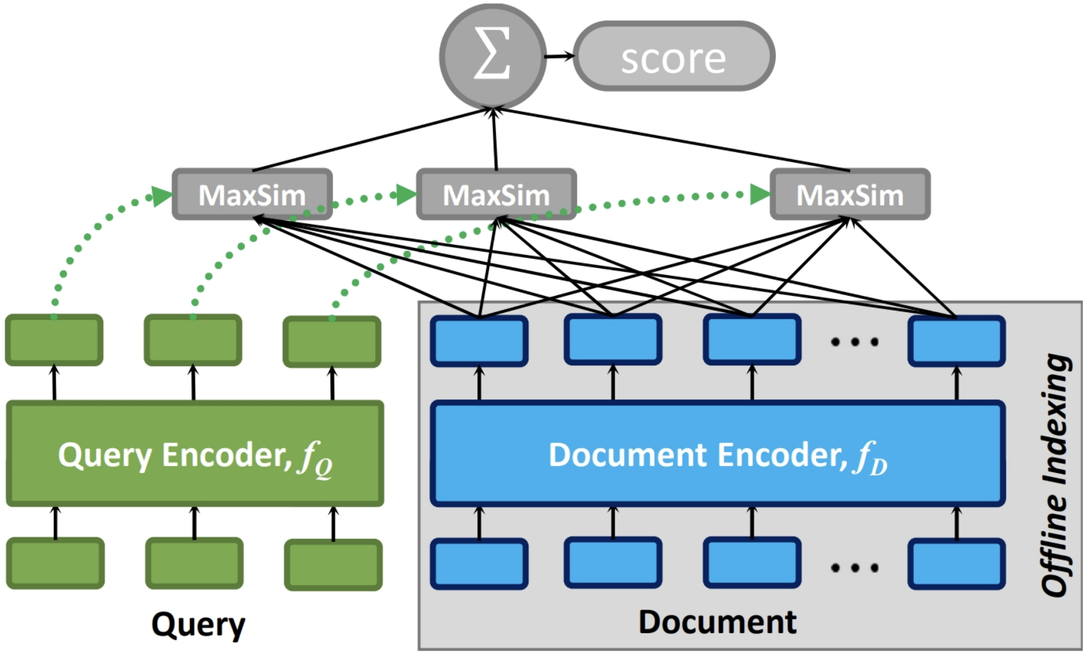


Figure 3: The general architecture of ColBERT given a query q and a document d.

GRIT

- LLM-based retriever

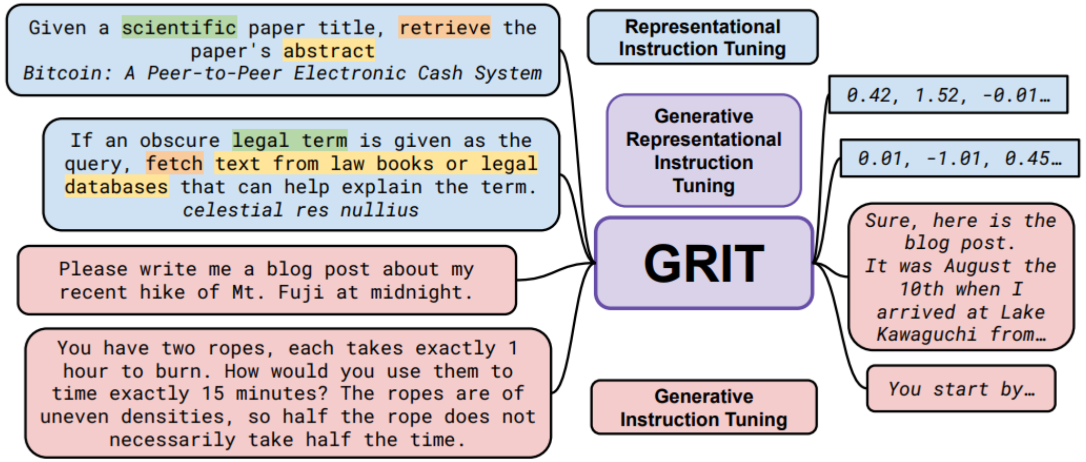


Figure 2: **GRIT.** The same model handles both text representation and generation tasks based on the given instruction. For representation tasks, instructions ideally contain the target domain, intent, and unit [5]. The representation is a tensor of numbers, while the generative output is text.

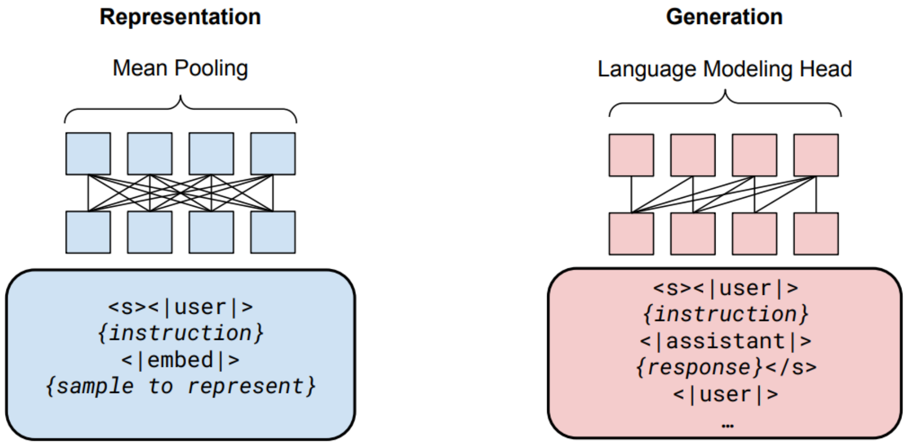


Figure 3: **GRITLM architecture and format.** *Left:* GRITLM uses bidirectional attention over the input for embedding tasks. Mean pooling is applied over the final hidden state to yield the final representation. *Right:* GRITLM uses causal attention over the input for generative tasks. A language modeling head on top of the hidden states predicts the next tokens. The format supports conversations with multiple turns (indicated with "…").

Self-RAG

- Self-Reflective Retrieval-Augmented Generation
- Learn to reflect on its own generation process by generating both task output and intermittent special tokens (i.e., reflection tokens).

Process

- Determines if augmenting with retrieved passages would be helpful
  - If so, it outputs a retrieval token that calls a retriever model on demand

[2310.11511] Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection (arxiv.org)
[2312.10997] Retrieval-Augmented Generation for Large Language Models: A Survey

Self-RAG

Inference process

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.

Table 1: Four types of reflection tokens used in Self-Rag. Each type uses several tokens to represent its output values. The bottom three rows are three types of Critique tokens, and **the bold text** indicates the most desirable critique tokens. x, y, d indicate input, output, and a relevant passage, respectively.

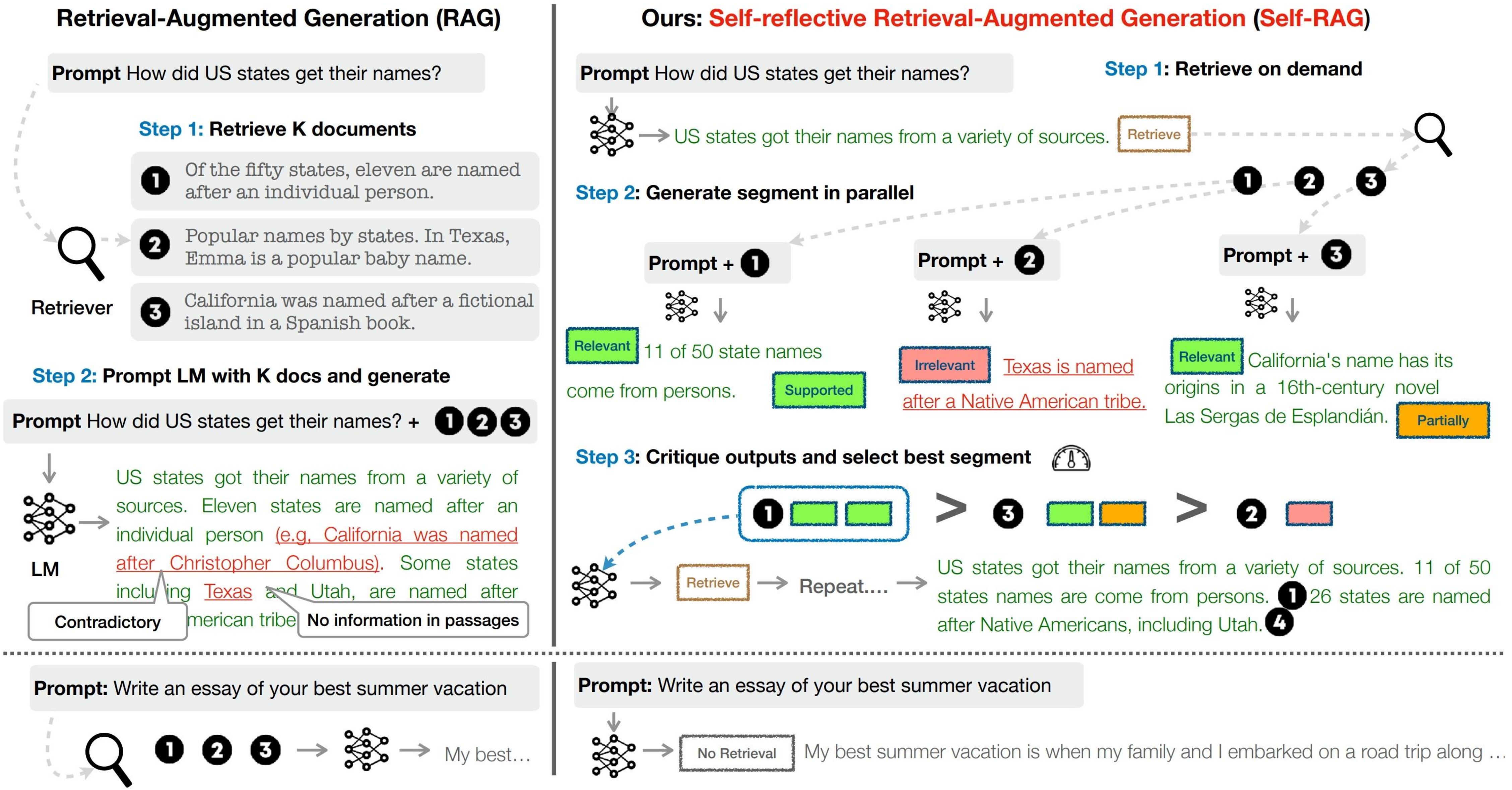
Algorithm 1 Self-Rag Inference

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
M predicts [Retrieve] given (x, y_{<t})
if [Retrieve] == Yes then
    Retrieve relevant text passages D using R given (x, y_{t-1})        ▷ Retrieve
    M predicts [IsRel] given x, d and y_t given x, d, y_{<t} for each d ∈ D   ▷ Generate
    M predicts [IsSup] and [IsUse] given x, y_t, d for each d ∈ D         ▷ Critique
    Rank y_t based on [IsRel], [IsSup], [IsUse]                ▷ Detailed in Section 3.3
else if [Retrieve] == No then
    M_gen predicts y_t given x                                             ▷ Generate
    M_gen predicts [IsUse] given x, y_t                                    ▷ Critique
```

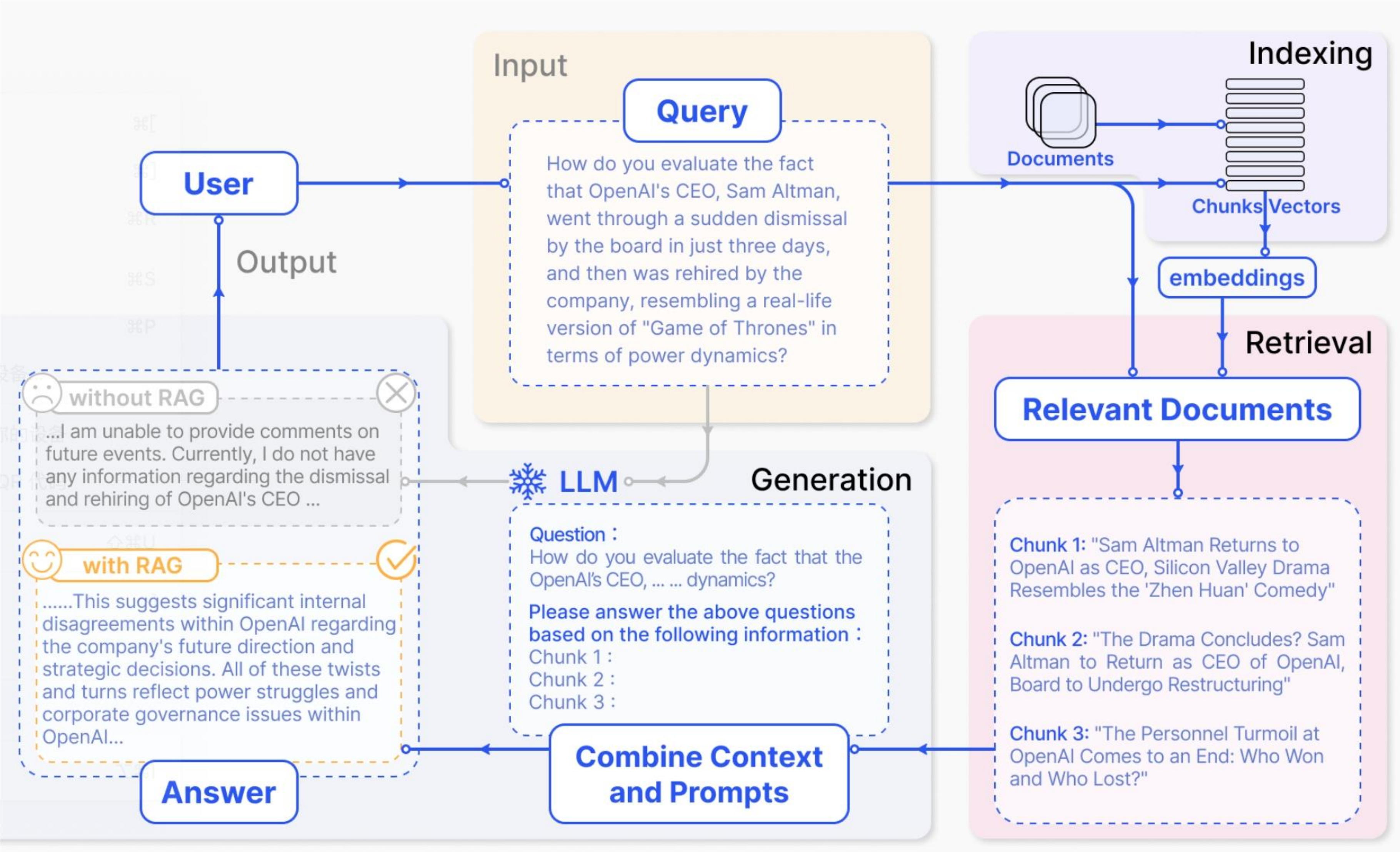
[2310.11511] Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection (arxiv.org)

Self-RAG



Retrieval-Augmented Generation (RAG)
Prompt How did US states get their names?
Step 1: Retrieve K documents
Of the fifty states, eleven are named after an individual person.
Popular names by states. In Texas, Emma is a popular baby name.
California was named after a fictional island in a Spanish book.
Retriever
Step 2: Prompt LM with K docs and generate
Prompt How did US states get their names? +
LM
US states got their names from a variety of sources. Eleven states are named after an individual person (e.g, California was named after Christopher Columbus). Some states including Texas and Utah, are named after Native American tribe.
Contradictory
No information in passages
Ours: Self-reflective Retrieval-Augmented Generation (Self-RAG)
Prompt How did US states get their names?
Step 1: Retrieve on demand
US states got their names from a variety of sources.
Retrieve
Step 2: Generate segment in parallel
Prompt +
Prompt +
Prompt +
Relevant
11 of 50 state names come from persons.
Supported
Irrelevant
Texas is named after a Native American tribe.
Relevant
California's name has its origins in a 16th-century novel Las Sergas de Esplandián.
Partially
Step 3: Critique outputs and select best segment
Retrieve
Repeat....
US states got their names from a variety of sources. 11 of 50 states names are come from persons. 26 states are named after Native Americans, including Utah.
Prompt: Write an essay of your best summer vacation
My best…
Prompt: Write an essay of your best summer vacation
No Retrieval
My best summer vacation is when my family and I embarked on a road trip along …

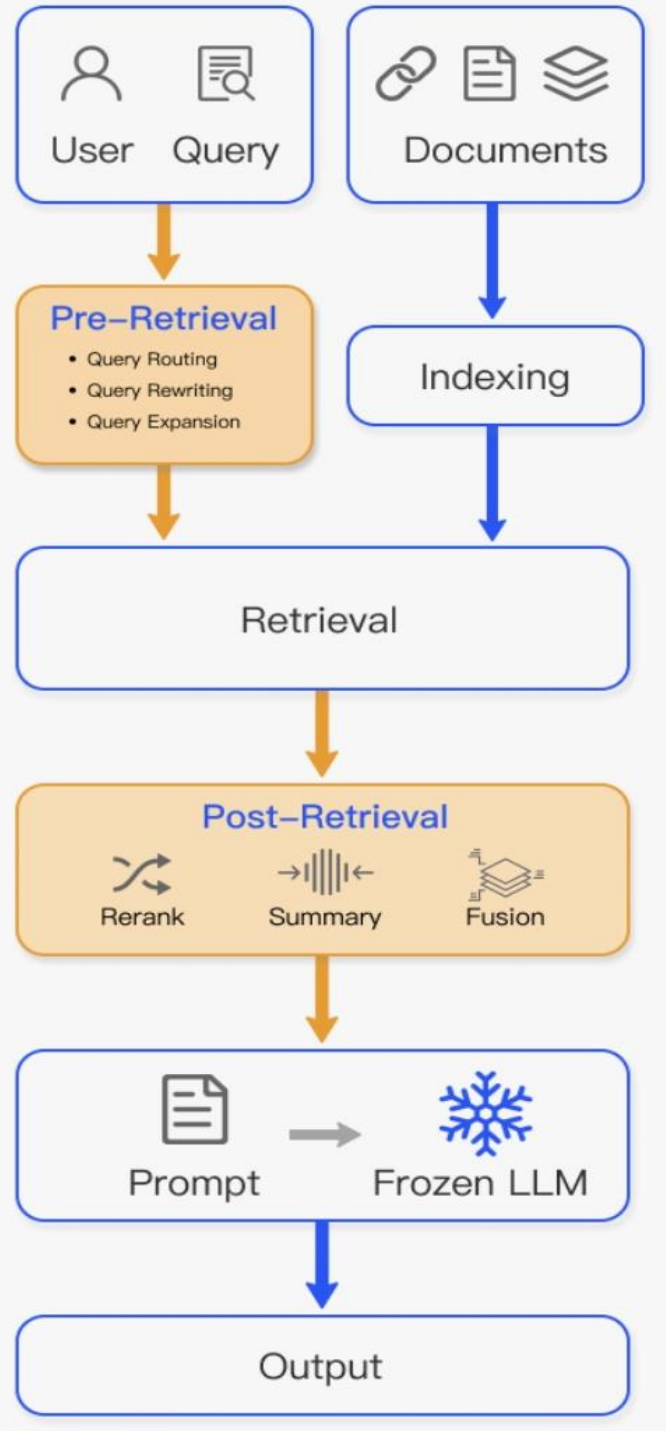
Example of RA-LLM



Next, we will dive deeper into the framework of RAG

Framework of RAG

- Key concepts
  - Query
  - Document
  - Indexing
  - Retriever
  - Generator



User Query

- The questions or instructions proposed by users
- E.g.,
  - Please share the details of the department meeting held yesterday.
  - 小儿感染诺如病毒有什么症状，应该如何治疗?
  - Tell me the best paper in the ICLR 2024 conference.
- Have different styles in different domains
- Can be nested and complex

Document

- Text chunks stores in the database
- Millions even trillions of documents
- Preprocessing
  - Text extraction (from html, pdf, etc.)
  - Tokenization
  - Chunking
  - Embedding
- Each document is represented with a single vector
  - Namely, indexing

Retriever

- Select the most relevant and similar document given the user query
- Calculate the cosine similarity of the query and document embeddings
- Different model structures
  - Dual encoders
  - Cross encoder
  - LLM

Pre-Retrieval process

- Optimizing the indexing structure
  - Goal: improve the quality of the content being indexed

Enhancing data granularity	Optimizing index structures	Adding metadata	Using mixed retrieval

- Optimizing the original query
  - Goal: make the user's original question clearer and more suitable for the retrieval task.

Query routing	Query transformation	Query expansion

Post-Retrieval process

- Rerank retrieved chunks
  - Relocate the most relevant content
- Context compressing
  - Selecting the essential information and shortening the context to be processed

Generator

- Use LLM to respond to the user query given
  - The retrieved documents
  - Instructions and suggestions
- Advanced generation
  - Chain-of-thought
  - Task planning
  - Self-reflection

Hands-on Coding Experience

- Build an RAG system with langchain toolkit

Reference

- awesome-papers-for-rag
- Retrieval-Augmented Generation for Large Language Models: A Survey
- langchain: Build context-aware reasoning applications
- PAI-RAG: An easy-to-use framework for modular RAG
- Building A RAG System with Gemma, MongoDB and Open Source Models

Thank you