Retrieval Augmented Generation

Ngo Van Linh
Hanoi University of Science and Technology

ONE LOVE. ONE FUTURE.

Outline

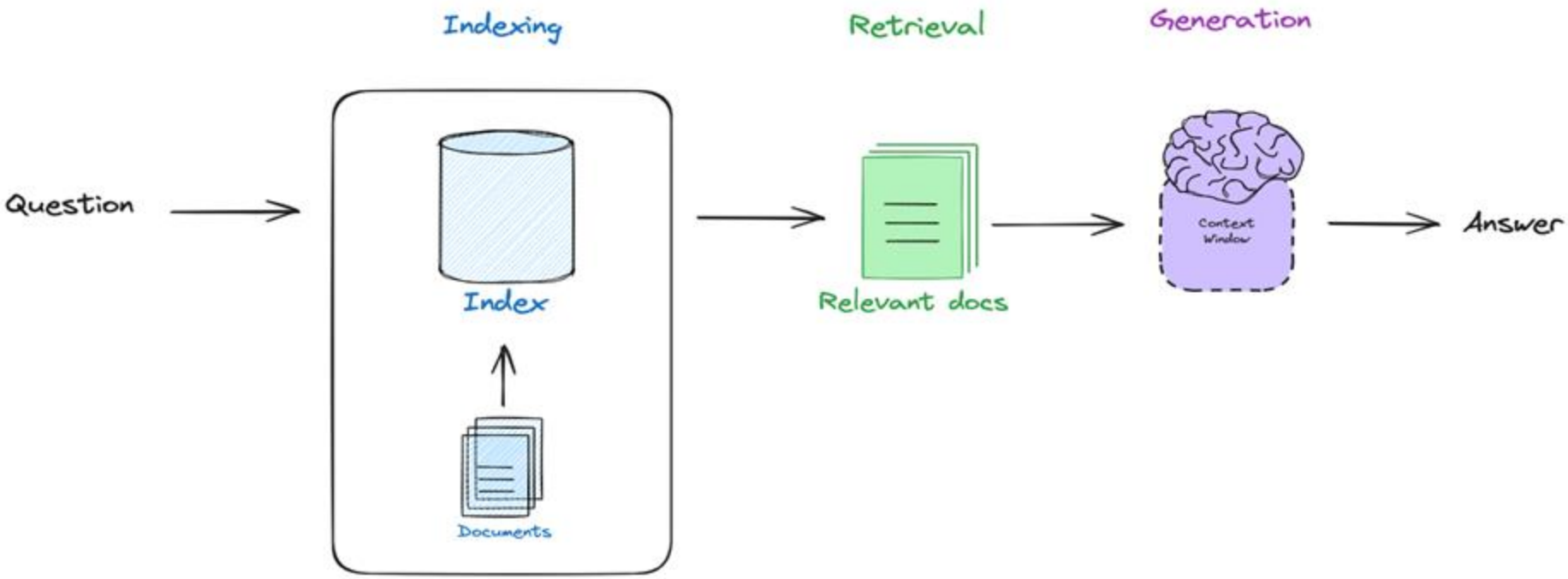
1. RAG Overview
2. Retrieval
   - Overview: Keyword search, Bi/Cross-Encoder
   - BGE
   - LLM for Retrieval
   - LLM2VEC
   - NV-Embed
3. RAG advanced techniques
   - Query Construction
   - Query Translation
   - Routing
   - Indexing
   - Retrieval
   - Generation
   - Why still need RAG in the era of Long Context LLMs?

1. RAG Overview: Why RAG?

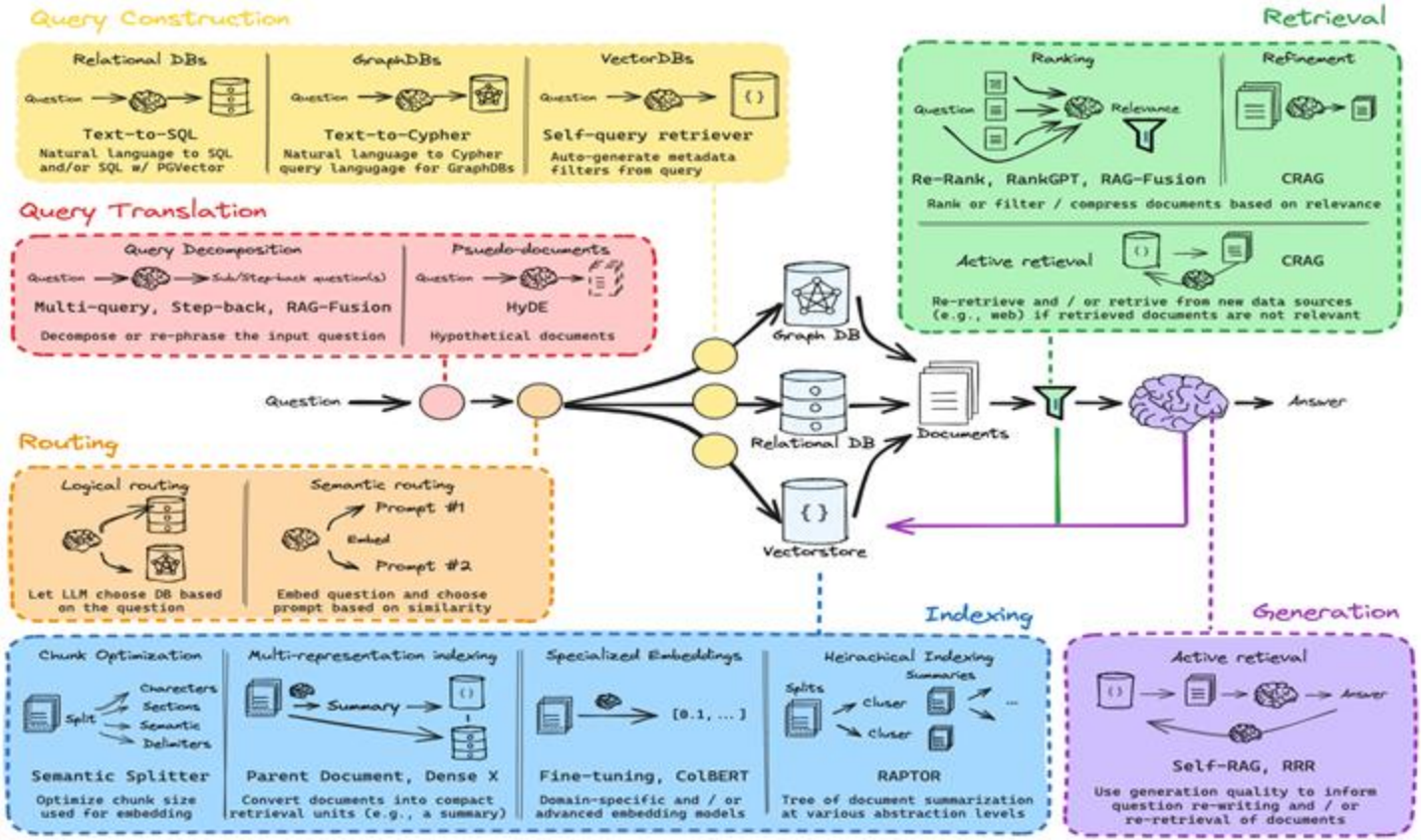
- LLMs excel at understanding and processing human language
- However,
  - Knowledge cutoff (GPT-3.5 is Sept 2021)
  - Hallucination
  - Bias
  - Fine-tune to specific data is expensive due to the model size
  - Limited context length
- LLMs are good at In-Context Learning
- Retrieval-augmented generation (RAG) provides LLMs with External Knowledge / Relevant Information

  ⇒ Improve quality

RAG Overview



RAG advanced techniques



https://github.com/langchain-ai/rag-from-scratch/blob/main/rag_from_scratch_1_to_4.ipynb

2. Retrieval

2.1. Tổng quan: Keyword search, Bi/Cross-Encoder

2.2. BGE-m3

2.3. LLM for Retrieval

2.4. LLM2VEC

2.5. NV-Embed

2.1. Tổng quan: Keyword search, Bi/Cross-Encoder

- Một số cách tiếp cận cơ bản:
  - Keyword search, feature search
  - Cross-encoder
  - Bi-encoder

BM25 (Best Matching)

- Ước lượng mức độ liên quan của các tài liệu đối với một truy vấn, sử dụng biểu diễn tài liệu tương tự biểu diễn TF-IDF
- Ưu điểm
  - Đơn giản, dễ hiểu, hiệu quả
  - Sử dụng chuẩn hóa độ dài tài liệu
  - Tốc độ nhanh
- Hạn chế
  - Không xem xét ngữ nghĩa và ngữ cảnh
  - Giả định độc lập thống kê giữa các từ truy vấn

BM25 (Best Matching)

Cho một câu truy vấn Q chứa các từ $q_1, q_2, \ldots, q_n$, điểm BM25 của một tài liệu D là:

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{\text{TF}(q_i, D) \cdot (k_1 + 1)}{\text{TF}(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})}$$

trong đó $\text{TF}(q_i, D)$ là số lần từ q_i xuất hiện trong tài liệu D, $|D|$ là độ dài của tài liệu D tính bằng số từ, $avgdl$ là độ dài trung bình của các tài liệu, k_1 và b là các siêu tham số, $\text{IDF}(q_i)$ là trọng số IDF của từ truy vấn q_i, thường được tính bởi:

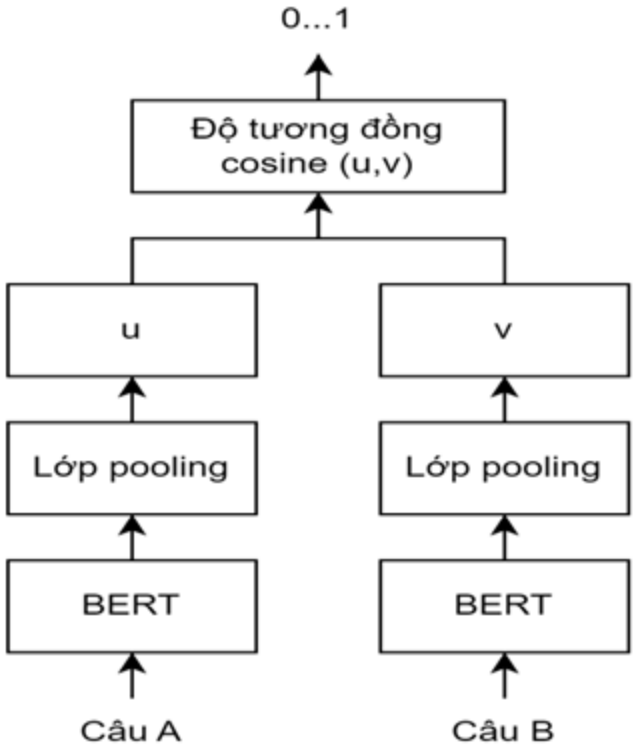
$$\text{IDF}(q_i) = \ln(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1)$$

trong đó N là tổng số tài liệu và $n(q_i)$ là số tài liệu chứa từ q_i.

Do các tài liệu dài thường có nhiều lần xuất hiện của một từ, siêu tham số b có tác dụng chuẩn hóa độ dài tài liệu với $\frac{|D|}{avgdl}$. Siêu tham số k_1 giảm thiểu ảnh hưởng của các từ có tần suất quá cao do chúng thường mang ít thông tin quan trọng.

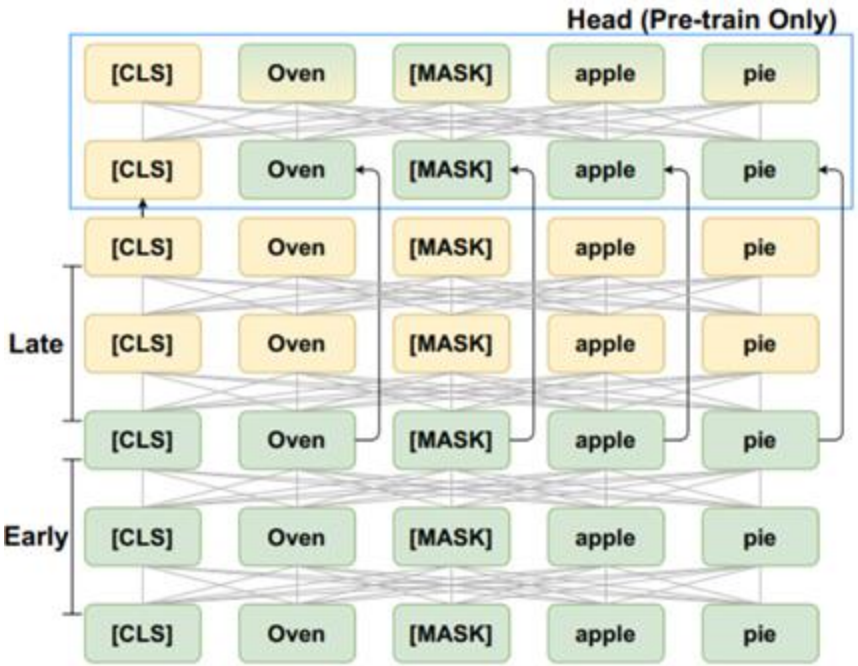
Bi-encoder

- Ưu điểm
  - Truy xuất dựa trên độ tương đồng ngữ nghĩa
  - Tốc độ truy xuất nhanh
  - Các véc-tơ nhúng có thể được tính trước



Kiến trúc chung của bi-encoder

Condenser



Kiến trúc Condenser

L. Gao **and** J. Callan, "Condenser: A pre-training architecture for dense retrieval," EMNLP 2021.

Contrastive learning

- Được ứng dụng thành công và rộng rãi trong truy xuất thông tin
- Hàm mất mát tương phản
- Các kỹ thuật chính
  - Kích thước batch lớn
  - Khai phá mẫu âm khó

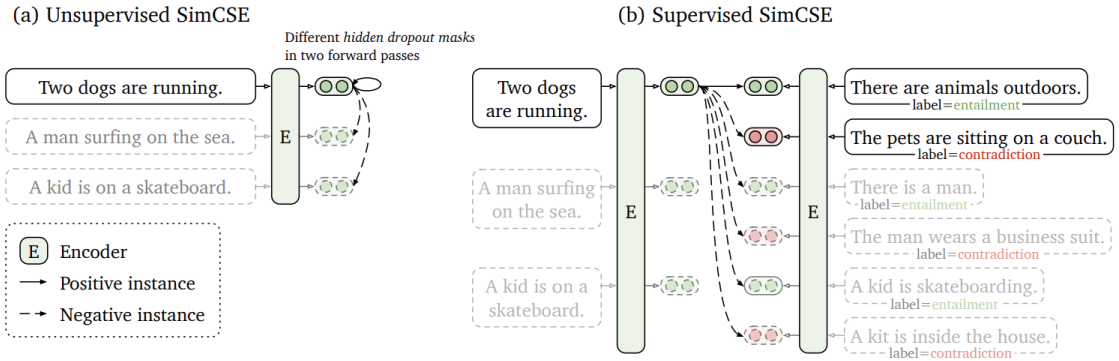


Figure 1: (a) Unsupervised SimCSE predicts the input sentence itself from in-batch negatives, with different hidden dropout masks applied. (b) Supervised SimCSE leverages the NLI datasets and takes the entailment (premise-hypothesis) pairs as positives, and contradiction pairs as well as other in-batch instances as negatives.

Gao, Tianyu, Xingcheng Yao, and Danqi Chen. "Simcse: Simple contrastive learning of sentence embeddings." *EMNLP* (2021).

coCondenser

- coCondenser được tiền huấn luyện với hàm mất mát tương phản không giám sát để warmup không gian nhúng
  - Trích rút hai đoạn văn từ mỗi tài liệu [Izacard et.al 2021]
  - Học tương phản trên các đoạn văn này

Izacard, Gautier, et al. "Unsupervised dense information retrieval with contrastive learning." *arXiv preprint arXiv:2112.09118* (2021).

RetroMAE

- Masked Auto-Encoder (MAE): A moderate ratio for encoder: 15-30%, and an aggressive ratio for decoder: 50-70%.

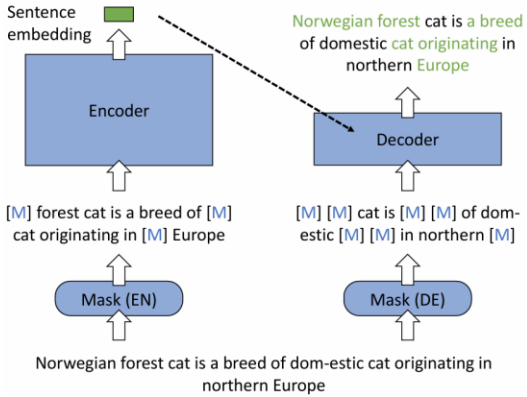


Figure 1: RetroMAE. The encoder utilizes a full-scale BERT, whose input is moderately masked. The decoder is a one-layer transformer, whose input is aggressively masked. The original input is recovered based on the sentence embedding and the decoder's input via MLM.

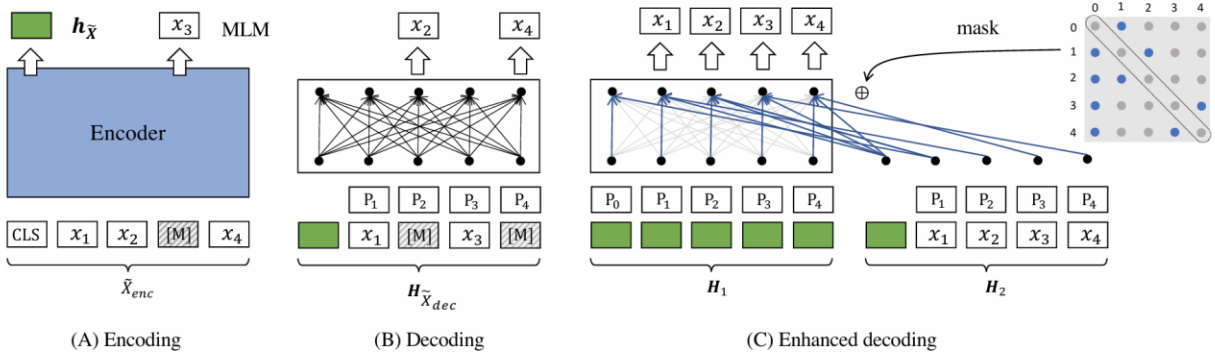


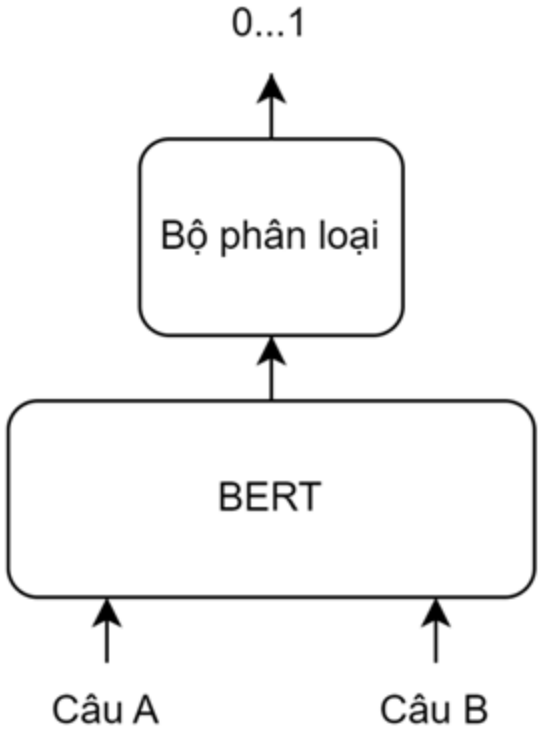
Figure 2: RetroMAE pre-training workflow. (A) Encoding: the input is moderately masked and encoded as the sentence embedding (the green rectangle). (B) Decoding: the input is aggressively masked, and joined with the sentence embedding to reconstruct the masked tokens (the shadowed tokens). (C) Enhanced encoding: all input tokens are reconstructed based on the sentence embedding and the visible context in each row (defined in Eq. 7); the main diagonal positions are filled with $-\infty$ (grey), and positions for the visible context are filled with 0 (blue).

Xiao, S., Liu, Z., Shao, Y., & Cao, Z. (2022, December). RetroMAE: Pre-Training Retrieval-oriented Language Models Via Masked Auto-Encoder. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing* (pp. 538-548).

Cross-encoder

- So với kiến trúc bi-encoder
  - Thường độ chính xác cao hơn ✓
  - Tốc độ chậm hơn ✗

⇒ **Sử dụng kết hợp với bi-encoder**



Kiến trúc chung của cross-encoder

2.2. BGE_M3

- Three challenges:
  - Most of the embedding models are tailored only for English, leaving few viable options for the other languages.
  - The existing embedding models are usually trained for one single retrieval functionality. However, typical IR systems call for the compound workflow of multiple retrieval methods.
  - Most of the embedding models can only support short inputs
- Contribution
  - Multi-linguality
  - Multi-functionality: Dense retrieval, lexical (sparse) retrieval, and multi-vector retrieval
  - Multi-granularity: Input granularities, spanning from short inputs like sentences and passages, to long documents of up to 8,192 input tokens.
  - Propose a novel training framework of self-knowledge distillation and efficient batching strategy

Chen, J., Xiao, S., Zhang, P., Luo, K., Lian, D., & Liu, Z. (2024). Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Architecture

- Adopt a further pre-trained XLM-RoBERTa12 as the foundational model
- Extend the max position to 8192


```
{
  "_name_or_path": "",
  "architectures": [
    "XLMRobertaModel"
  ],
  "attention_probs_dropout_prob": 0.1,
  "bos_token_id": 0,
  "classifier_dropout": null,
  "eos_token_id": 2,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 1024,
  "initializer_range": 0.02,
  "intermediate_size": 4096,
  "layer_norm_eps": 1e-05,
  "max position embeddings": 8194,
  "model_type": "xlm-roberta",
  "num_attention_heads": 16,
  "num_hidden_layers": 24,
  "output_past": true,
  "pad_token_id": 1,
  "position_embedding_type": "absolute",
  "torch_dtype": "float32",
  "transformers_version": "4.33.0",
  "type_vocab_size": 1,
  "use_cache": true,
  "vocab_size": 250002
}
```

Datasets: Unsupervised and Supervised datasets

Data Source	Language	Size
Unsupervised Data		
MTP	EN, ZH	291.1M
S2ORC, Wikipeida	EN	48.3M
xP3, mC4, CC-News	Multi-Lingual	488.4M
NLLB, CCMatrix	Cross-Lingual	391.3M
CodeSearchNet	Text-Code	344.1K
Total	–	1.2B
Fine-tuning Data		
MS MARCO, HotpotQA, NQ, NLI, etc.	EN	1.1M
DuReader, T^2-Ranking, NLI-zh, etc.	ZH	386.6K
MIRACL, Mr.TyDi	Multi-Lingual	88.9K
MultiLongDoc	Multi-Lingual	41.4K

Table 1: Specification of training data.

- Generate synthetic data to mitigate the shortage of long document retrieval tasks and introduce extra multi-lingual fine-tuning data (denoted as MultiLongDoc)
- Sample lengthy articles from Wikipedia, Wudao and mC4 datasets and randomly choose paragraphs from them.
- Use GPT3.5 to generate questions based on these paragraphs.
- The generated question and the sampled article constitute a new text pair to the fine-tuning data

Training process:

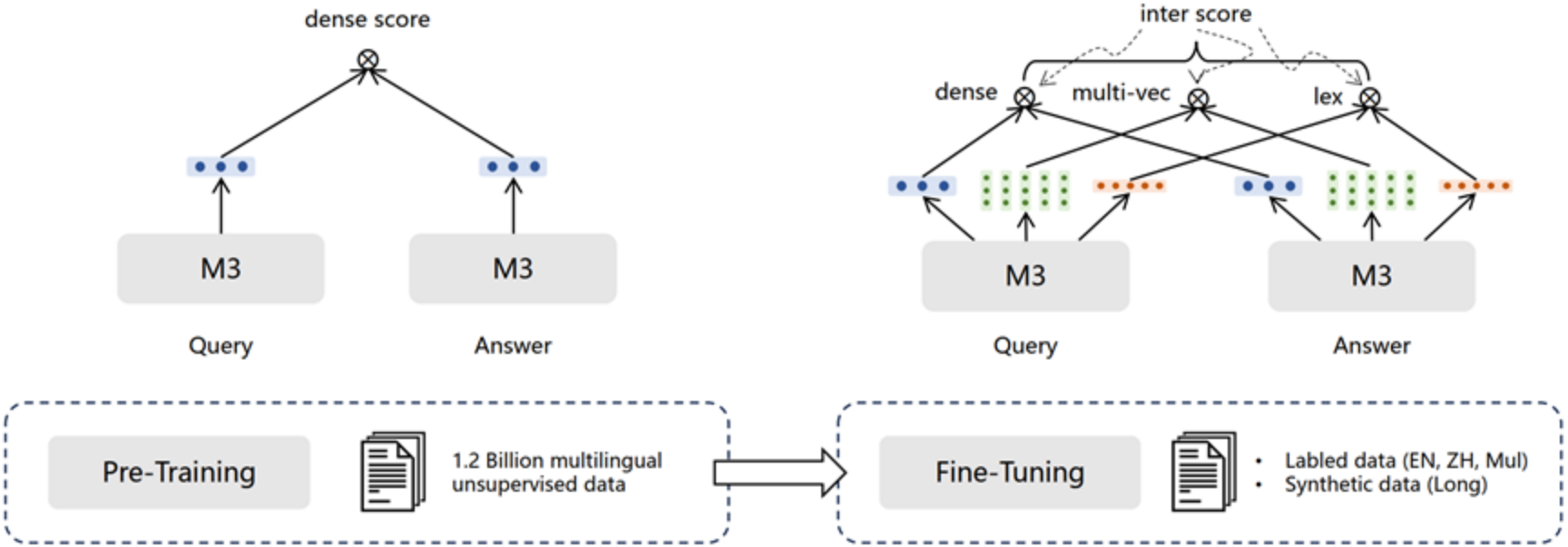
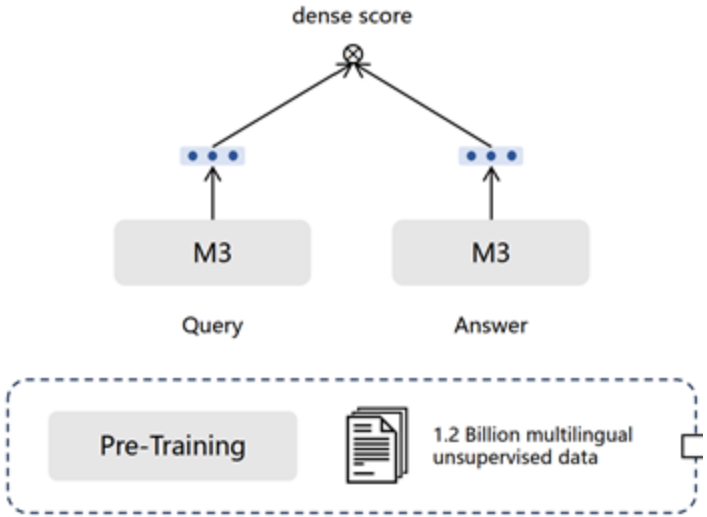


Figure 2: Multi-stage training process of M3-Embedding with self-knowledge distillation.

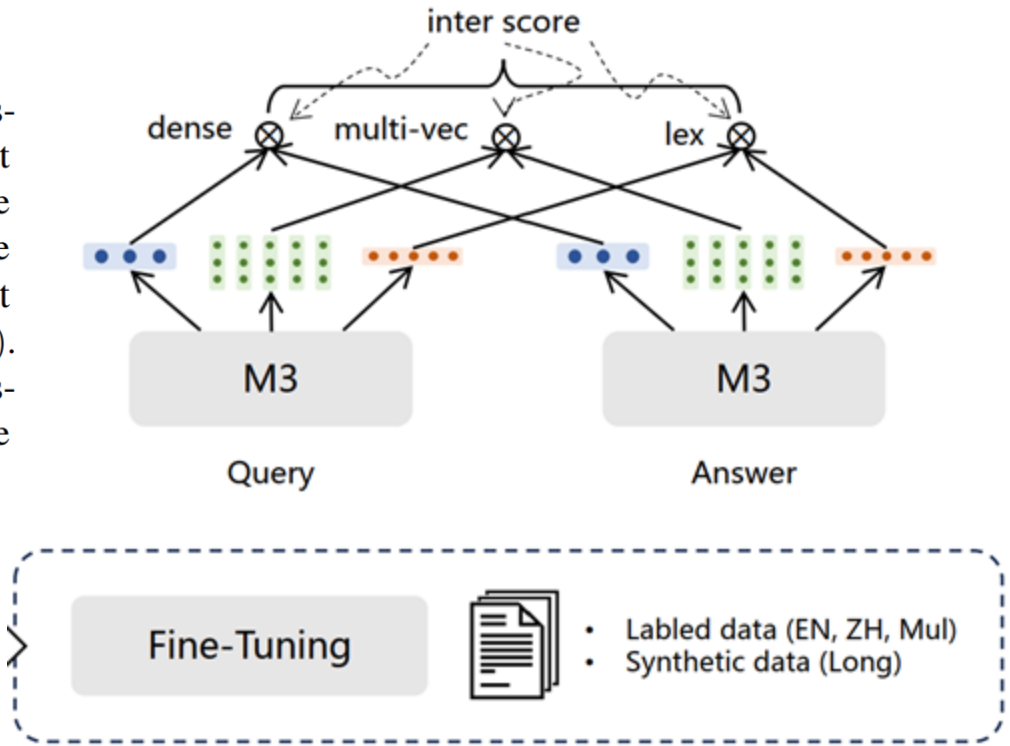
Phase 1: Pre-training

- The text encoder is pre-trained with the massive unsupervised data, where only the dense retrieval is trained in the basic form of contrastive learning



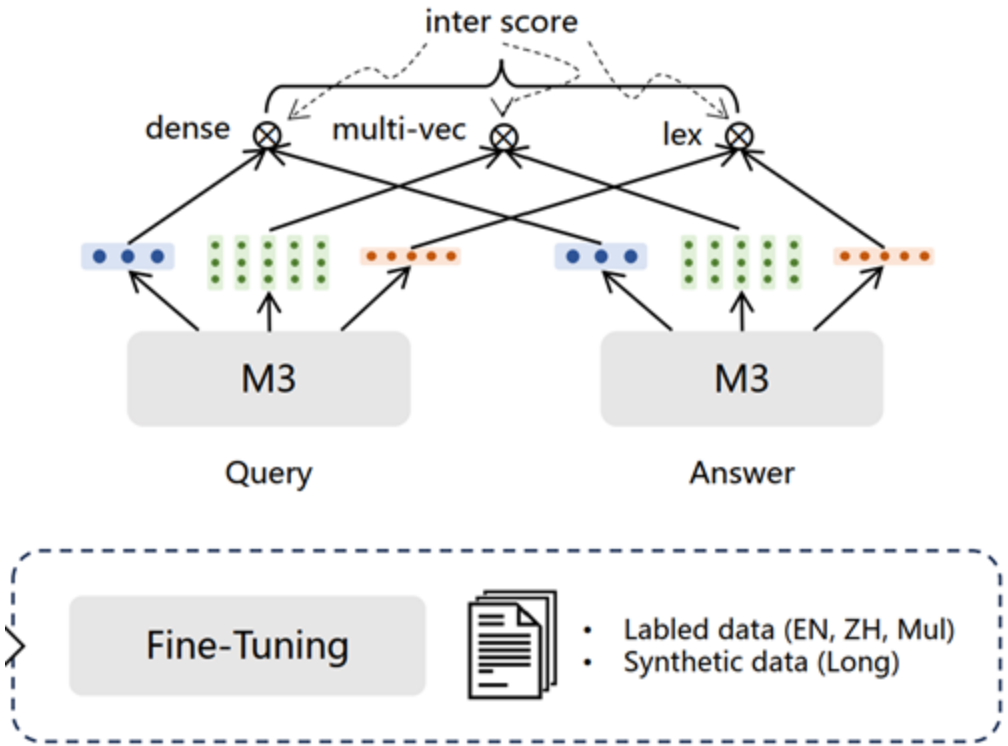
Phase 2: Finetuning

• **Dense retrieval**. The input query q is transformed into the hidden states $\mathbf{H_q}$ based on a text encoder. We use the normalized hidden state of the special token "[CLS]" for the representation of the query: $e_q = norm(\mathbf{H_q}[0])$. Similarly, we can get the embedding of passage p as $e_p = norm(\mathbf{H_p}[0])$. Thus, the relevance score between query and passage is measured by the inner product between the two embeddings e_q and e_p: $s_{dense} \leftarrow \langle e_p, e_q \rangle$.



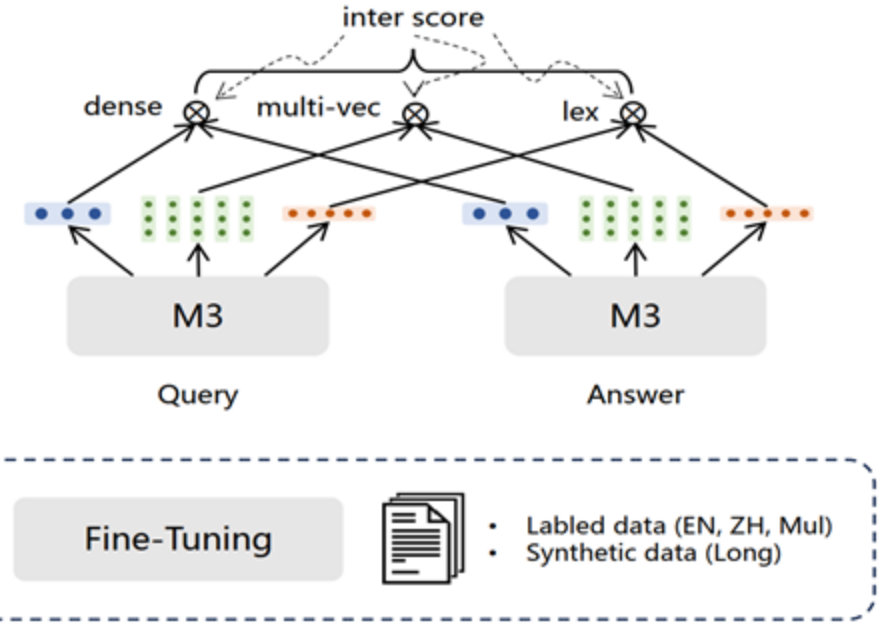
Phase 2: Finetuning

• **Lexical Retrieval**. The output embeddings are also used to estimate the importance of each term to facilitate lexical retrieval. For each term t within the query (a term is corresponding to a token in our work), the term weight is computed as $w_{q_t} \leftarrow \text{Relu}(\mathbf{W}_{lex}^T \mathbf{H}_{\mathbf{q}}[i]))$, where $\mathbf{W}_{lex} \in \mathcal{R}^{d \times 1}$ is the matrix mapping the hidden state to a float number. If a term t appears multiple times in the query, we only retain its max weight. We use the same way to compute the weight of each term in the passage. Based on the estimation term weights, the relevance score between query and passage is computed by the joint importance of the co-existed terms (denoted as $q \cap p$) within the query and passage: $s_{lex} \leftarrow \sum_{t \in q \cap p} (w_{q_t} * w_{p_t})$.



Phase 2: Finetuning

• **Multi-Vector Retrieval**. As an extension of dense retrieval, the multi-vector method makes use of the entire output embeddings for the representation of query and passage: $E_q = norm(\mathbf{W}_{mul}^T \mathbf{H_q})$, $E_p = norm(\mathbf{W}_{mul}^T \mathbf{H_p})$, where $\mathbf{W}_{mul} \in \mathbb{R}^{d \times d}$ is the learnable projection matrix. Following ColBert(Khattab and Zaharia, 2020), we use late-interaction to compute the fine-grained relevance score: $s_{mul} \leftarrow \frac{1}{N} \sum_{i=1}^{N} \max_{j=1}^{M} E_q[i] \cdot E_p^T[j]$; N and M are the lengths of query and passage.



- The final retrieval result is re-ranked based on the integrated relevance score:

$$s_{rank} \leftarrow w_1 \cdot s_{dense} + w_2 \cdot s_{lex} + w_3 \cdot s_{mul}$$

Loss function

- The weighted sum of different prediction scores:

$$s_{inter} \leftarrow w_1 \cdot s_{dense} + w_2 \cdot s_{lex} + w_3 \cdot s_{mul}.$$

- Contrastive loss for each component:

$$\mathcal{L}_{s(\cdot)} = -\log \frac{\exp(s(q,p^*)/\tau)}{\sum_{p\in\{p^*,P'\}} \exp(s(q,p)/\tau)}.$$

where , p∗ and P′ stand for the positive and negative samples to the query q; s(·) is any of the functions within $s_{dense(.)}, s_{lex(.)}, s_{mul(.)}, s_{inter(.)}$

- The weighted sum of losses:

$$\mathcal{L} \leftarrow \left(\lambda_1 \cdot \mathcal{L}_{dense} + \lambda_2 \cdot \mathcal{L}_{lex} + \lambda_3 \cdot \mathcal{L}_{mul} + \mathcal{L}_{inter}\right)/4.$$

- Hyperparameter setting:

$$w_1 = 1, w_2 = 0.3, w_3 = 1, \lambda_1 = 1, \lambda_2 = 0.1 \text{ and } \lambda_3 = 1$$

Self-knowledge distillation

- Employ the integration score s_{inter} as the teacher, each functional score is a student:

$$\mathcal{L}'_* \leftarrow -p(s_{inter}) * \log p(s_*).$$

where p(·) is the softmax activation; s_* is any of the members within $s_{dense(.)}, s_{lex(.)}, s_{mul(.)}$

- Integrate and normalize the modified loss function

$$\mathcal{L}' \leftarrow \left(\lambda_1 \cdot \mathcal{L}'_{dense} + \lambda_2 \cdot \mathcal{L}'_{lex} + \lambda_3 \cdot \mathcal{L}'_{mul}\right)/3.$$

- Final loss: The linear combination of contrastive loss and self-knowledge distillation:

$$\mathcal{L}_{final} \leftarrow \left(\mathcal{L} + \mathcal{L}'\right)/2.$$

Efficient Batching

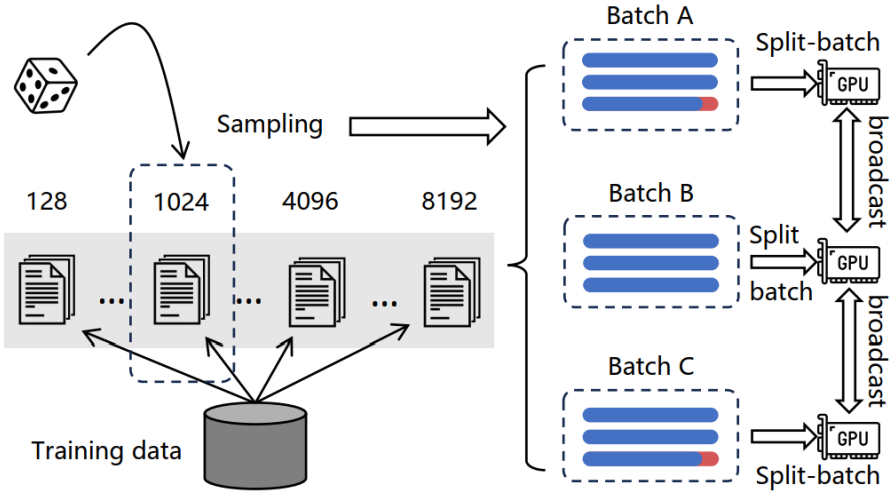


Figure 3: **Efficient Batching.** (Data is grouped and sampled by length. Gradient-checkpointing and cross-GPU broadcasting are enabled to save memory.)

Experimental Results

Model	Avg	ar	bn	en	es	fa	fi	fr	hi	id	ja	ko	ru	sw	te	th	zh	de	yo
Baselines (*Prior Work*)																			
BM25	31.9	39.5	48.2	26.7	7.7	28.7	45.8	11.5	35.0	29.7	31.2	37.1	25.6	35.1	38.3	49.1	17.5	12.0	56.1
mDPR	41.8	49.9	44.3	39.4	47.8	48.0	47.2	43.5	38.3	27.2	43.9	41.9	40.7	29.9	35.6	35.8	51.2	49.0	39.6
mContriever	43.1	52.5	50.1	36.4	41.8	21.5	60.2	31.4	28.6	39.2	42.4	48.3	39.1	56.0	52.8	51.7	41.0	40.8	41.5
$mE5_{large}$	66.6	76.0	75.9	52.9	52.9	59.0	77.8	54.5	62.0	52.9	70.6	66.5	67.4	74.9	84.6	80.2	56.0	56.4	78.3
$E5_{mistral\text{-}7b}$	63.4	73.3	70.3	57.3	52.2	52.1	74.7	55.2	52.1	52.7	66.8	61.8	67.7	68.4	73.9	74.0	54.0	54.1	79.7
OpenAI-3	54.9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
M3-Embedding (*Our Work*)																			
Dense	69.2	78.4	80.0	56.9	56.1	60.9	78.6	58.3	59.5	56.1	72.8	69.9	70.1	78.7	86.2	82.6	62.7	56.7	81.8
Sparse	53.9	67.1	68.9	43.8	38.6	45.1	65.4	35.3	48.2	48.9	56.1	61.5	44.5	57.9	79.1	70.9	36.1	32.5	70.0
Multi-vec	70.5	79.6	81.0	59.3	57.8	62.0	80.1	59.4	61.5	58.3	74.5	71.2	71.2	79.1	87.9	83.0	63.7	58.0	82.4
Dense+Sparse	70.4	79.6	80.7	58.8	58.1	62.3	79.7	58.0	62.9	58.3	73.9	71.2	69.8	78.5	87.2	83.1	63.5	57.7	83.3
All	**71.5**	**80.2**	**81.5**	**59.6**	**59.7**	**63.4**	**80.4**	**61.2**	**63.3**	**59.0**	**75.2**	**72.1**	**71.7**	**79.6**	**88.1**	**83.7**	**64.9**	**59.8**	**83.5**

Table 1: **Multi-lingual retrieval performance on the MIRACL dev set** (measured by nDCG@10).

Experimental Results

	Max Length	Avg	ar	de	en	es	fr	hi	it	ja	ko	pt	ru	th	zh
Baselines (*Prior Work*)															
BM25	8192	53.6	45.1	52.6	57.0	78.0	75.7	43.7	70.9	36.2	25.7	82.6	61.3	33.6	34.6
mDPR	512	23.5	15.6	17.1	23.9	34.1	39.6	14.6	35.4	23.7	16.5	43.3	28.8	3.4	9.5
mContriever	512	31.0	25.4	24.2	28.7	44.6	50.3	17.2	43.2	27.3	23.6	56.6	37.7	9.0	15.3
$mE5_{large}$	512	34.2	33.0	26.9	33.0	51.1	49.5	21.0	43.1	29.9	27.1	58.7	42.4	15.9	13.2
$E5_{mistral\text{-}7b}$	8192	42.6	29.6	40.6	43.3	70.2	60.5	23.2	55.3	41.6	32.7	69.5	52.4	18.2	16.8
text-embedding-ada-002	8191	32.5	16.3	34.4	38.7	59.8	53.9	8.0	46.5	28.6	20.7	60.6	34.8	9.0	11.2
jina-embeddings-v2-base-en	8192	-	-	-	37.0	-	-	-	-	-	-	-	-	-	-
M3-Embedding (*Our Work*)															
Dense	8192	52.5	47.6	46.1	48.9	74.8	73.8	40.7	62.7	50.9	42.9	74.4	59.5	33.6	26.0
Sparse	8192	62.2	58.7	53.0	62.1	87.4	82.7	49.6	74.7	53.9	47.9	85.2	72.9	40.3	40.5
Multi-vec	8192	57.6	56.6	50.4	55.8	79.5	77.2	46.6	66.8	52.8	48.8	77.5	64.2	39.4	32.7
Dense+Sparse	8192	64.8	63.0	56.4	**64.2**	**88.7**	**84.2**	**52.3**	**75.8**	58.5	53.1	**86.0**	**75.6**	42.9	**42.0**
All	8192	**65.0**	**64.7**	**57.9**	63.8	86.8	83.9	52.2	75.5	**60.1**	**55.7**	85.4	73.8	**44.7**	40.0
M3-w.o.long															
Dense-w.o.long	8192	41.2	35.4	35.2	37.5	64.0	59.3	28.8	53.1	41.7	29.8	63.5	51.1	19.5	16.5
Dense-w.o.long (MCLS)	8192	45.0	37.9	43.3	41.2	67.7	64.6	32.0	55.8	43.4	33.1	67.8	52.8	27.2	18.2

Table 3: **Evaluation of multilingual long-doc retrieval on the MLDR test set** (measured by nDCG@10).

2.3. LLM for Retrieval: Motivations

- Existing multi-stage approaches suffer from several drawbacks
  - They entail a complex multi-stage training pipeline that demands substantial engineering efforts to curate large amounts of relevance pairs
  - They rely on manually collected datasets that are often constrained by the diversity of tasks and the coverage of languages.
  - Most existing methods employ BERT-style encoders as the backbone, neglecting the recent advances of training better LLMs and related techniques such as context length extension

Improving Text Embeddings with Large Language Models

Liang Wang, Nan Yang, Xiaolong Huang,
Linjun Yang, Rangan Majumder, Furu Wei
Microsoft Corporation
{wangliang,nanya,xiaolhu,yang.linjun,ranganm,fuwei}@microsoft.com

Abstract

In this paper, we introduce a novel and simple method for obtaining high-quality text embeddings using only synthetic data and less than 1k training steps. Unlike existing methods that often depend on multi-stage intermediate pre-training with billions of weakly-supervised text pairs, followed by fine-tuning with a few labeled datasets, our method does not require building complex training pipelines or relying on manually collected datasets that are often constrained by task diversity and language coverage. We leverage proprietary LLMs to generate diverse synthetic data for hundreds of thousands of text embedding tasks across 93 languages. We then fine-tune open-source decoder-only LLMs on the synthetic data using standard contrastive loss. Experiments demonstrate that our method achieves strong performance on highly competitive text embedding benchmarks without using any labeled data. Furthermore, when fine-tuned with a mixture of synthetic and labeled data, our model sets new state-of-the-art results on the BEIR and MTEB benchmarks.

attribution of generated text is another important application of text embeddings (Gao et al., 2023) that can improve the interpretability and trustworthiness of LLMs.

Previous studies have demonstrated that weighted average of pre-trained word embeddings (Pennington et al., 2014; Arora et al., 2017) is a strong baseline for measuring semantic similarity. However, these methods fail to capture the rich contextual information of natural language. With the advent of pre-trained language models (Devlin et al., 2019), Sentence-BERT (Reimers and Gurevych, 2019) and SimCSE (Gao et al., 2021) have been proposed to learn text embeddings by fine-tuning BERT on natural language inference (NLI) datasets. To further enhance the performance and robustness of text embeddings, state-of-the-art methods like E5 (Wang et al., 2022b) and BGE (Xiao et al., 2023) employ a more complex multi-stage training paradigm that first pre-trains on billions of weakly-supervised text pairs, and then fine-tunes on several high-quality labeled datasets.

Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., & Wei, F. (2024). Improving text embeddings with large language models. *ACL 2024*.

LLM for Retrieval: Contributions

- Use LLMs to generate synthetic data for a diverse range of text embedding tasks in 93 languages, covering hundreds of thousands of embedding tasks
  - Use a two-step prompting strategy that first prompts the LLMs to brainstorm a pool of candidate tasks, and then prompts the LLMs to generate data conditioned on a given task from the pool.

- Fine-tune powerful open-source LLMs rather than small BERT-style models

Generating synthetic data with GPT4

Brainstorm a list of potentially useful text retrieval tasks.
Here are a few examples for your reference:
- Provided a scientific claim as query, retrieve documents that help verify or refute the claim.
- Search for documents that answers a FAQ-style query on children's nutrition.

Please adhere to the following guidelines:
- Specify what the query is, and what the des
- Each retrieval task should cover a wide ran

Your output should always be a python list of s
corresponds to a distinct retrieval task in one s
creative!

["Retrieve company's financial reports fo
"Given a book name as a query, retrieve
"Search for scientific research papers su
... *(omitted for space)*]

new session

You have been assigned a retrieval task: *{task}*
Your mission is to write one text retrieval example for this task in JSON format. The JSON object must contain the following keys:
- **"user_query"**: a string, a random user search query specified by the retrieval task.
- **"positive_document"**: a string, a relevant document for the user query.
- **"hard_negative_document"**: a string, a hard negative document that only appears relevant to the query.

Please adhere to the following guidelines:
- The "user_query" should be *{query_type}*, *{query_length}*, *{clarity}*, and diverse in topic.
- All documents should be at least *{num_words}* words long.
- Both the query and documents should be in *{language}*.
... *(omitted some for space)*

Your output must always be a JSON object only, do not explain yourself or output anything else. Be creative!

{**"user_query":** "How to use Microsoft Power BI for data analysis",
"positive_document": "Microsoft Power BI is a sophisticated tool that requires time and practice to master. In this tutorial, we'll show you how to navigate Power BI ... *(omitted)* ",
"hard_negative_document": "Excel is an incredibly powerful tool for managing and analyzing large amounts of data. Our tutorial series focuses on how you...*(omitted)*" }

Finetune Mistral7B

- Append an [EOS] token to the end of the query and document, and then feed them into the LLM to obtain the query and document embeddings.
- Contrastive loss

$$\min \quad \mathbb{L} = -\log \frac{\phi(q^+_{\text{inst}}, d^+)}{\phi(q^+_{\text{inst}}, d^+) + \sum_{n_i \in \mathbb{N}} (\phi(q^+_{\text{inst}}, n_i))}$$

Experimental Results: Data

- Generate 500k examples with 150k unique instructions using Azure OpenAI Service 2, among which 25% are generated by *GPT-35-Turbo* and others are generated by *GPT-4*. The total token consumption is about 180M.

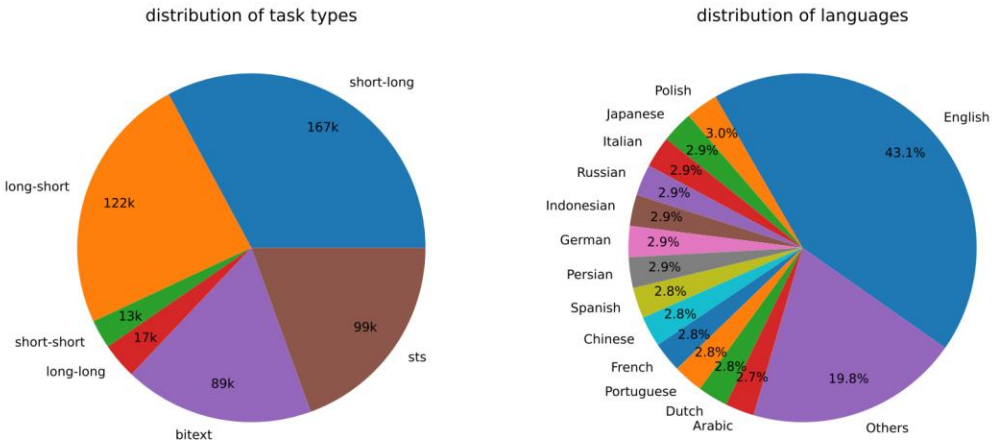


Figure 2: Task type and language statistics of the generated synthetic data (see Section 3.1 for task type definitions). The "Others" category contains the remaining languages from the XLM-R language list.

Model finetuning and Evaluation

The pretrained Mistral-7b [19] checkpoint is fine-tuned for 1 epoch using the loss in Equation 2. We follow the training recipe from RankLLaMA [24] and utilize LoRA [17] with rank 16. To further reduce GPU memory requirement, techniques including gradient checkpointing, mixed precision training, and DeepSpeed ZeRO-3 are applied.

For the training data, we utilize both the generated synthetic data and a collection of 13 public datasets, yielding approximately 1.8M examples after sampling. More details are available in Appendix A. To provide a fair comparison with some previous work, we also report results when the only labeled supervision is the MS-MARCO passage ranking [5] dataset.

We evaluate the trained model on the MTEB benchmark [28]. Note that the retrieval category in MTEB corresponds to the 15 publicly available datasets in the BEIR benchmark [42]. Evaluation of one model takes about 3 days on 8 V100 GPUs due to the need to encode a large number of documents. Although our model can accommodate sequence length beyond 512, we only evaluate on the first 512 tokens for efficiency. Official metrics are reported for each category. For more details about the evaluation protocol, please refer to the original papers [28, 42].

Results

# of datasets →	Class. 12	Clust. 11	PairClass. 3	Rerank 4	Retr. 15	STS 10	Summ. 1	Avg 56
Unsupervised Models								
Glove (Pennington et al., 2014)	57.3	27.7	70.9	43.3	21.6	61.9	28.9	42.0
$\text{SimCSE}_{\text{bert-unsup}}$ (Gao et al., 2021)	62.5	29.0	70.3	46.5	20.3	74.3	31.2	45.5
Supervised Models								
$\text{SimCSE}_{\text{bert-sup}}$ (Gao et al., 2021)	67.3	33.4	73.7	47.5	21.8	79.1	23.3	48.7
Contriever (Izacard et al., 2021)	66.7	41.1	82.5	53.1	41.9	76.5	30.4	56.0
GTR_{xxl} (Ni et al., 2022b)	67.4	42.4	86.1	56.7	48.5	78.4	30.6	59.0
$\text{Sentence-T5}_{\text{xxl}}$ (Ni et al., 2022a)	73.4	43.7	85.1	56.4	42.2	82.6	30.1	59.5
$\text{E5}_{\text{large-v2}}$ (Wang et al., 2022b)	75.2	44.5	86.0	56.6	50.6	82.1	30.2	62.3
$\text{GTE}_{\text{large}}$ (Li et al., 2023)	73.3	46.8	85.0	59.1	52.2	83.4	31.7	63.1
$\text{BGE}_{\text{large-en-v1.5}}$ (Xiao et al., 2023)	76.0	46.1	87.1	60.0	54.3	83.1	31.6	64.2
Ours								
$\text{E5}_{\text{mistral-7b}}$ + full data	**78.5**	50.3	**88.3**	**60.2**	**56.9**	**84.6**	31.4	**66.6**
w/ synthetic data only	78.2	**50.5**	86.0	59.0	46.9	81.2	31.9	63.1
w/ synthetic + msmarco	78.3	49.9	87.1	59.5	52.2	81.2	**32.7**	64.5

Table 1: Results on the MTEB benchmark (Muennighoff et al., 2023) (56 datasets in the English subset). The numbers are averaged for each category. Please refer to Table 17 for the scores per dataset.

Results

Table 2: Comparison with commercial models and the model that tops the MTEB leaderboard (as of 2023-12-22). For the commercial models listed here, little details are available on their model architectures and training data.

Model	BEIR	MTEB
OpenAI text-embedding-3-large	55.4	64.6
Cohere-embed-english-v3.0	55.0	64.5
voyage-lite-01-instruct	55.6	64.5
UAE-Large-V1	54.7	64.6
$E5_{\text{mistral-7b}}$ + full data	**56.9**	**66.6**

5.1 Is Contrastive Pre-training Necessary?

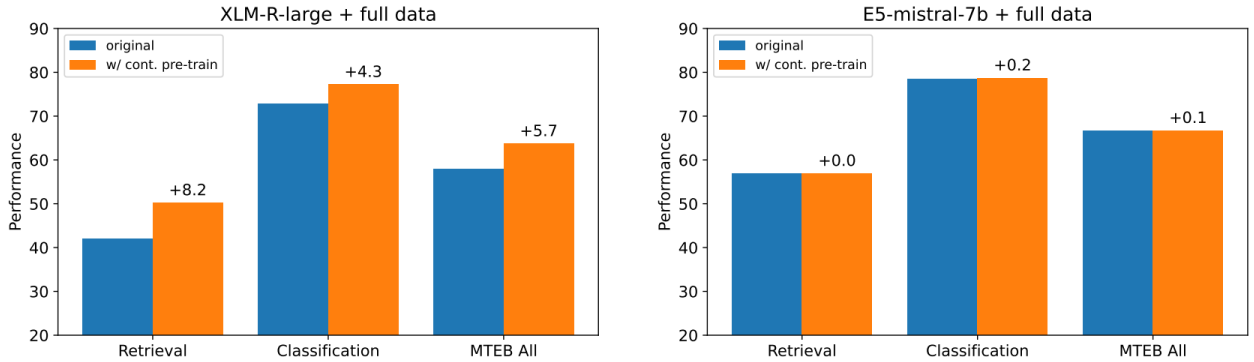
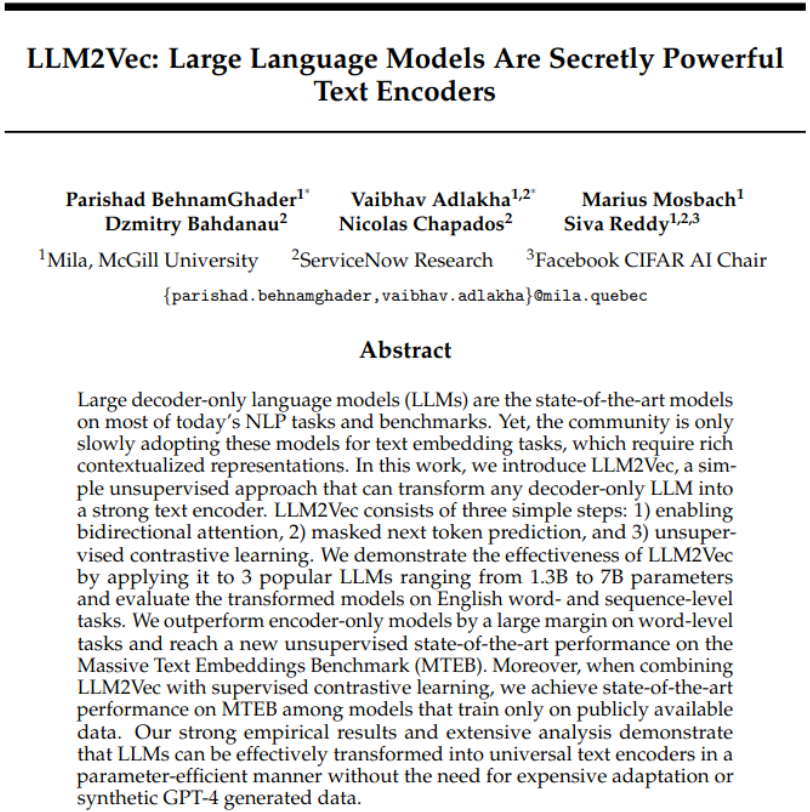


Figure 3: Effects of contrastive pre-training. Detailed numbers are in Appendix Table 6.

2.4. LLM2VEC

- Contributions:
  - A simple approach to convert large language models into text embedding models
  - Enable bidirectional attention in LLMs
  - Propose next token prediction objective to help model learn bidirectional attention



LLM2Vec: Large Language Models Are Secretly Powerful Text Encoders

Parishad BehnamGhader[1*] Vaibhav Adlakha[1,2*] Marius Mosbach[1]
Dzmitry Bahdanau[2] Nicolas Chapados[2] Siva Reddy[1,2,3]

[1]Mila, McGill University [2]ServiceNow Research [3]Facebook CIFAR AI Chair

{parishad.behnamghader,vaibhav.adlakha}@mila.quebec

Abstract

Large decoder-only language models (LLMs) are the state-of-the-art models on most of today's NLP tasks and benchmarks. Yet, the community is only slowly adopting these models for text embedding tasks, which require rich contextualized representations. In this work, we introduce LLM2Vec, a simple unsupervised approach that can transform any decoder-only LLM into a strong text encoder. LLM2Vec consists of three simple steps: 1) enabling bidirectional attention, 2) masked next token prediction, and 3) unsupervised contrastive learning. We demonstrate the effectiveness of LLM2Vec by applying it to 3 popular LLMs ranging from 1.3B to 7B parameters and evaluate the transformed models on English word- and sequence-level tasks. We outperform encoder-only models by a large margin on word-level tasks and reach a new unsupervised state-of-the-art performance on the Massive Text Embeddings Benchmark (MTEB). Moreover, when combining LLM2Vec with supervised contrastive learning, we achieve state-of-the-art performance on MTEB among models that train only on publicly available data. Our strong empirical results and extensive analysis demonstrate that LLMs can be effectively transformed into universal text encoders in a parameter-efficient manner without the need for expensive adaptation or synthetic GPT-4 generated data.

BehnamGhader, P., Adlakha, V., Mosbach, M., Bahdanau, D., Chapados, N., & Reddy, S. (2024). Llm2vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961*.

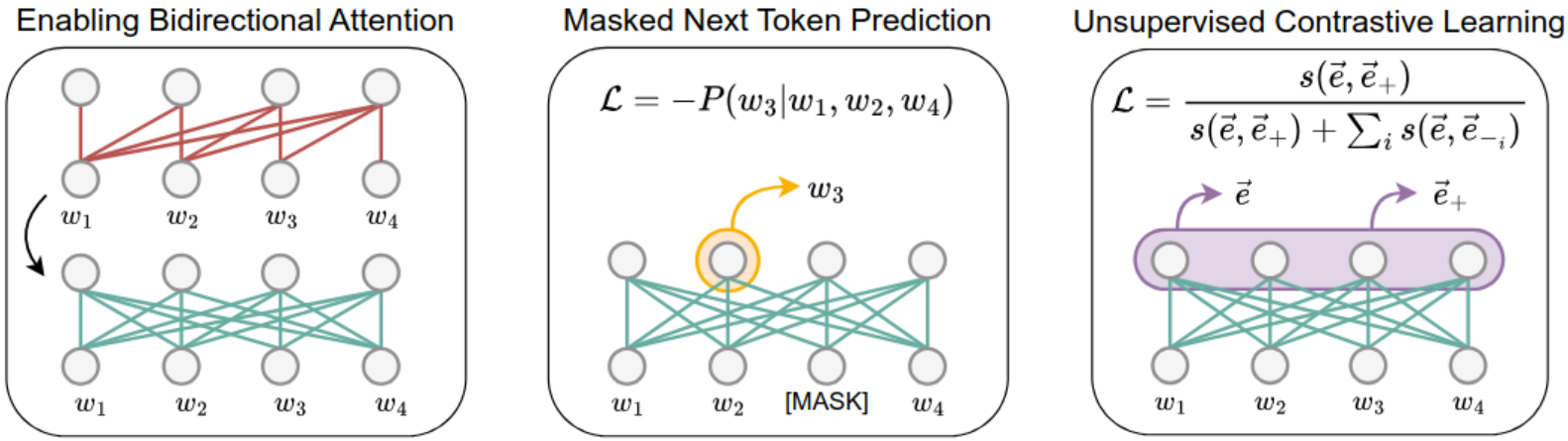
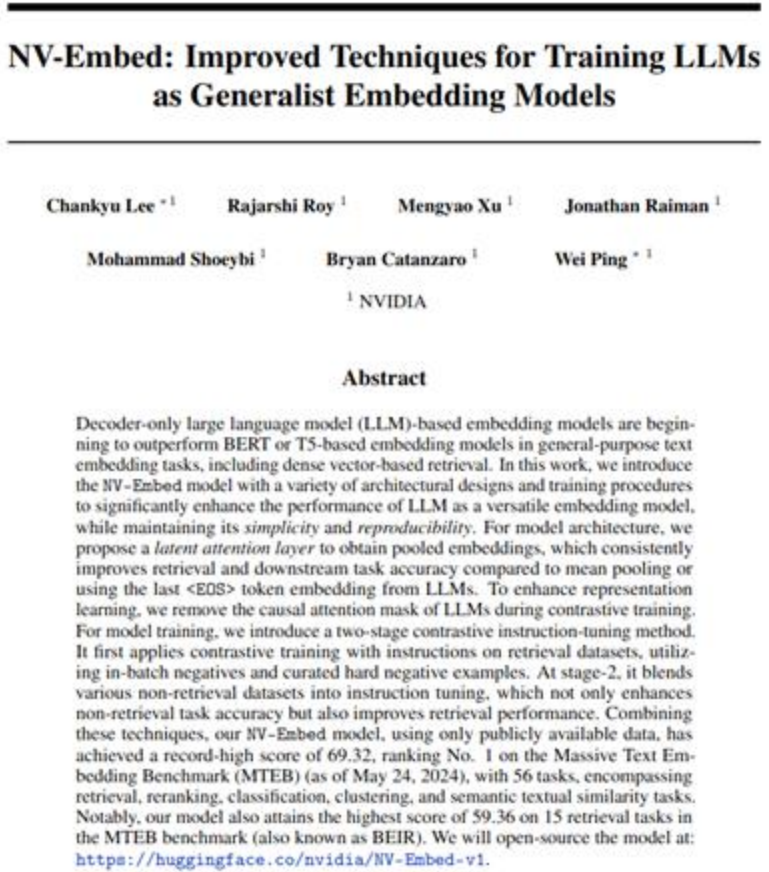


Figure 1: The 3 steps of LLM2Vec. First, we enable bidirectional attention to overcome the restrictions of causal attention **(Bi)**. Second, we adapt the model to use bidirectional attention by masked next token prediction training **(MNTP)**. Third, we apply unsupervised contrastive learning with mean pooling to learn better sequence representations **(SimCSE)**.

NV-Embed

Contributions:

- Propose a new pooling strategy
- Two-stage contrastive instruction-tuning for retrieval and non-retrieval tasks



NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models

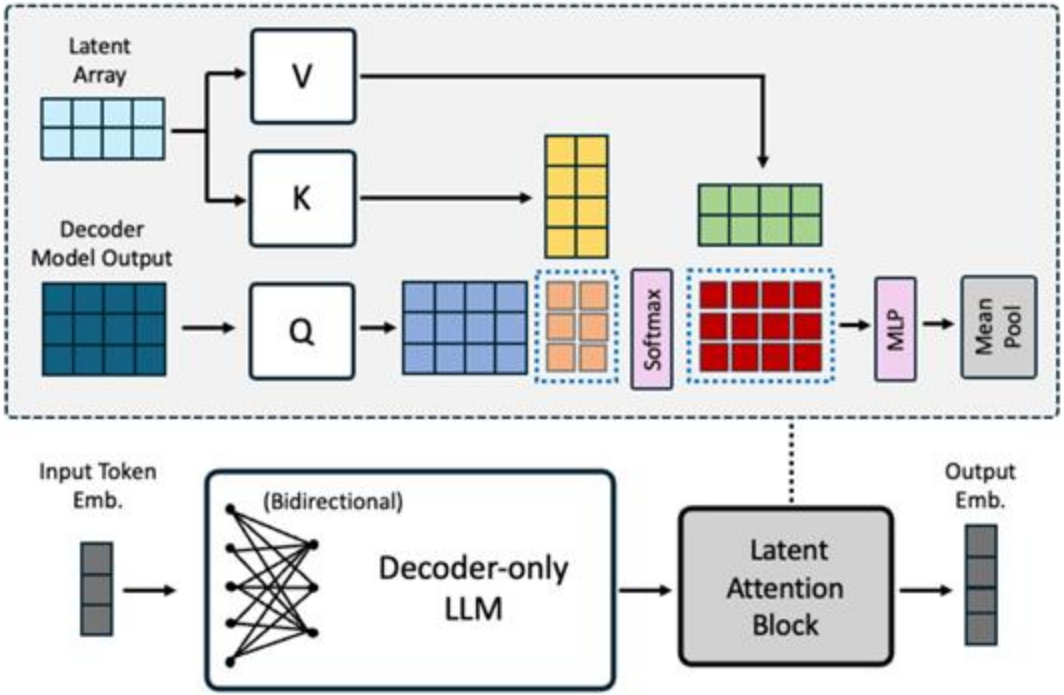
Chankyu Lee [*1] Rajarshi Roy [1] Mengyao Xu [1] Jonathan Raiman [1]
Mohammad Shoeybi [1] Bryan Catanzaro [1] Wei Ping [*1]
[1] NVIDIA

Abstract

Decoder-only large language model (LLM)-based embedding models are beginning to outperform BERT or T5-based embedding models in general-purpose text embedding tasks, including dense vector-based retrieval. In this work, we introduce the NV-Embed model with a variety of architectural designs and training procedures to significantly enhance the performance of LLM as a versatile embedding model, while maintaining its *simplicity* and *reproducibility*. For model architecture, we propose a *latent attention layer* to obtain pooled embeddings, which consistently improves retrieval and downstream task accuracy compared to mean pooling or using the last <EOS> token embedding from LLMs. To enhance representation learning, we remove the causal attention mask of LLMs during contrastive training. For model training, we introduce a two-stage contrastive instruction-tuning method. It first applies contrastive training with instructions on retrieval datasets, utilizing in-batch negatives and curated hard negative examples. At stage-2, it blends various non-retrieval datasets into instruction tuning, which not only enhances non-retrieval task accuracy but also improves retrieval performance. Combining these techniques, our NV-Embed model, using only publicly available data, has achieved a record-high score of 69.32, ranking No. 1 on the Massive Text Embedding Benchmark (MTEB) (as of May 24, 2024), with 56 tasks, encompassing retrieval, reranking, classification, clustering, and semantic textual similarity tasks. Notably, our model also attains the highest score of 59.36 on 15 retrieval tasks in the MTEB benchmark (also known as BEIR). We will open-source the model at: https://huggingface.co/nvidia/NV-Embed-v1.

Lee, C., Roy, R., Xu, M., Raiman, J., Shoeybi, M., Catanzaro, B., & Ping, W. (2024). NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models. *arXiv preprint arXiv:2405.17428*.

NV-Embed



- Final Representations attend into a “learnable dictionary” before mean pooling
  ⇒ sparse dictionary learning
  ⇒ improve performance on retrieval tasks

NV-Embed

Two-stage tuning:

1. Train retrieval data with in-batch trick

2. Train both retrieval data and non-retrieval data without in-batch trick

⇒ In-batch trick is helpful for retrieval tasks but harmful for non-retrieval such as clustering or classification

⇒ In the second stage, the negative samples are selected before instead of exploiting in-batch samples

Experimental results

Table 1: Top MTEB leaderboard models as of 2024-05-22. We use the original model names on the leaderboard for clarity.

Embedding Task Mertric	Retrieval (15) nDCG@10	Rerank (4) MAP	Cluter. (11) V-Meas.	PairClass. (3) AP	Class. (12) Acc.	STS (10) Spear.	Summ.(1) Spear.	Avg. (56)
`NV-Embed`	**59.36**	60.59	52.80	86.91	87.35	82.84	31.2	**69.32**
`NV-Embed` (mean pool)	58.71	60.75	52.80	85.85	87.06	82.53	30.49	68.98
Voyage-large-2-instruct	58.28	60.09	53.35	89.24	81.49	84.58	30.84	68.28
SFR-Embedding	59.00	60.64	51.67	88.54	78.33	85.05	31.16	67.56
Gte-Qwen1.5-7B-instruct	56.24	60.13	55.83	87.38	79.6	82.42	31.46	67.34
Voyage-lite-02-instruct	56.6	58.24	52.42	86.87	79.25	85.79	31.01	67.13
GritLM-7B	57.41	60.49	50.61	87.16	79.46	83.35	30.37	66.76
E5-mistral-7b-instruct	56.9	60.21	50.26	88.34	78.47	84.66	31.4	66.63
Google-gecko	55.7	58.9	47.48	87.61	81.17	85.07	32.63	66.31
LLM2Vec-Meta-Llama-3	56.63	59.69	46.45	87.79	75.92	83.58	30.94	65.01
Text-embed-3-large (OpenAI)	55.44	59.16	49.01	85.72	75.45	81.73	29.92	64.59

Leaderboard

- **MTEB: Massive Text Embedding Benchmark**
- **Leaderboard: https://huggingface.co/spaces/mteb/leaderboard**

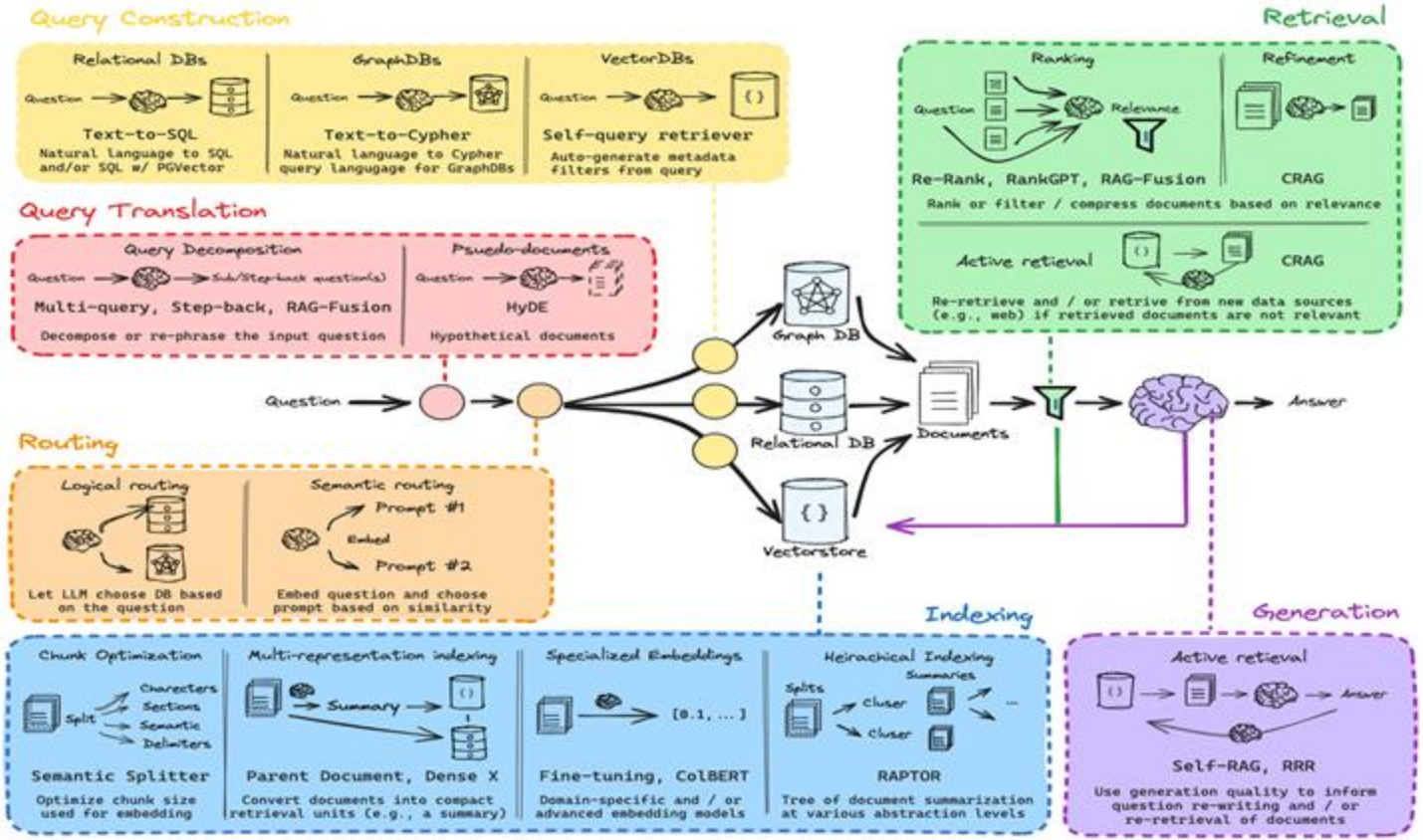
Leaderboard: MTEB

Rank	Model	Model Size (Million Parameters)	Memory Usage (GB, fp32)	Embedding Dimensions	Max Tokens	Average (56 datasets)	Classification Average (12 datasets)	Clustering Average (11 datasets)
1	bge-en-icl	7111	26.49	4096	32768	71.67	88.95	57.89
2	stella_en_1.5B_v5	1543	5.75	8192	131072	71.19	87.63	57.69
3	SFR-Embedding-2_R	7111	26.49	4096	32768	70.31	89.05	56.17
4	gte-Qwen2-7B-instruct	7613	28.36	3584	131072	70.24	86.58	56.92
5	stella_en_400M_v5	435	1.62	8192	8192	70.11	86.67	56.7
6	bge-multilingual-gemma2	9242	34.43	3584	8192	69.88	88.08	54.65
7	NV-Embed-v1	7851	29.25	4096	32768	69.32	87.35	52.8
8	voyage-large-2-instruct			1024	16000	68.23	81.49	53.35
9	Linq-Embed-Mistral	7111	26.49	4096	32768	68.17	80.2	51.42
10	SFR-Embedding-Mistral	7111	26.49	4096	32768	67.56	78.33	51.67

Leaderboard: MTEB

Rank	Model	Model Size (Million Parameters)	Memory Usage (GB, fp32)	Embedding Dimensions	Max Tokens	Average (56 datasets)	Classification Average (12 datasets)	Clustering Average (11 datasets)
16	GritLM-7B	7242	26.98	4096	32768	66.76	79.46	50.61
17	e5-mistral-7b-instruct	7111	26.49	4096	32768	66.63	78.47	50.26
18	google-gecko.text-embedding-p	1200	4.47	768	2048	66.31	81.17	47.48
19	TDTE					65.96	77.17	47.86
20	GritLM-8x7B	46703	173.98	4096	32768	65.66	78.53	50.14
21	gte-large-en-v1.5	434	1.62	1024	8192	65.39	77.75	47.96
22	LLM2Vec-Meta-Llama-3-supervis	7505	27.96	4096	8192	65.01	75.92	46.45
23	LLM2Vec-Mistral-supervised	7111	26.49	4096	32768	64.8	76.63	45.54
24	echo-mistral-7b-instruct-last	7111	26.49	4096	32768	64.68	77.43	46.32

3. RAG advanced techniques



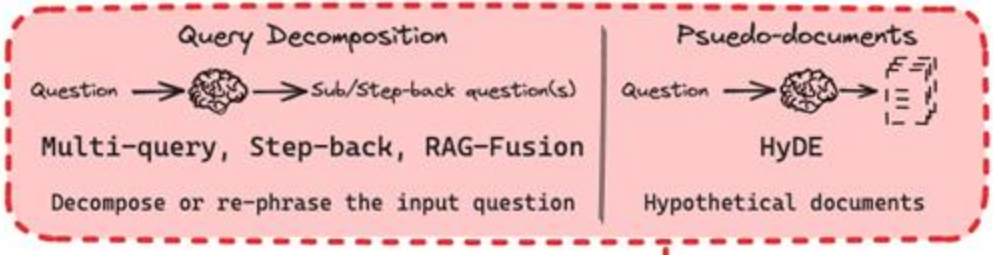
https://github.com/langchain-ai/rag-from-scratch/blob/main/rag_from_scratch_1_to_4.ipynb

Query Translation

- Distance-based retrieval can be sensitive to query wording and imperfect embedding
- Current Approach: Manual prompt tuning ⇒ Tedious

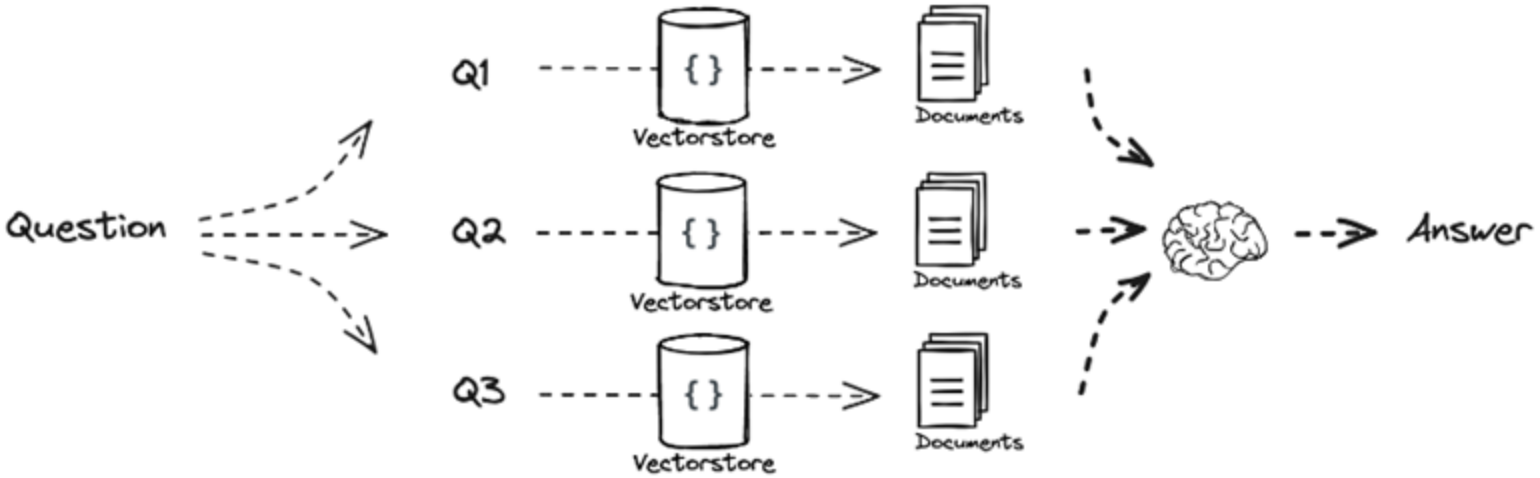
⇒ Automates prompt tuning with LLMs

- Approaches:
  - Multi-query
  - RAG-Fusion
  - Step-back
  - HyDE
  - …



Multi-Query

Generate different questions from the original one to retrieve more diverse documents



https://github.com/langchain-ai/rag-from-scratch/blob/main/rag_from_scratch_5_to_9.ipynb

Multi-Query

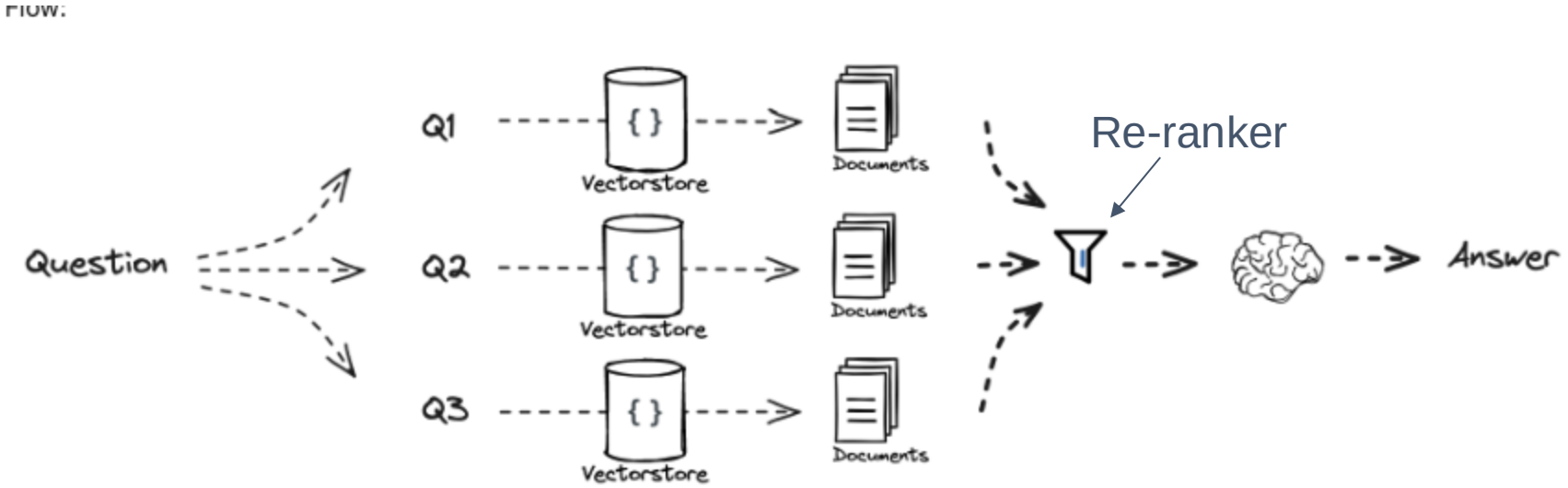
Concatenate retrieved documents into a context, and ask the LLM with the original question

```
# Multi Query: Different Perspectives
template = """You are an AI language model assistant. Your task is to generate five
different versions of the given user question to retrieve relevant documents from a vector
database. By generating multiple perspectives on the user question, your goal is to help
the user overcome some of the limitations of the distance-based similarity search.
Provide these alternative questions separated by newlines. Original question: {question}"""
prompt_perspectives = ChatPromptTemplate.from_template(template)
```

```
# RAG
template = """Answer the following question based on this context:

{context}

Question: {question}
"""
```

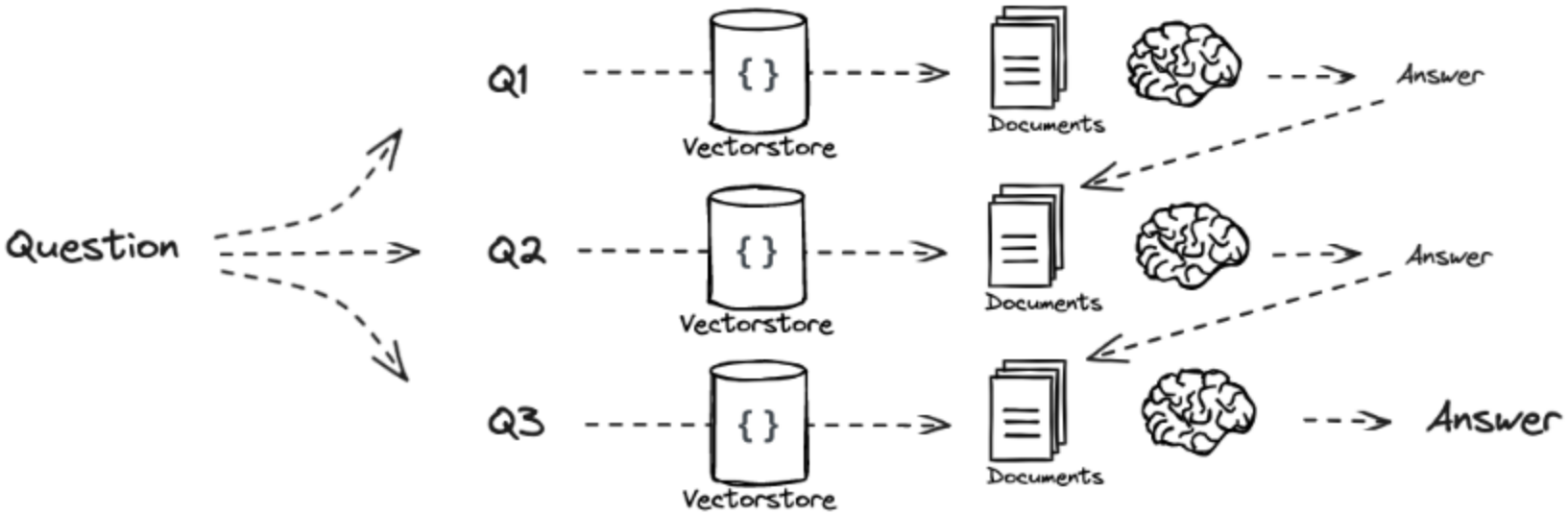
RAG-Fusion

Different from Multi-query, RAG-Fusion add a Re-ranker to put the most relevant documents into the beginning of the context ⇒ Mitigating “Lost-in-the-middle” problem.



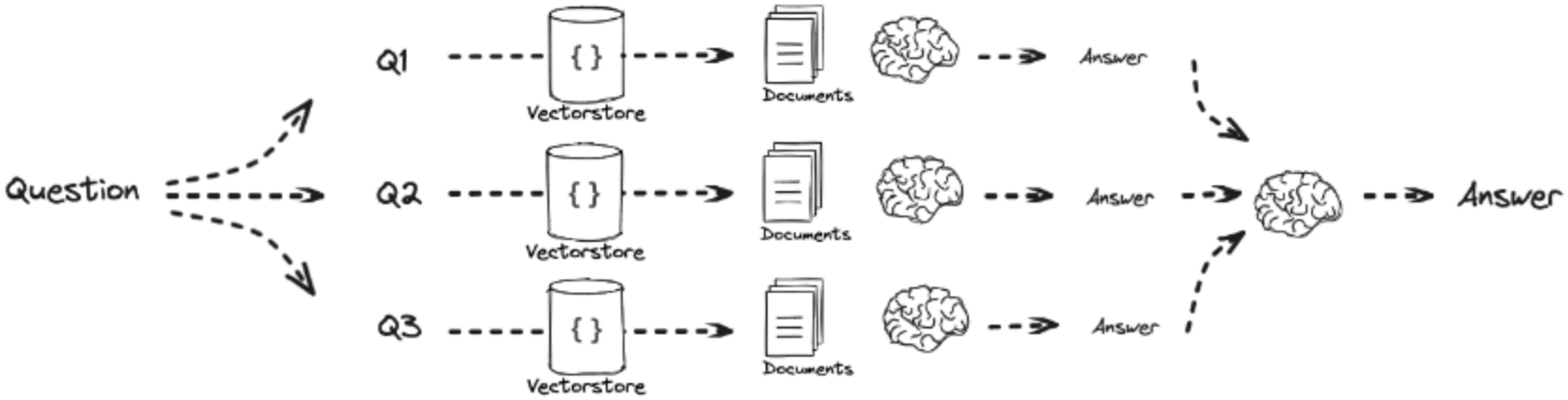
Recursive Query Decomposition

```
# Decomposition
template = """You are a helpful assistant that generates multiple sub-questions related to an input question. \n
The goal is to break down the input into a set of sub-problems / sub-questions that can be answers in isolation.
Generate multiple search queries related to: {question} \n
Output (3 queries):"""
prompt_decomposition = ChatPromptTemplate.from_template(template)
```



https://github.com/langchain-ai/rag-from-scratch/blob/main/rag_from_scratch_5_to_9.ipynb

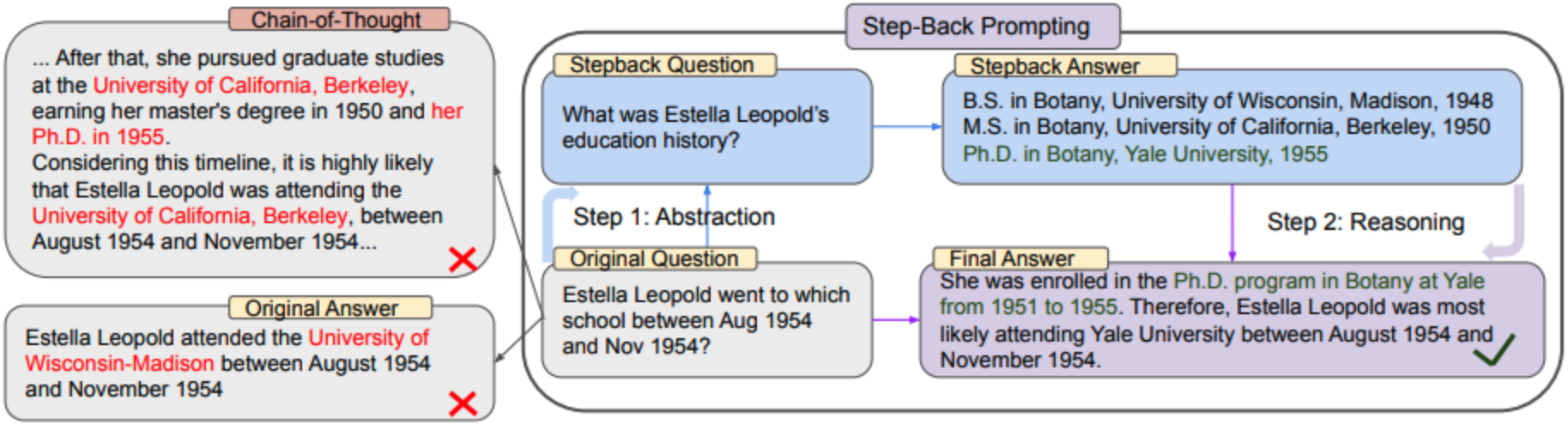
Individual Query Decomposition



Ask sub-questions separately and concatenate all question and answer pairs into context then ask LLM to synthesize the answer for the original question.

Step-Back

Prompt = """You are an expert at world knowledge. Your task is to step back and paraphrase a question to a more generic step-back question, which is easier to answer. Here are a few examples:""",



https://arxiv.org/pdf/2310.06117.pdf

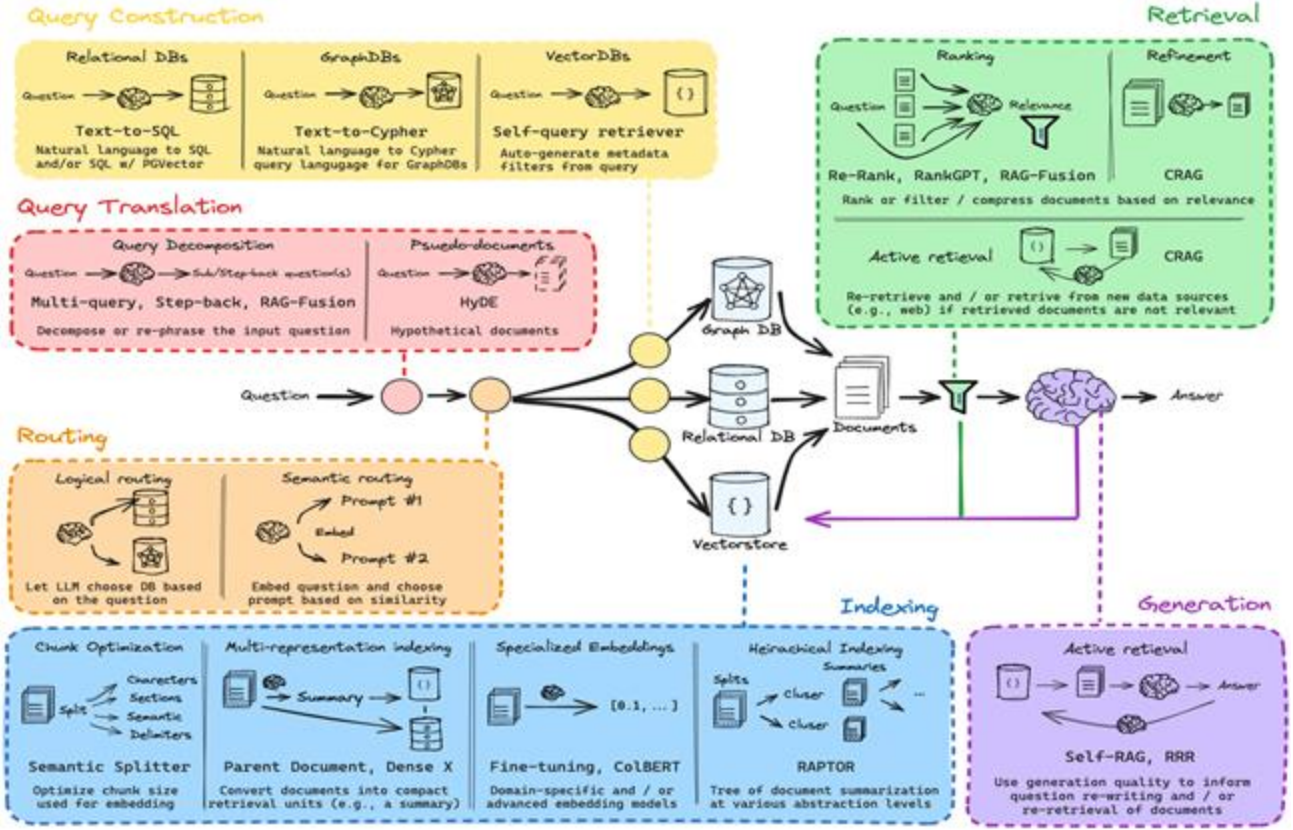
HyDE

Generates hypothetical answers to queries, embeds them for better retrieval.

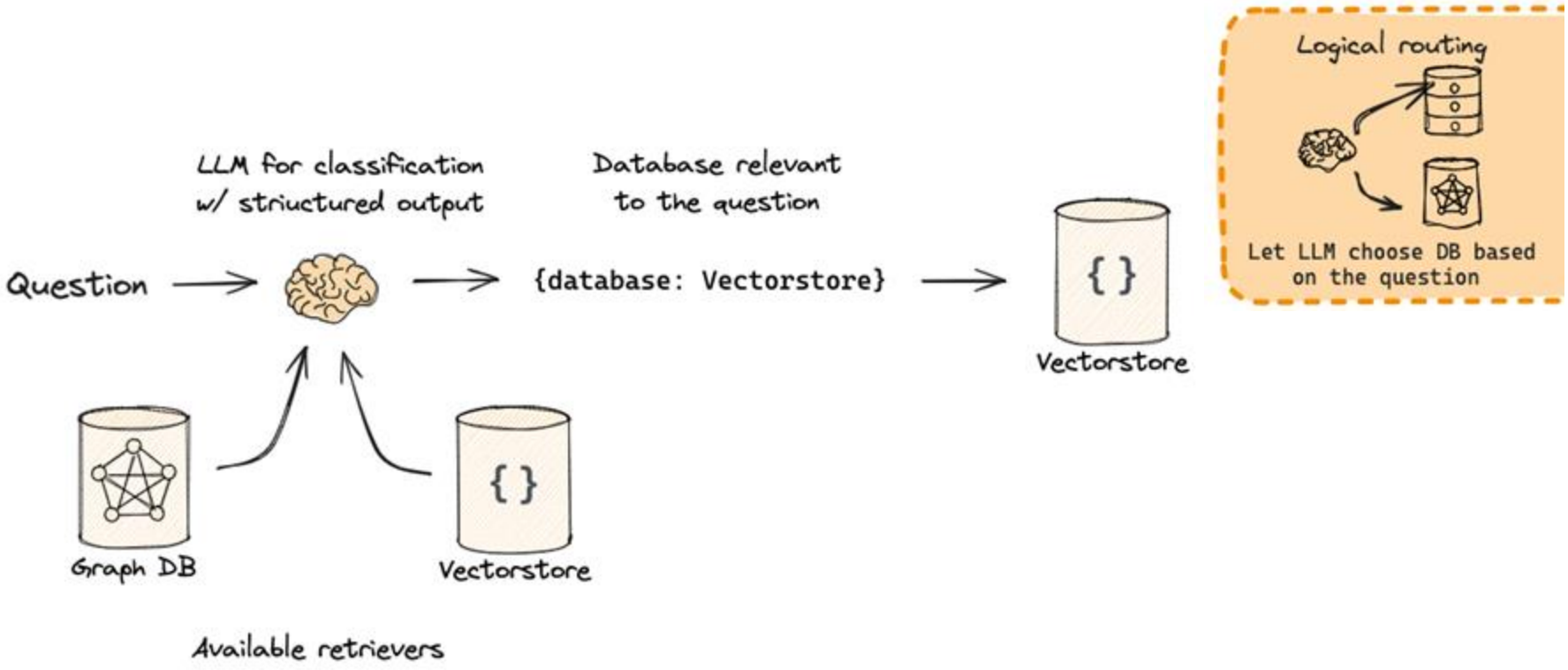
```
# HyDE document genration
template = """Please write a scientific paper passage to answer the question
Question: {question}
Passage:"""
prompt_hyde = ChatPromptTemplate.from_template(template)
```

https://github.com/langchain-ai/langchain/blob/master/cookbook/hypothetical_document_embeddings.ipynb

Routing

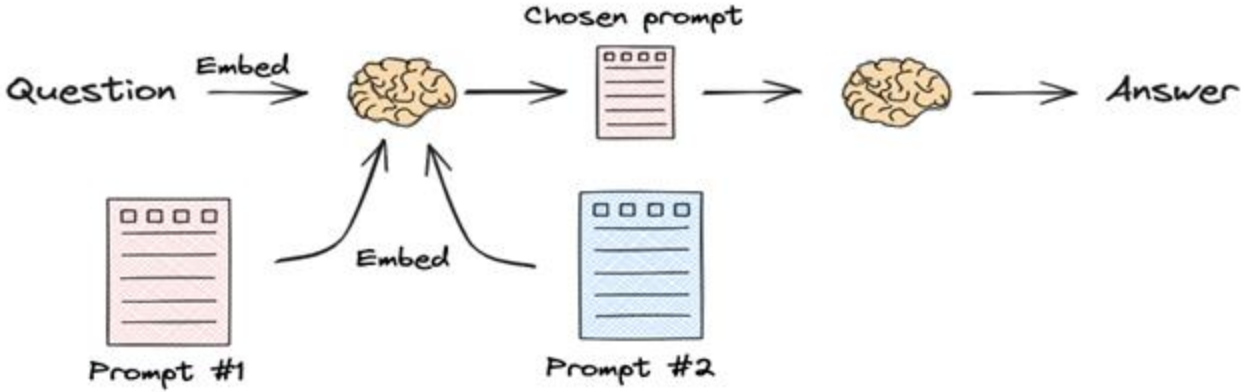

Query Construction
Relational DBs
Question
Text-to-SQL
Natural language to SQL and/or SQL w/ PGVector
GraphDBs
Question
Text-to-Cypher
Natural language to Cypher query language for GraphDBs
VectorDBs
Question
Self-query retriever
Auto-generate metadata filters from query
Retrieval
Ranking
Question
Relevance
Re-Rank, RankGPT, RAG-Fusion
Refinement
CRAG
Rank or filter / compress documents based on relevance
Active retrieval
CRAG
Re-retrieve and / or retrive from new data sources (e.g., web) if retrieved documents are not relevant
Query Translation
Query Decomposition
Question
Sub/Step-back question(s)
Multi-query, Step-back, RAG-Fusion
Decompose or re-phrase the input question
Psuedo-documents
Question
HyDE
Hypothetical documents
Graph DB
Question
Relational DB
Documents
Answer
Vectorstore
Routing
Logical routing
Let LLM choose DB based on the question
Semantic routing
Prompt #1
Embed
Prompt #2
Embed question and choose prompt based on similarity
Indexing
Chunk Optimization
Split
Characters
Sections
Semantic
Delimiters
Semantic Splitter
Optimize chunk size used for embedding
Multi-representation indexing
Summary
Parent Document, Dense X
Convert documents into compact retrieval units (e.g., a summary)
Specialized Embeddings
[0.1, ...]
Fine-tuning, ColBERT
Domain-specific and / or advanced embedding models
Heirachical Indexing
Splits
Summaries
Cluster
Cluster
RAPTOR
Tree of document summarization at various abstraction levels
Generation
Active retrieval
Answer
Self-RAG, RRR
Use generation quality to inform question re-writing and / or re-retrieval of documents

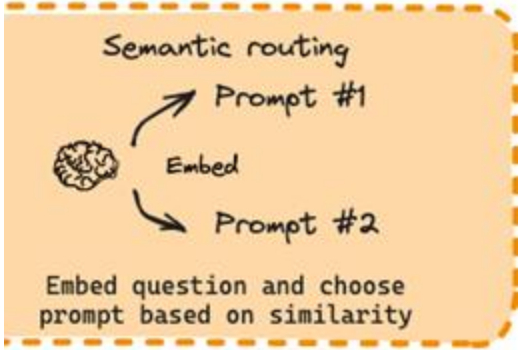
Routing



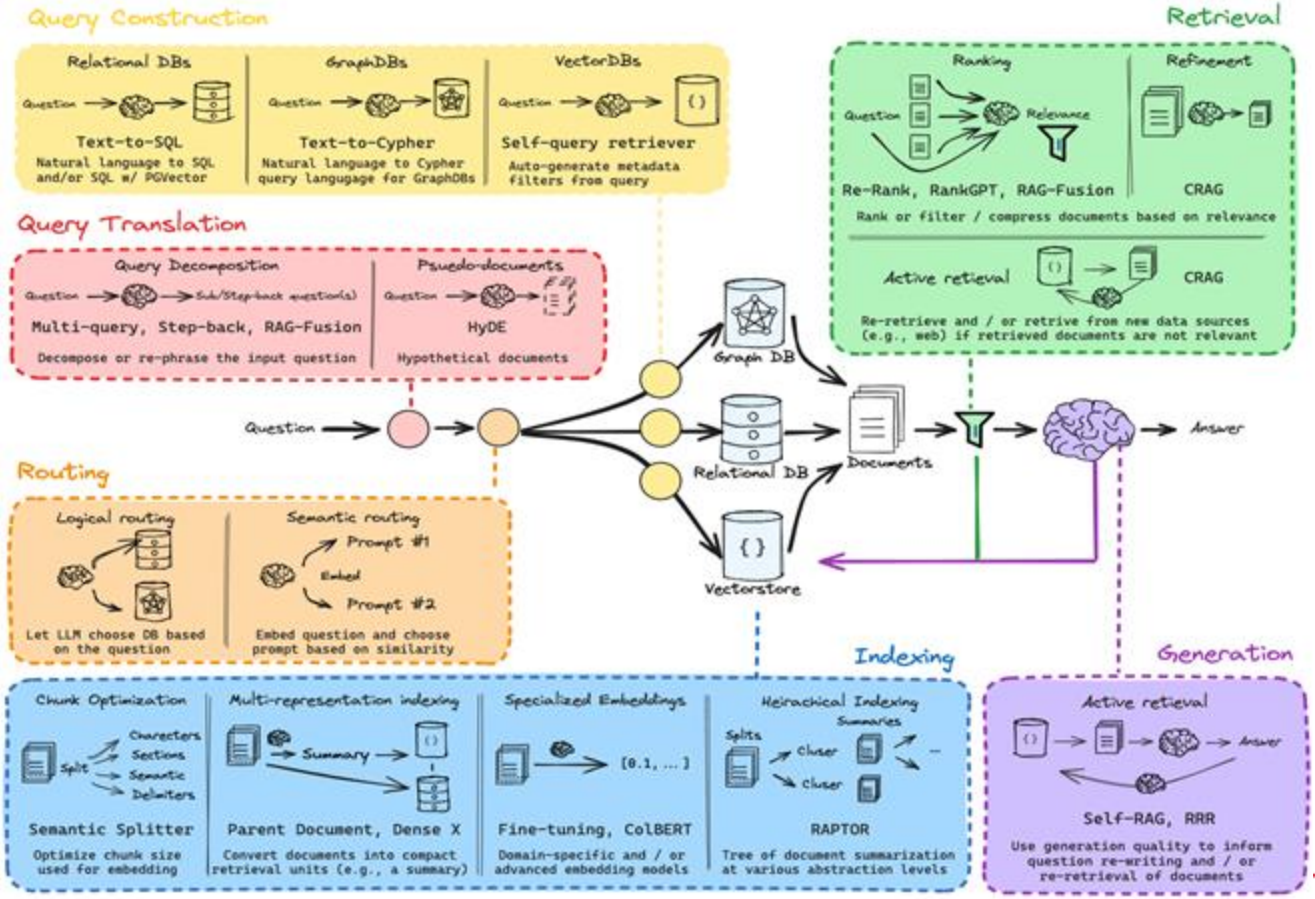
https://github.com/langchain-ai/rag-from-scratch/blob/main/rag_from_scratch_10_and_11.ipynb

Routing





https://github.com/langchain-ai/rag-from-scratch/blob/main/rag_from_scratch_10_and_11.ipynb

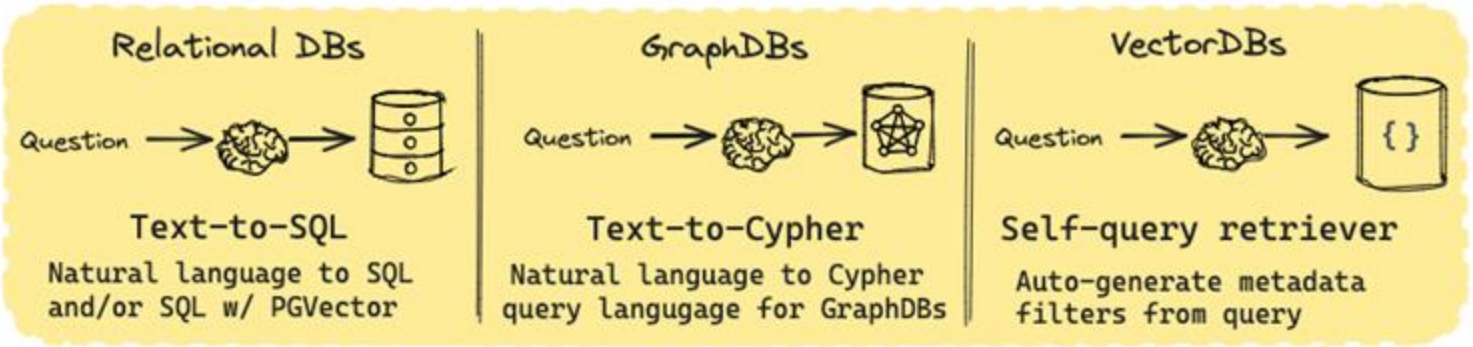

Query Construction
Relational DBs
Question
Text-to-SQL
Natural language to SQL and/or SQL w/ PGVector
GraphDBs
Question
Text-to-Cypher
Natural language to Cypher query langugage for GraphDBs
VectorDBs
Question
{}
Self-query retriever
Auto-generate metadata filters from query
Retrieval
Ranking
Question
Relevance
Refinement
Re-Rank, RankGPT, RAG-Fusion
CRAG
Rank or filter / compress documents based on relevance
Active retrieval
{}
CRAG
Re-retrieve and / or retrive from new data sources (e.g., web) if retrieved documents are not relevant
Query Translation
Query Decomposition
Question
Sub/Step-back question(s)
Multi-query, Step-back, RAG-Fusion
Decompose or re-phrase the input question
Psuedo-documents
Question
HyDE
Hypothetical documents
Graph DB
Question
Relational DB
Documents
Answer
Vectorstore
{}
Routing
Logical routing
Let LLM choose DB based on the question
Semantic routing
Prompt #1
Embed
Prompt #2
Embed question and choose prompt based on similarity
Indexing
Generation
Chunk Optimization
Split
Characters
Sections
Semantic
Delimiters
Semantic Splitter
Optimize chunk size used for embedding
Multi-representation indexing
Summary
{}
Parent Document, Dense X
Convert documents into compact retrieval units (e.g., a summary)
Specialized Embeddings
[0.1, ...]
Fine-tuning, ColBERT
Domain-specific and / or advanced embedding models
Heirachical Indexing
Splits
Summaries
Cluster
Cluster
RAPTOR
Tree of document summarization at various abstraction levels
Active retrieval
{}
Answer
Self-RAG, RRR
Use generation quality to inform question re-writing and / or re-retrieval of documents

Query Construction

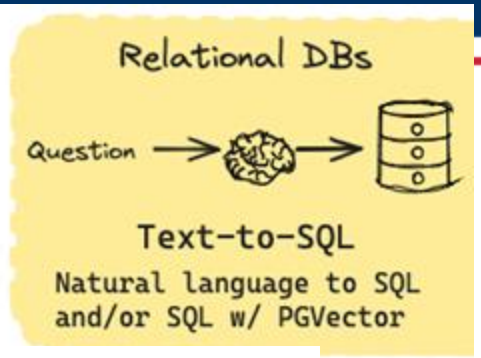
- Most data has some structure: SQL, Graph
- Previous use embedding to retrieve unstructured data

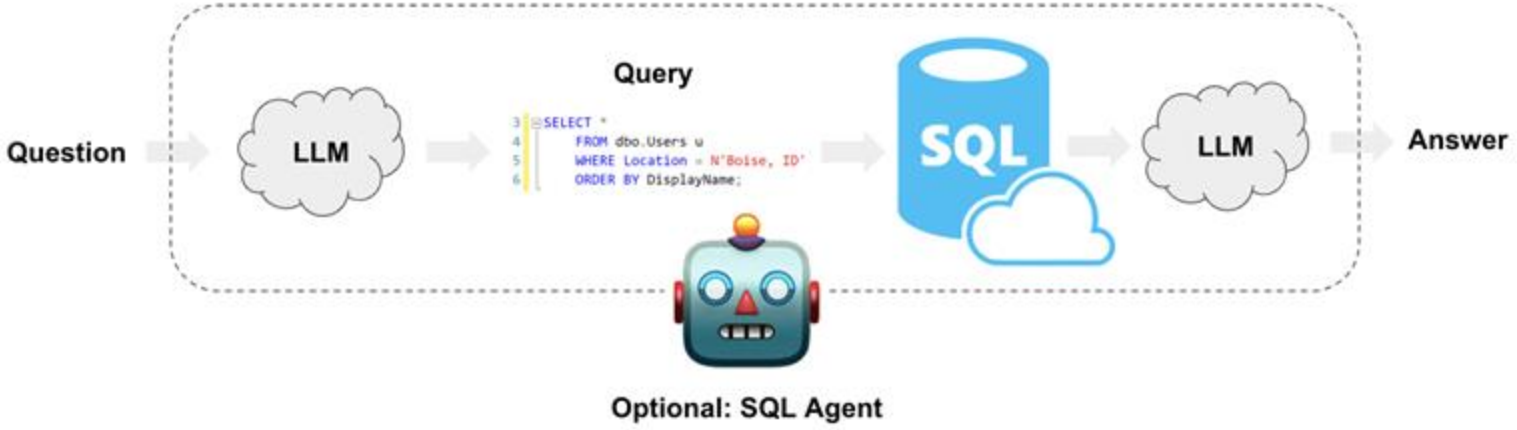
  ⇒ How's about structured data?

- Query Construction converts natural language into a specific query syntax

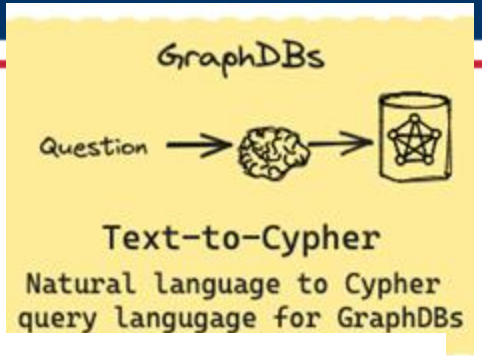


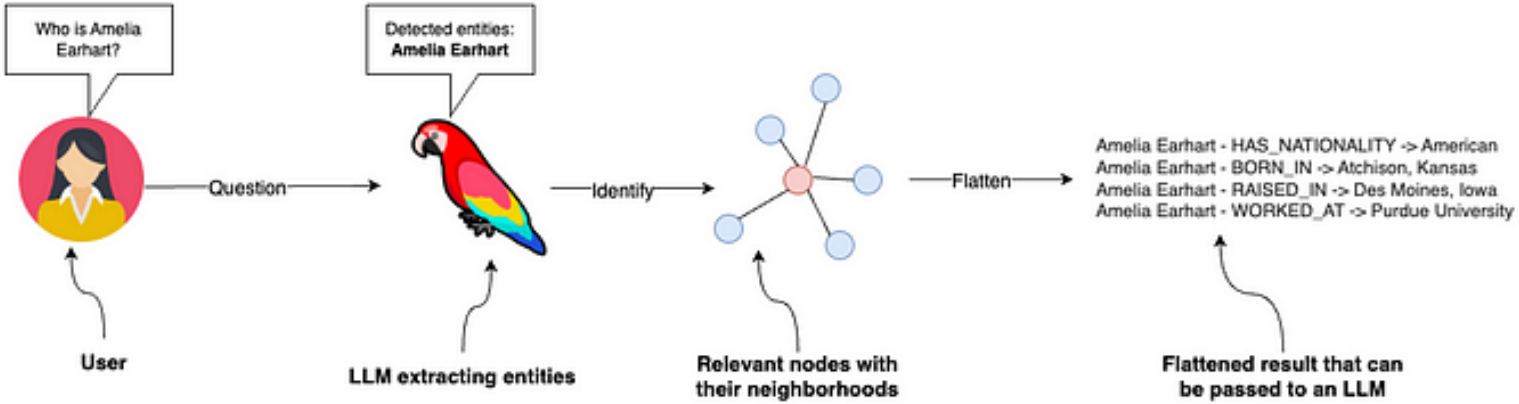
Query Construction

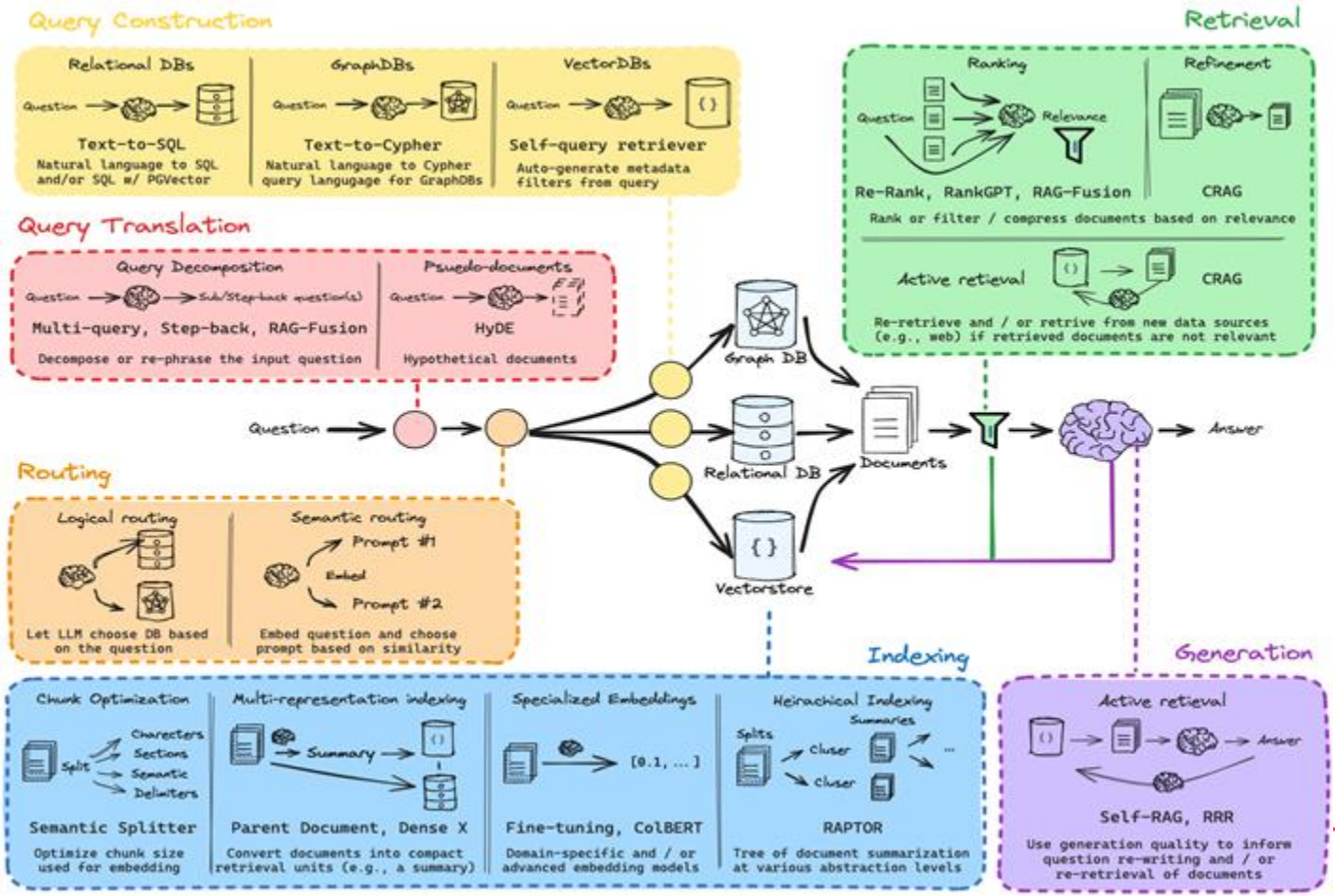




Query Construction

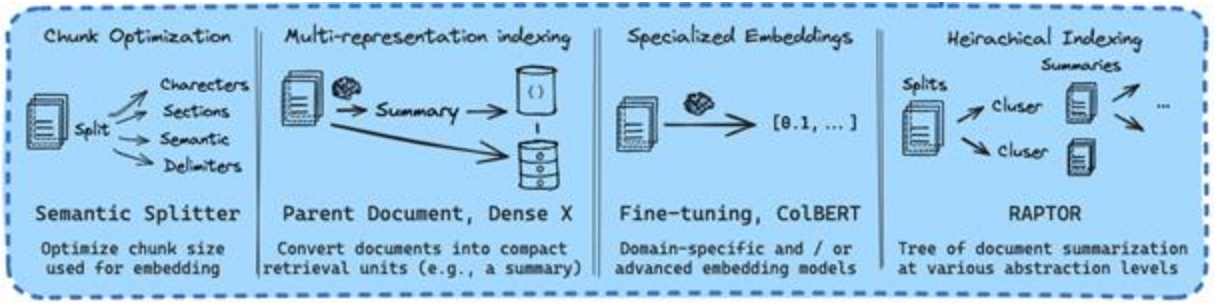


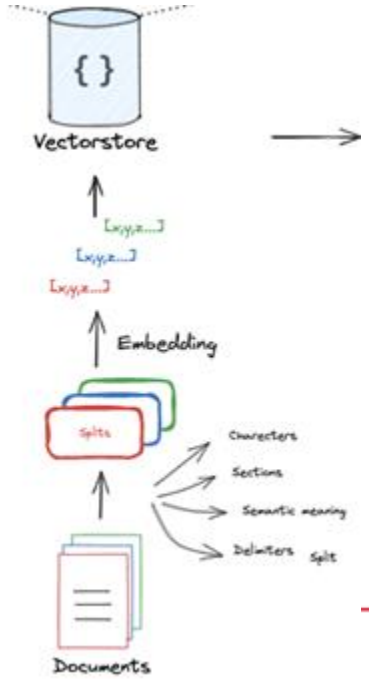



Query Construction
Relational DBs
Question
Text-to-SQL
Natural language to SQL and/or SQL w/ PGVector
GraphDBs
Question
Text-to-Cypher
Natural language to Cypher query langugage for GraphDBs
VectorDBs
Question
{ }
Self-query retriever
Auto-generate metadata filters from query
Retrieval
Ranking
Question
Relevance
Refinement
Re-Rank, RankGPT, RAG-Fusion
CRAG
Rank or filter / compress documents based on relevance
Active retieval
{ }
CRAG
Re-retrieve and / or retrive from new data sources (e.g., web) if retrieved documents are not relevant
Query Translation
Query Decomposition
Question
Sub/Step-back question(s)
Multi-query, Step-back, RAG-Fusion
Decompose or re-phrase the input question
Psuedo-documents
Question
HyDE
Hypothetical documents
Graph DB
Question
Relational DB
Documents
Answer
Vectorstore
Routing
Logical routing
Let LLM choose DB based on the question
Semantic routing
Prompt #1
Embed
Prompt #2
Embed question and choose prompt based on similarity
Indexing
Generation
Chunk Optimization
Split
Characters
Sections
Semantic
Delimiters
Semantic Splitter
Optimize chunk size used for embedding
Multi-representation indexing
Summary
{ }
Parent Document, Dense X
Convert documents into compact retrieval units (e.g., a summary)
Specialized Embeddings
[0.1, ...]
Fine-tuning, ColBERT
Domain-specific and / or advanced embedding models
Heirachical Indexing
Summaries
Splits
Cluser
Cluser
...
RAPTOR
Tree of document summarization at various abstraction levels
Active retieval
{ }
Answer
Self-RAG, RRR
Use generation quality to inform question re-writing and / or re-retrieval of documents

Indexing

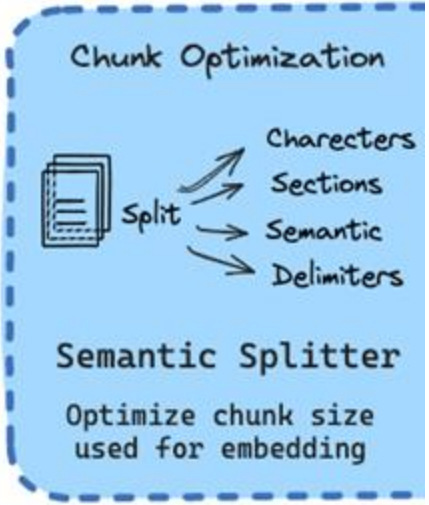
- Documents will be processed, segmented, transformed into Embeddings.
- The quality of index construction determines whether the correct context can be obtained in the retrieval phase.





Indexing

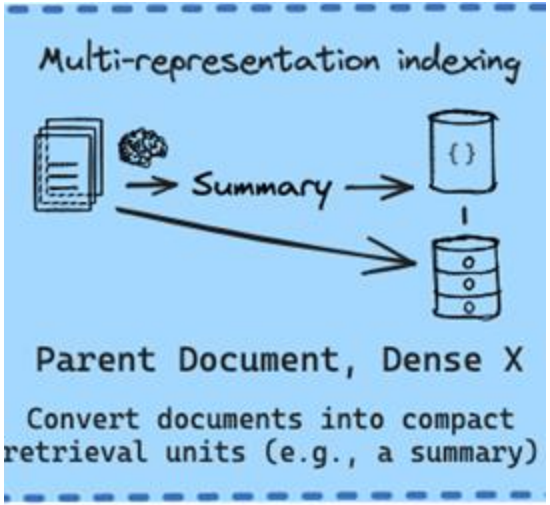
- **Balancing Context and Efficiency:**
  - **Fixed-Size Chunking:**
    - Common method (e.g., 100, 256, 512 tokens).
    - Larger chunks: More context, but more noise (longer processing, higher cost).
    - Smaller chunks: Less noise, but less context.
  - **Alternative Approaches:**
    - Recursive Splits & Sliding Windows: Layered retrieval with better context, but complex.
- **Challenge:** Finding the optimal balance between semantic completeness and context length.



Multi-Representation Indexing

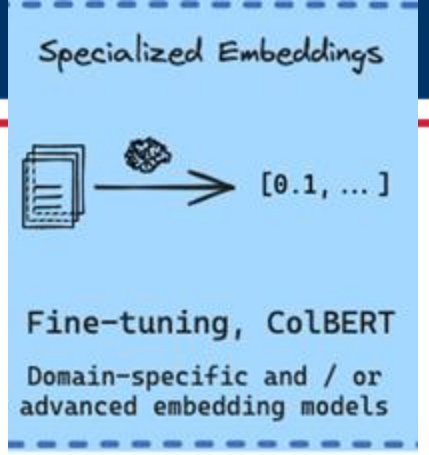
- **Challenge:** Balancing meaning (small chunks) and context (large documents) in retrieval.

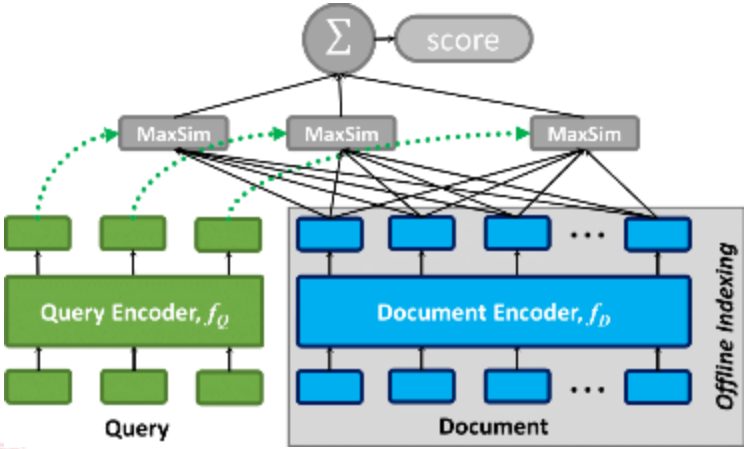
- **Parent Document Retriever:**
  - Splits documents into small chunks for accurate embeddings.
  - Stores parent document IDs for each chunk.
  - Retrieves relevant chunks during search.
  - Returns the complete parent documents for retrieved chunks (ensures context).

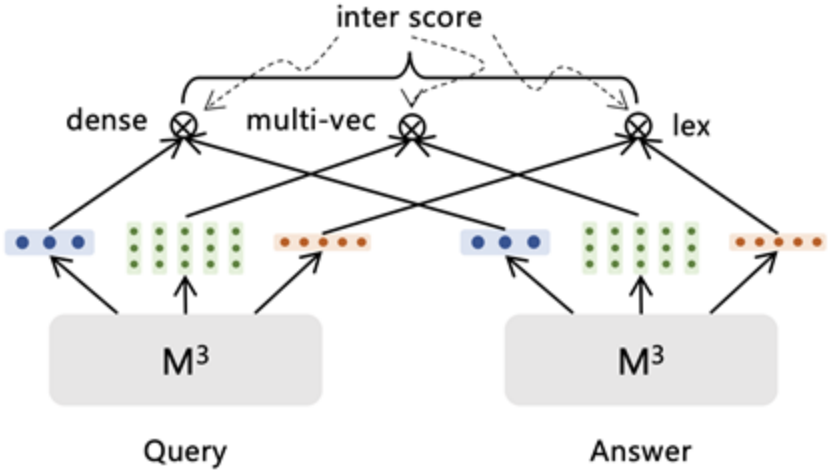


Specialized Embeddings

- Improve embedding leads to better retrieval
- Domain-specific Finetuning embedding models
- Advanced embeddings models



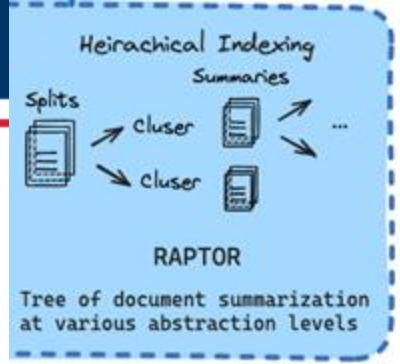


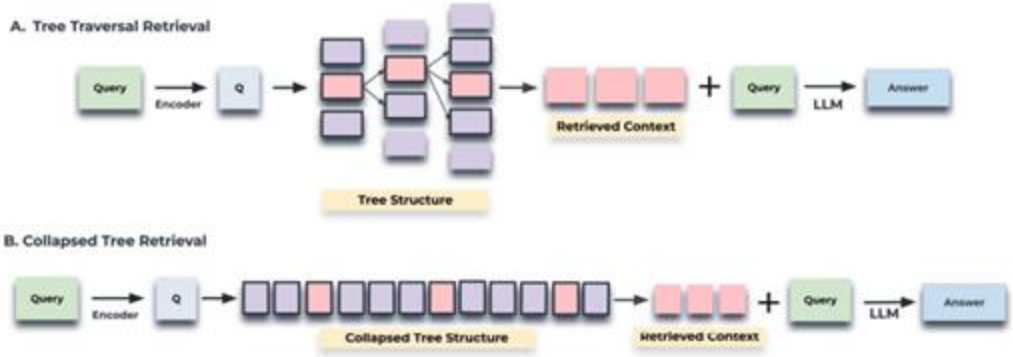


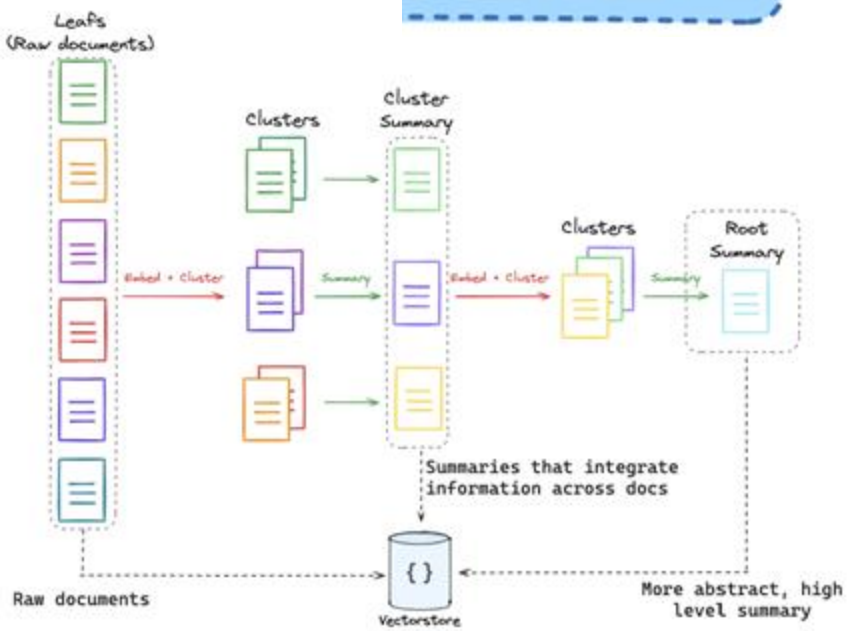
ColBERT: https://github.com/stanford-futuredata/ColBERT

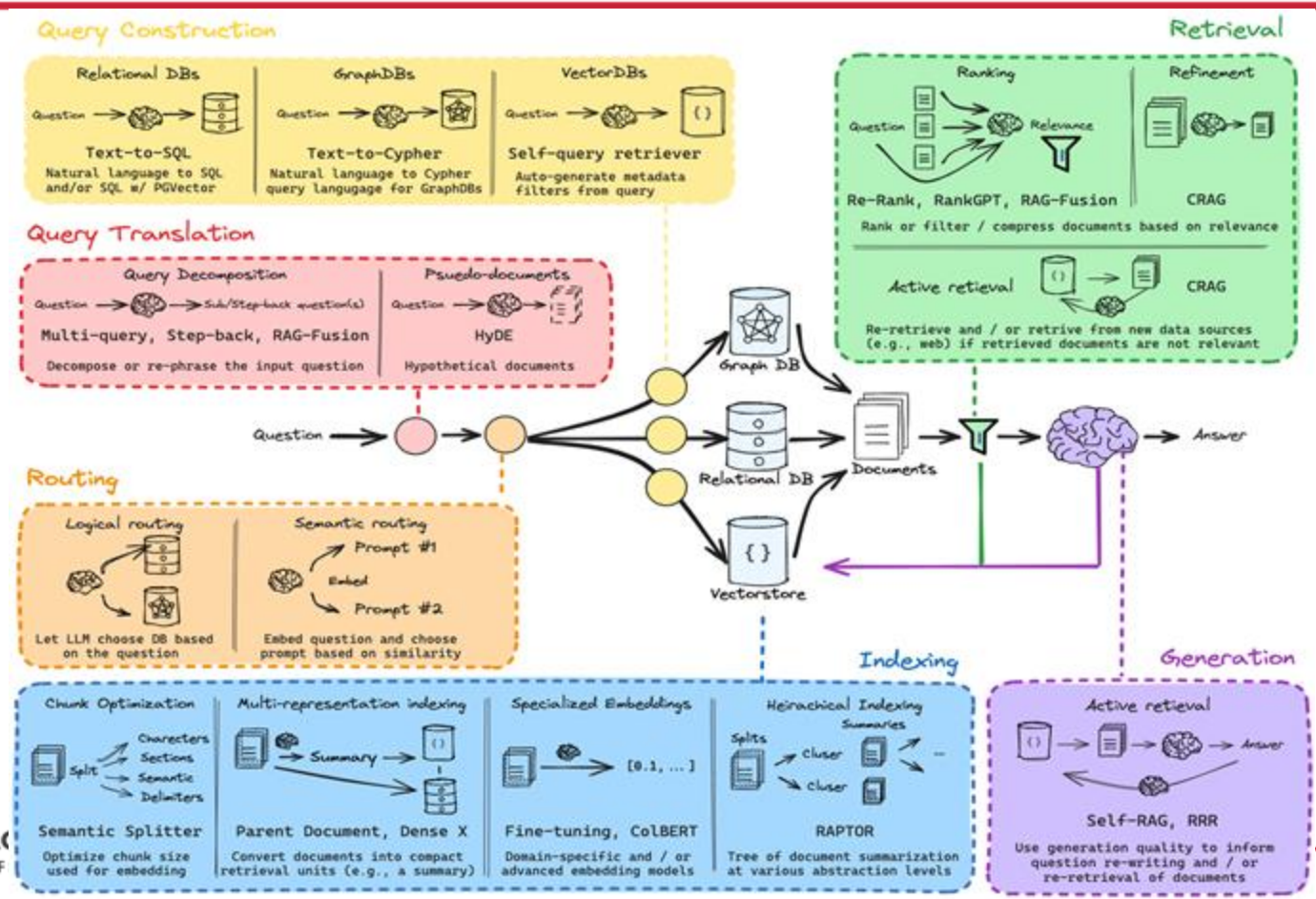
bge-m3 embedding: https://huggingface.co/BAAI/bge-m3

Hierarchical Indexing







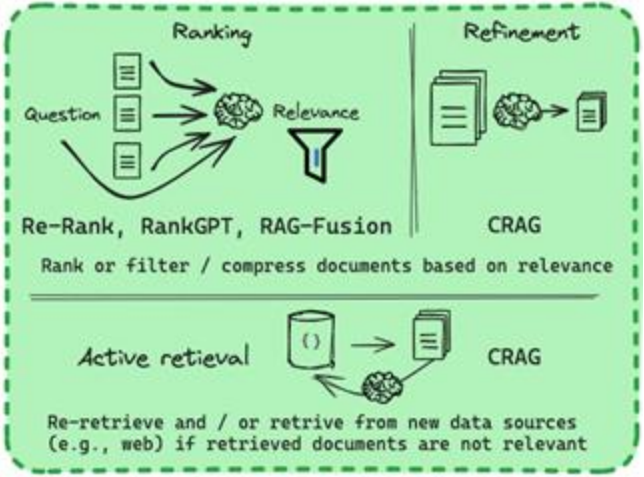

Query Construction
Relational DBs
Question
Text-to-SQL
Natural language to SQL and/or SQL w/ PGVector
GraphDBs
Question
Text-to-Cypher
Natural language to Cypher query langugage for GraphDBs
VectorDBs
Question
{}
Self-query retriever
Auto-generate metadata filters from query
Retrieval
Ranking
Question
Relevance
Re-Rank, RankGPT, RAG-Fusion
Refinement
CRAG
Rank or filter / compress documents based on relevance
Active retieval
{}
CRAG
Re-retrieve and / or retrive from new data sources (e.g., web) if retrieved documents are not relevant
Query Translation
Query Decomposition
Question
Sub/Step-back question(s)
Multi-query, Step-back, RAG-Fusion
Decompose or re-phrase the input question
Psuedo-documents
Question
HyDE
Hypothetical documents
Graph DB
Question
Relational DB
Documents
Answer
Vectorstore
{}
Routing
Logical routing
Let LLM choose DB based on the question
Semantic routing
Prompt #1
Embed
Prompt #2
Embed question and choose prompt based on similarity
Indexing
Chunk Optimization
Split
Characters
Sections
Semantic
Delimiters
Semantic Splitter
Optimize chunk size used for embedding
Multi-representation indexing
Summary
{}
Parent Document, Dense X
Convert documents into compact retrieval units (e.g., a summary)
Specialized Embeddings
[0.1, ...]
Fine-tuning, ColBERT
Domain-specific and / or advanced embedding models
Hierachical Indexing
Summaries
Splits
Cluser
Cluser
...
RAPTOR
Tree of document summarization at various abstraction levels
Generation
Active retieval
{}
Answer
Self-RAG, RRR
Use generation quality to inform question re-writing and / or re-retrieval of documents

Retrieval

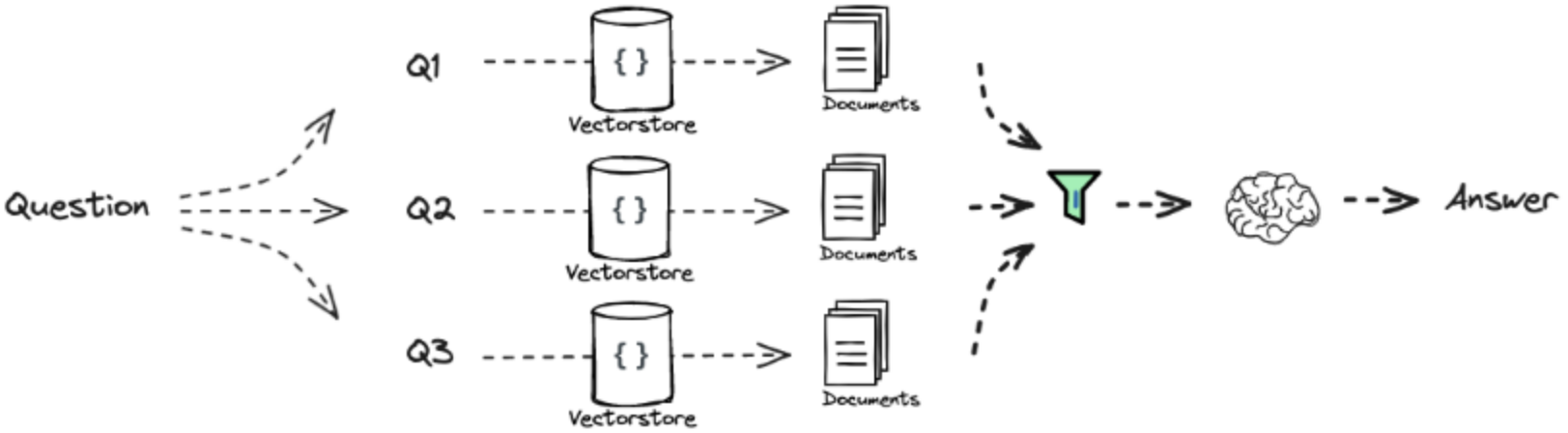
- RAG relies on external knowledge to enhance LLMs,

⇒ The type of **retrieval source** and the **granularity** of retrieval units both affect the final generation results

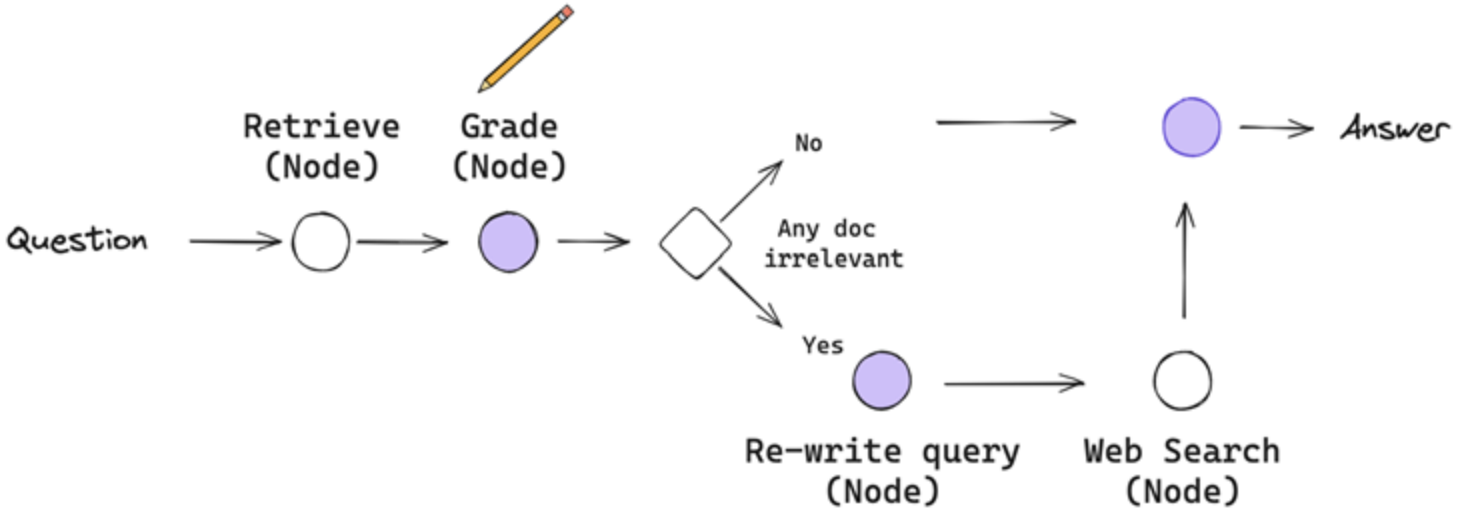
- So far, we mentioned unstructured/ structured data source, LLM-generated content
- In text, retrieval granularity ranges from fine to coarse, including Token, Phrase, Sentence, Proposition, Chunks, Document



RAG-Fusion


Question
Q1
Q2
Q3
Vectorstore
Vectorstore
Vectorstore
Documents
Documents
Documents
Answer

Corrective-RAG



https://github.com/langchain-ai/langgraph/blob/main/examples/rag/langgraph_crag.ipynb

Grade node’s prompt to check whether the given document is relevant or not

```
# Prompt
system = """You are a grader assessing relevance of a retrieved document to a user question. \n
    If the document contains keyword(s) or semantic meaning related to the question, grade it as relevant. \n
    Give a binary score 'yes' or 'no' score to indicate whether the document is relevant to the question."""
grade_prompt = ChatPromptTemplate.from_messages(
    [
        ("system", system),
        ("human", "Retrieved document: \n\n {document} \n\n User question: {question}"),
    ]
)
```

If does not find any relevant docs based on the input question, ask LLM to re-write the question then use this question for web search

```
question = "agent memory"
```

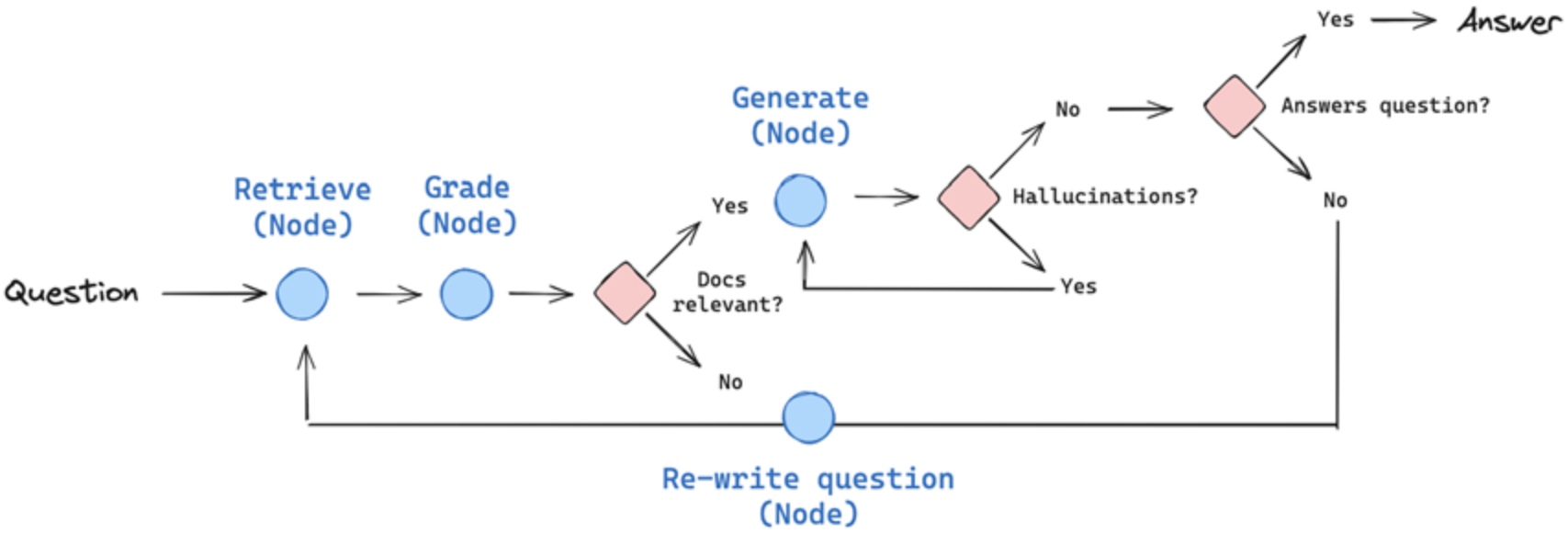
```
# Prompt
system = """You a question re-writer that converts an input question to a better version that is optimized \n
     for web search. Look at the input and try to reason about the underlying semantic intent / meaning."""
re_write_prompt = ChatPromptTemplate.from_messages(
    [
        ("system", system),
        (
            "human",
            "Here is the initial question: \n\n {question} \n Formulate an improved question.",
        ),
    ]
)

question_rewriter = re_write_prompt | llm | StrOutputParser()
question_rewriter.invoke({"question": question})
```

```
'What is the role of memory in artificial intelligence agents?'
```

Generation

- After retrieval, it is not good to directly input all the retrieved docs to LLM
- Problem:
  - Redundant infor interfere with final generation
  - “Lost in the middle” problem with long context
- Solutions
  - Reranking: ColBert, Reranker models, etc.,
  - Context compression: uses small LM to remove low-informatic words

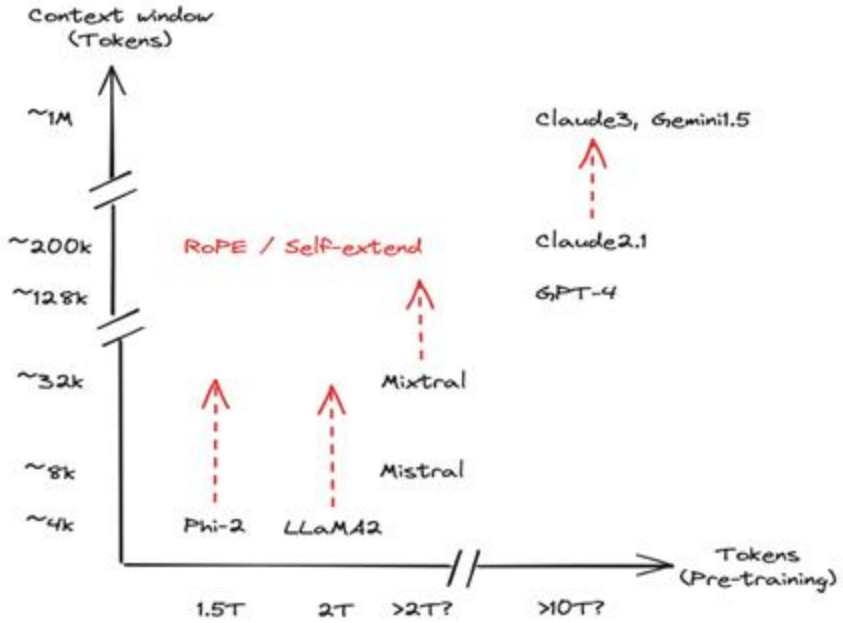
Self-RAG



https://github.com/langchain-ai/langgraph/blob/main/examples/rag/langgraph_self_rag.ipynb

Do We Still Need RAG?

- Context length is improved significantly
- Should we feed all the data into the context, then query?

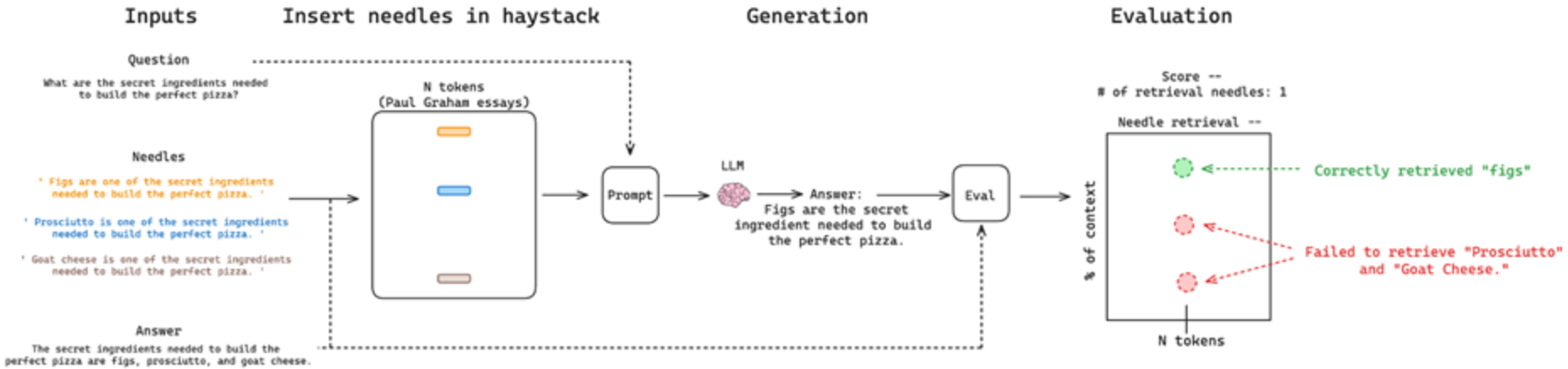


⇒ Possibly No

Long context:

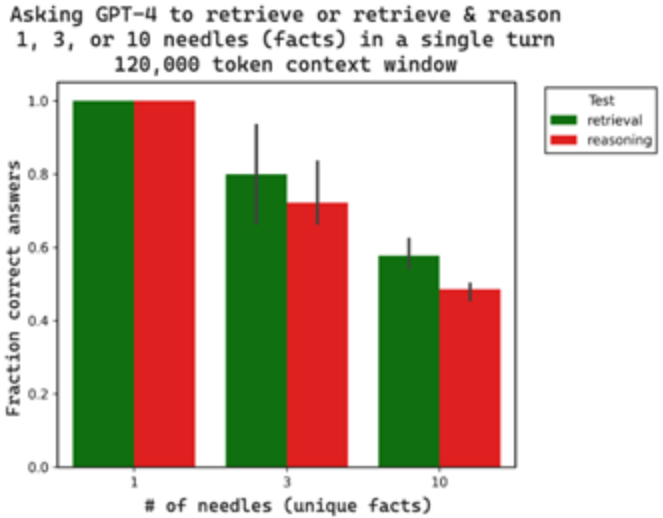
- More API's fee / Memory
- Needle in a Haystack problem

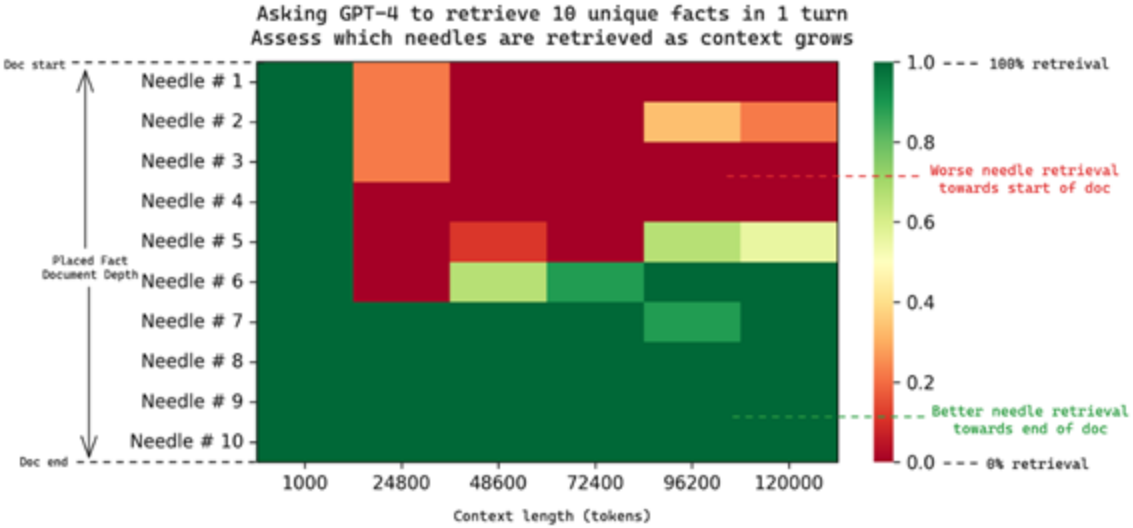
Needle In A Haystack: test reasoning & retrieval in long context LLMs



https://github.com/gkamradt/LLMTest_NeedleInAHaystack

Retrieval is not guaranteed, reasoning harder than retrieval





https://youtu.be/UlmyyYQGhzc
https://blog.langchain.dev/multi-needle-in-a-haystack/

LLMs perform worse with long context benchmarks

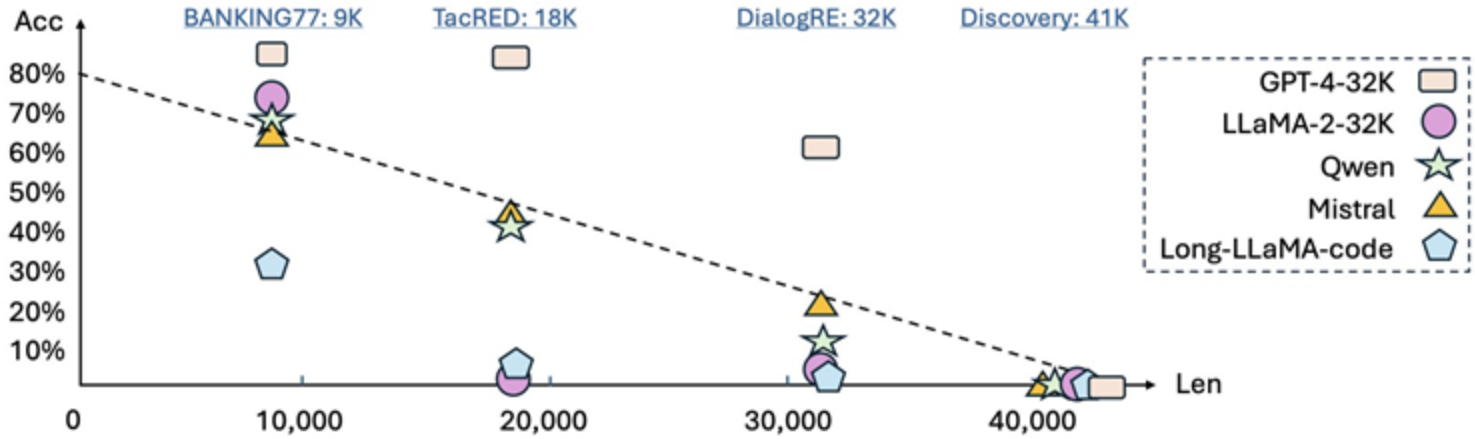


Figure 3: Results for representative models across different evaluation datasets. The performance greatly decreases as the task becomes more challenging. Some models even decays linearly w.r.t the demonstration length.

https://arxiv.org/abs/2404.02060

Thank you for your attention!