创业公司视角下的RAG实践

王 楠
Jina AI联合创始人兼CTO

Berlin · Beijing · Shenzhen

Jina AI

- 全球化分布式办公
- 开源软件
- 专注AI开发工具

Berlin HQ

Beijing

Shenzhen

Raised	Office	Members	AI Company
$ 38M	Berlin / Beijing / Shenzhen	30+	Top-tier

RAG是少数落地的GenAI应用场景

RAG是为数不多的GenAI落地场景

幻觉

- 基于检索结果
- 保证可解释性和可回溯性

回答可以验证追溯

知识更新成本高

- 更新检索知识库
- 支持增删改查

知识可以频繁更新

私有知识注入难

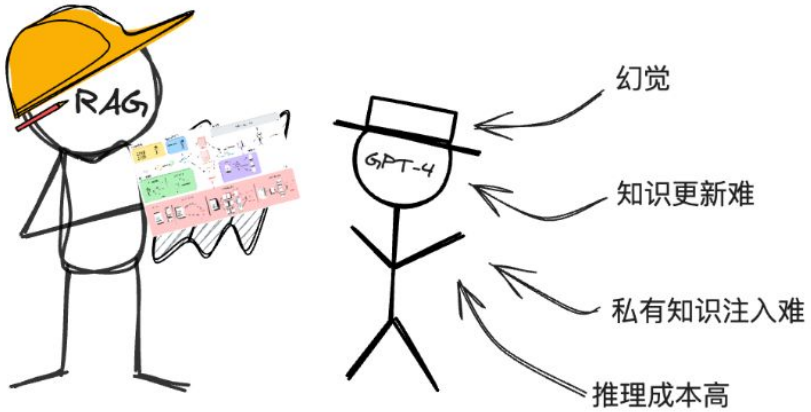
- 支持本地部署
- 本地存储私有数据
- 不需要微调模型

私有数据安全

推理成本高

- 推理成本低
- 减少LLM输入，降低LLM推理成本

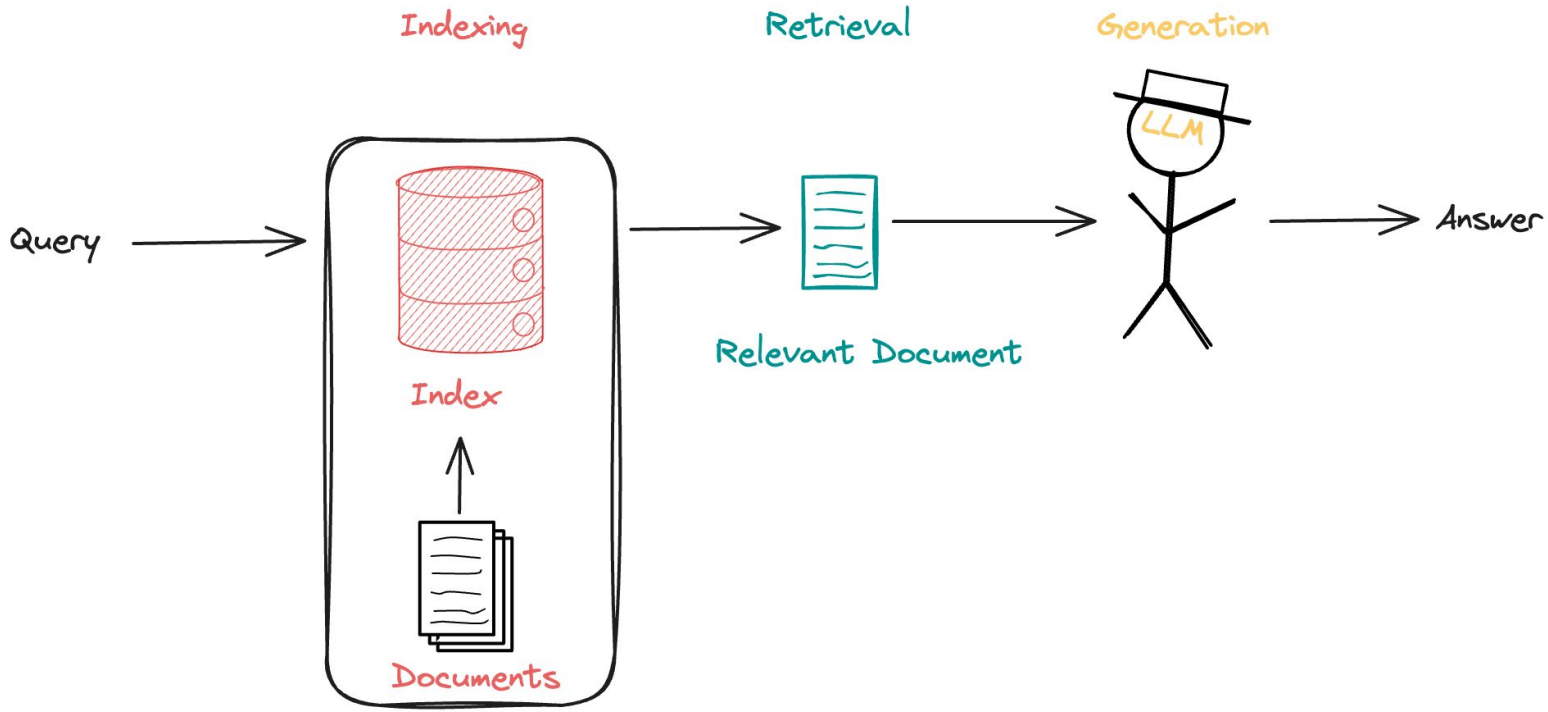
有效降低LLM成本



RAG非常复杂

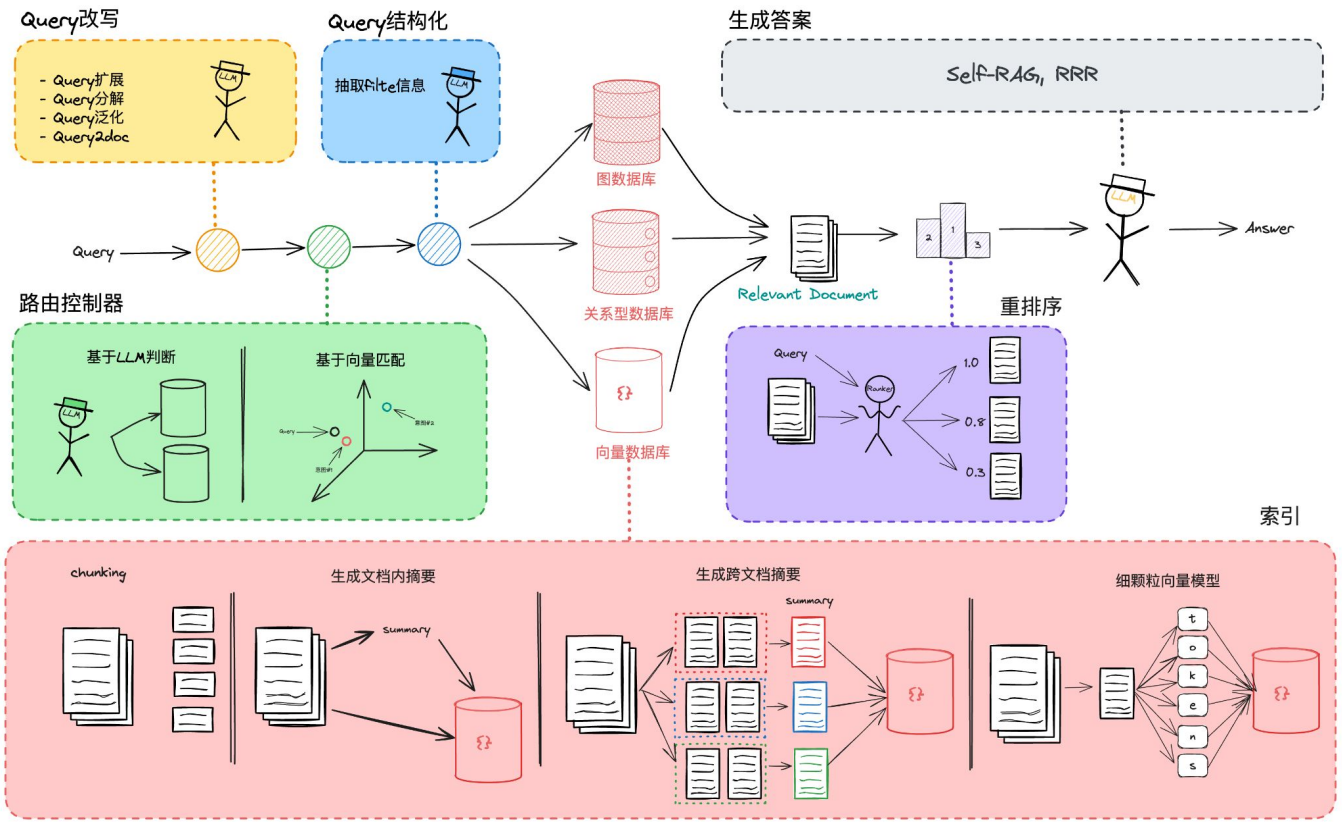
RAG-as-a-Service还没有准备好

POC的RAG流水线


Indexing
Retrieval
Generation
LLM
Query
Answer
Index
Relevant Document
Documents

实际生产的RAG流水线

- Query改写
- 路由控制器
- Query结构化
- 索引优化
- 重排序
- ...

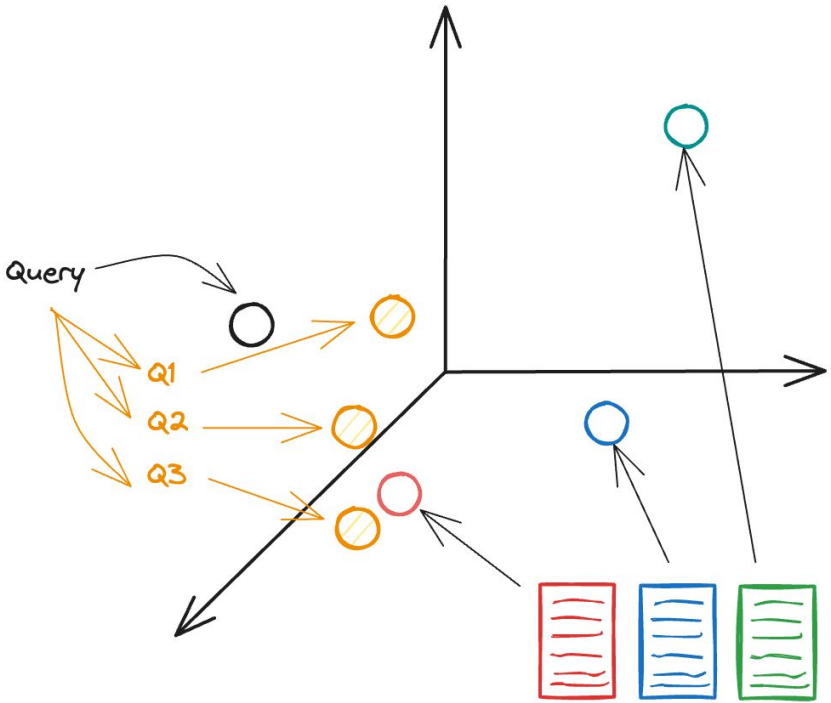


Query改写

使Query和Document更容易匹配

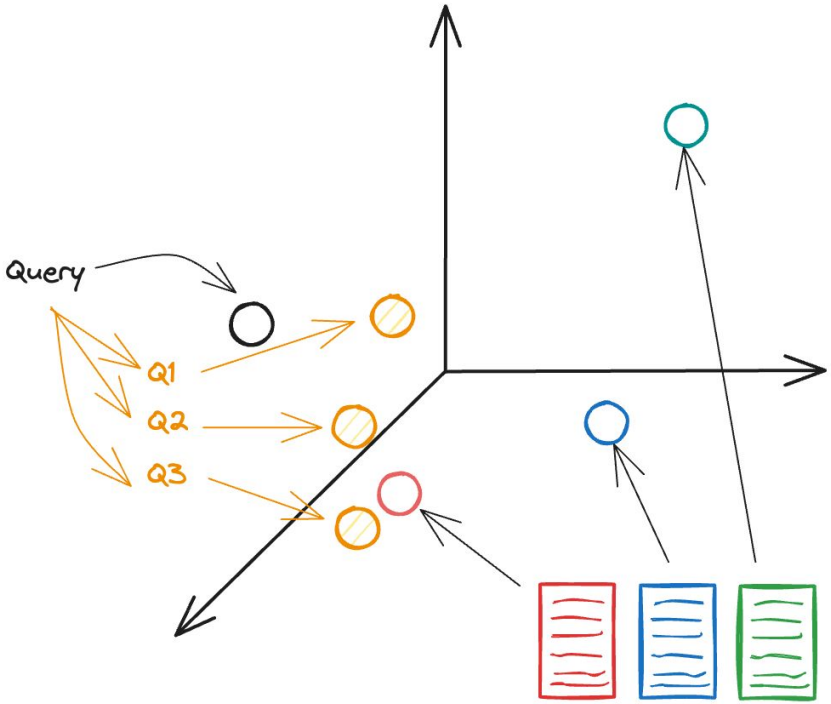
举例:"怎么添加wx支付信息?"

- 等价的Query
  - "怎么添加微信支付信息?"
- 更抽象的Query
  - "如何补充支付信息?"
- 更具体的Query
  - "如何找到用户设置页面?"
  - "如何在用户设置页面中补充支付信息?"
  - "如何选择微信支付?"

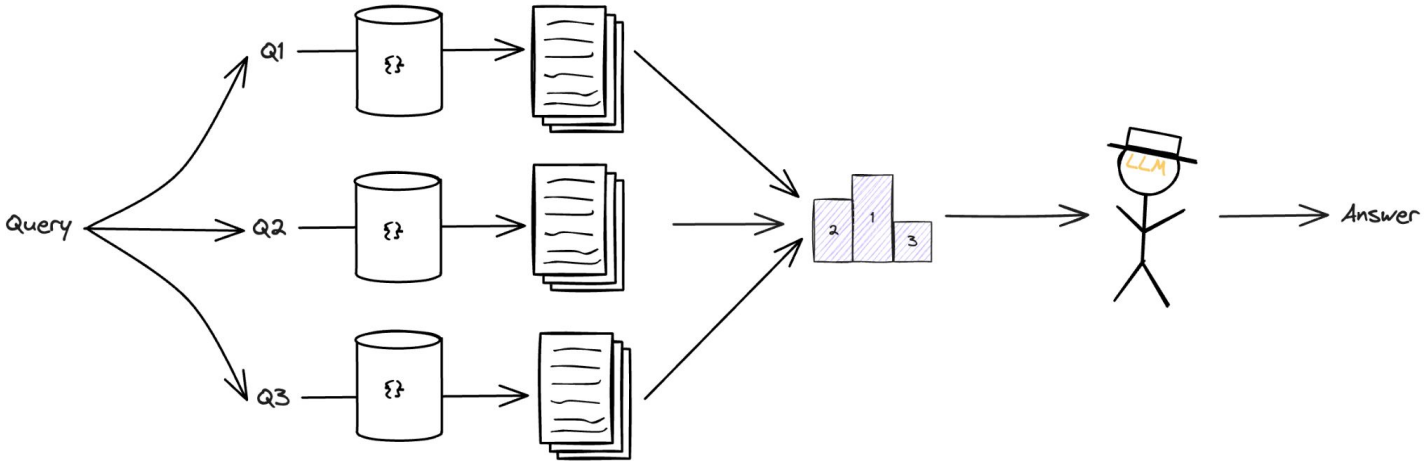


Query改写

- 使Query更接近语料
  - ”怎么添加wx支付信息?“
- 等价的Query
  - ”怎么添加微信支付信息?“
- 更抽象的Query
  - ”如何补充支付信息？“
- 更具体的Query
  - ”如何找到用户设置页面？”
  - “如何在用户设置页面中补充支付信息？“
  - “如何选择微信支付？”



等价Query：并行多查询



- 使用LLM生成等价query
- 多路并行召回
- 使用排序模型或启发式合并

等价Query：并行多查询

Query: What is the BLEU score of transformer on WMT 2014?

Q1: Can you provide the BLEU score achieved by the transformer model on the WMT 2014 dataset?

Q2: What was the BLEU score obtained by the transformer architecture when evaluated on the WMT 2014 dataset?

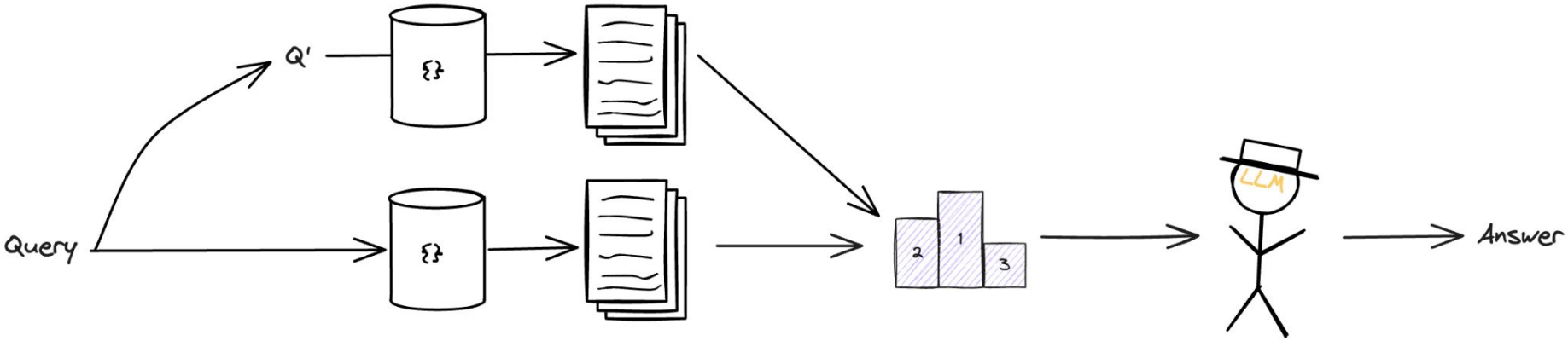
Q3: How well did the transformer model perform in terms of BLEU score on the WMT 2014 dataset?

```
The BLEU score of the
Transformer on WMT 2014 is
28.4, establishing a new
state-of-the-art score.
Additionally, the big model
achieves a BLEU score of 41.0
on the English-to-French
translation task
```

```
The BLEU score of the
Transformer on the WMT 2014
English-to-German translation
task is 28.4, and on the WMT
2014 English-to-French
translation task, it is 41.8.
```

更抽象的Query：回撤

- 使用LLM对问题进行抽象
- 使用原始Query和改写后Query并行召回
- 适用于处理细颗粒的具体问题



更抽象的Query：回撤

Query: How many GPUs are needed for training transformer?

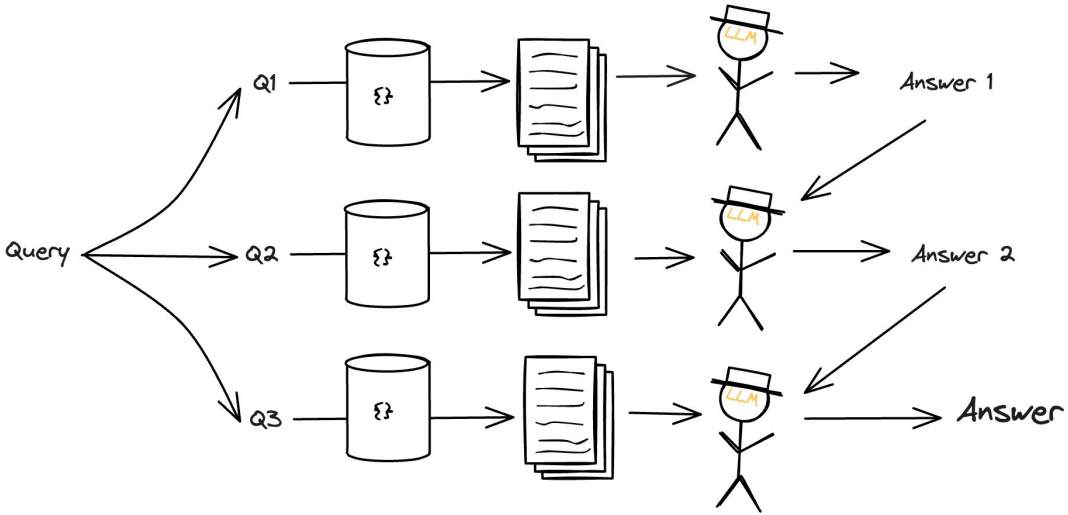
Q': what hardware is required for training transformer?'

Based on the provided context, the recommended number of GPUs for training transformer models efficiently is eight. The Transformer model achieved a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, which was a small fraction of the training costs of the best models from the literature.

To train the Transformer model, a total of eight GPUs are needed. This is based on the information provided in the documents, where it states that the models were trained on one machine with 8 NVIDIA P100 GPUs. The training process involved both base models and big models, with different hyperparameters and training steps. The big models, described at the bottom line of table 3, were trained for 300,000 steps over 3.5 days using the eight GPUs. This setup allowed for efficient training of the Transformer model, showcasing its ability to achieve state-of-the-art results in machine translation tasks while being more parallelizable and requiring significantly less time to train compared to other models based on recurrent or convolutional neural networks.

更具体的Query：任务分解

- 使用LLM拆解问题
- 根据上一步问题的答案和当前的问题生成答案
- 多路串行召回
- 适合语义颗粒度大的抽象问题



等价Query：并行多查询

Query: What are encoder and decoder stacks in transformer?

Q1. How do encoder stacks work in a transformer model?

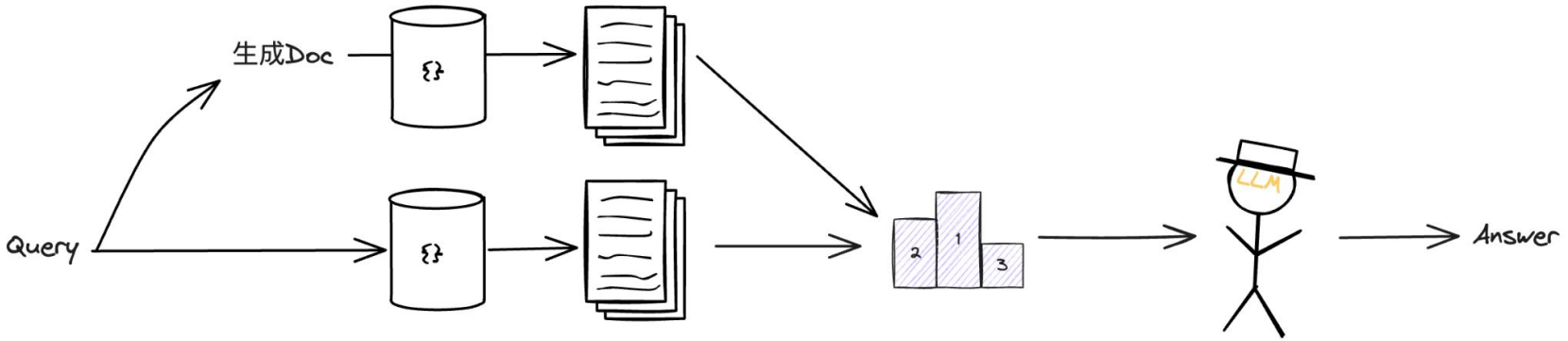
Q2. What is the purpose of decoder stacks in a transformer architecture?

Q3. Can you explain the differences between encoder and decoder stacks in a transformer?

In the Transformer model, the encoder stack is composed of a stack of N=6 identical layers, each consisting of two sub-layers - a multi-head self-attention mechanism and a simple, position-wise fully connected feed-forward network. The decoder stack is also composed of a stack of N=6 identical layers, with each layer containing two sub-layers from the encoder stack and an additional third sub-layer for multi-head aligned RNNs or convolution.

In a transformer model, the encoder stack is composed of N = 6 identical layers, each containing two sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. …
On the other hand, the decoder stack in a transformer also consists of N = 6 identical layers. … the decoder stack is responsible for producing the desired output sequence based on the continuous representations provided by the encoder.

根据query生成Document: HyDE



- Document有更丰富的信息，可以更精准的匹配
- 用LLM根据Query生成Document
- 使用原始Query和生成的Document并行召回

HyDE

Query: What learning rate is used when training transformer?

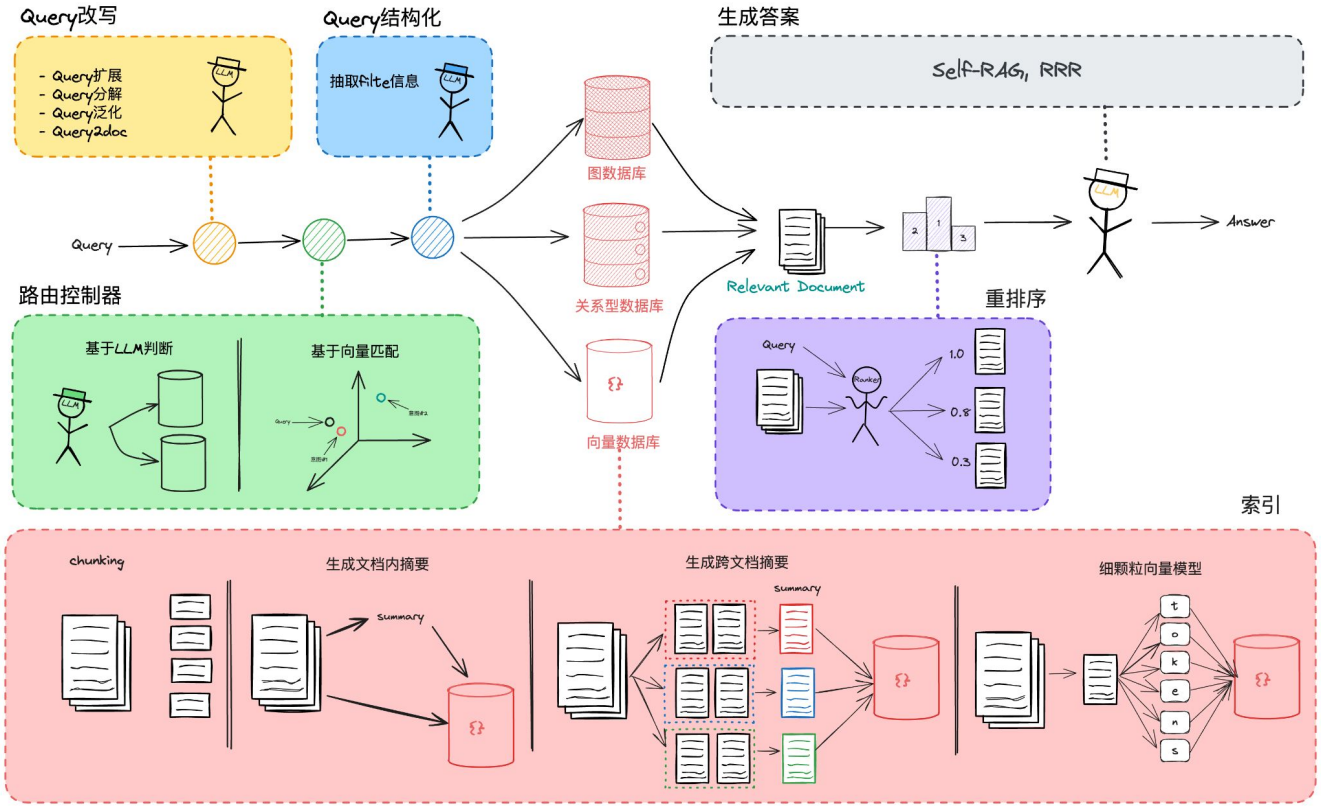
The learning rate used when training a transformer model is a crucial hyperparameter that significantly impacts the model's performance and convergence speed. In practice, a common learning rate schedule for training transformers involves using a warmup strategy followed by a decay schedule. During the warmup phase, the learning rate is gradually increased from a small value to the specified initial learning rate over a certain number of steps. This helps the model to stabilize and avoid large fluctuations in the early stages of training. \n\nAfter the warmup phase, the learning rate is typically decayed exponentially or according to a predefined schedule to fine-tune the model's parameters and improve convergence. The specific learning rate values and decay schedule can vary depending on the dataset, model architecture, and training objectives. ...

```
The learning rate used when training the transformer
is warmup_steps = 4000, decreasing proportionally to
the inverse square root of the step number. The base
model of the transformer uses a training cost of 3.3 ·
10^18 FLOPs, while the big model uses a training cost
of 2.3 · 10^19 FLOPs. The base model applies a dropout
rate of Pdrop = 0.1 during training.
```

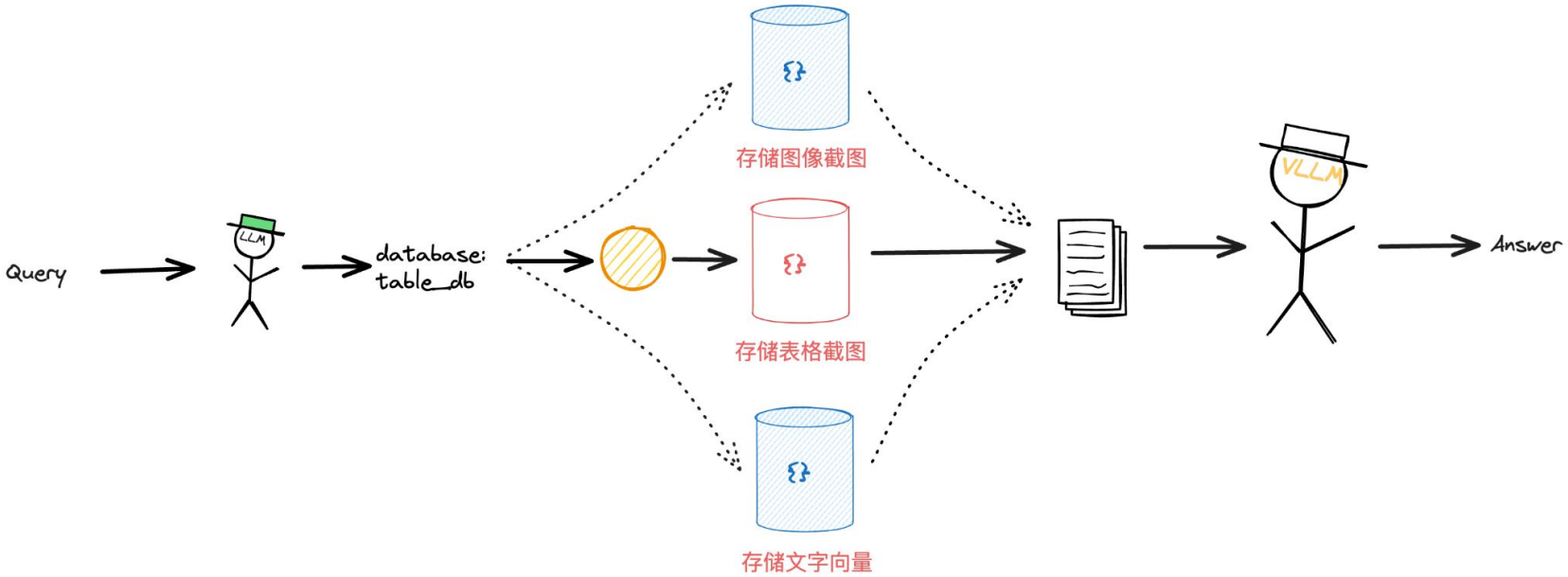
```
The learning rate used when training the Transformer
is determined by the formula:

lrate = d^-0.5 * model * min(step_num^-0.5, step_num
* warmup_steps^-1.5)

This formula increases the learning rate linearly
for the first warmup_steps training steps and then
decreases it proportionally to the inverse square
root of the step number. The warmup_steps value used
in this case is 4000.
```

路由控制器

- 意图识别
- 根据用户的意图决定召回路径



路由控制器


Query
LLM
database:
table_db
存储图像截图
存储表格截图
存储文字向量
VLLM
Answer

路由控制器

Query: summarize the conclusion from table 1.

datasource='table_images'

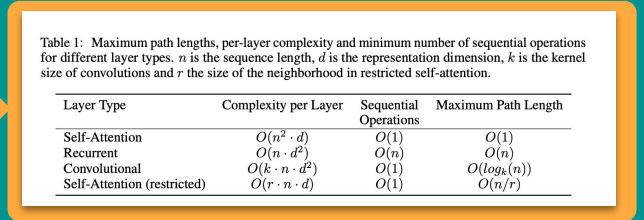


Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

`summarize the conclusion from table 1.`

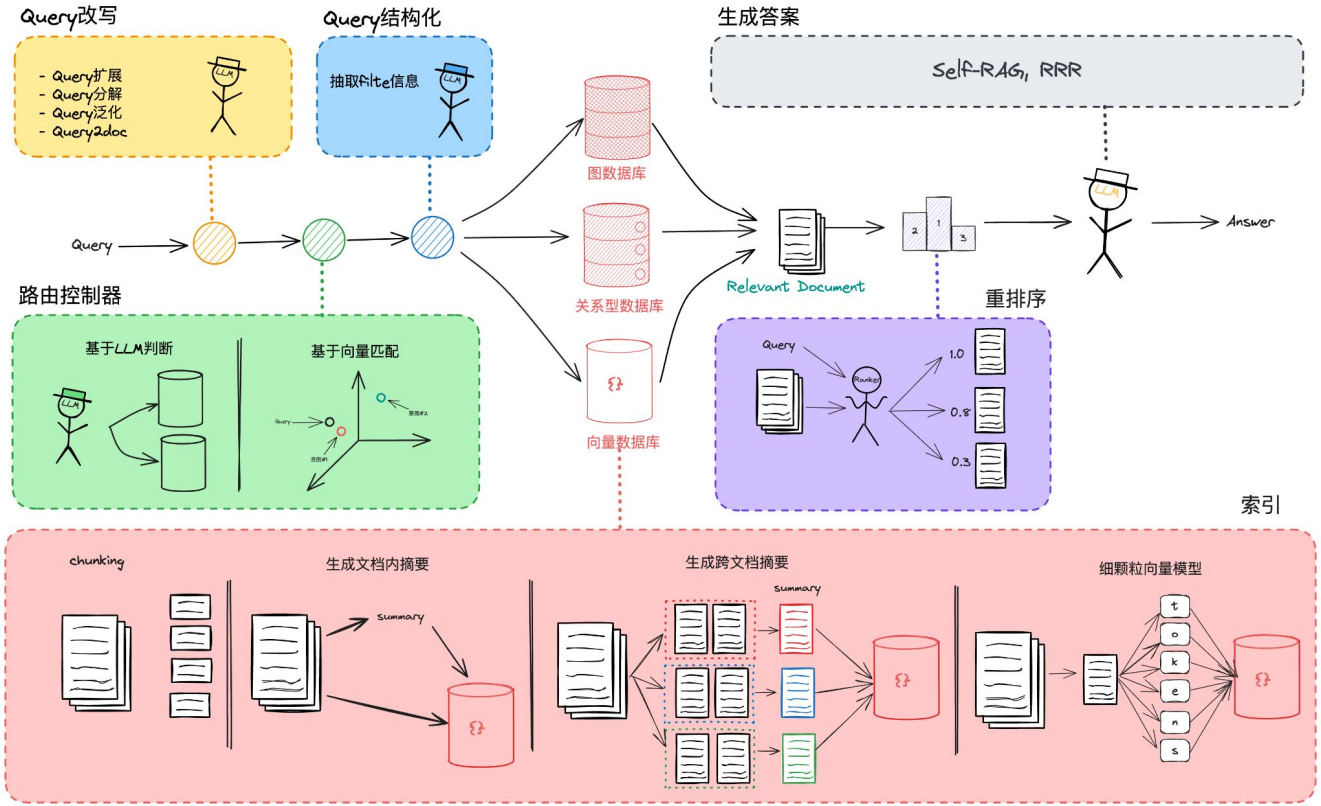
Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

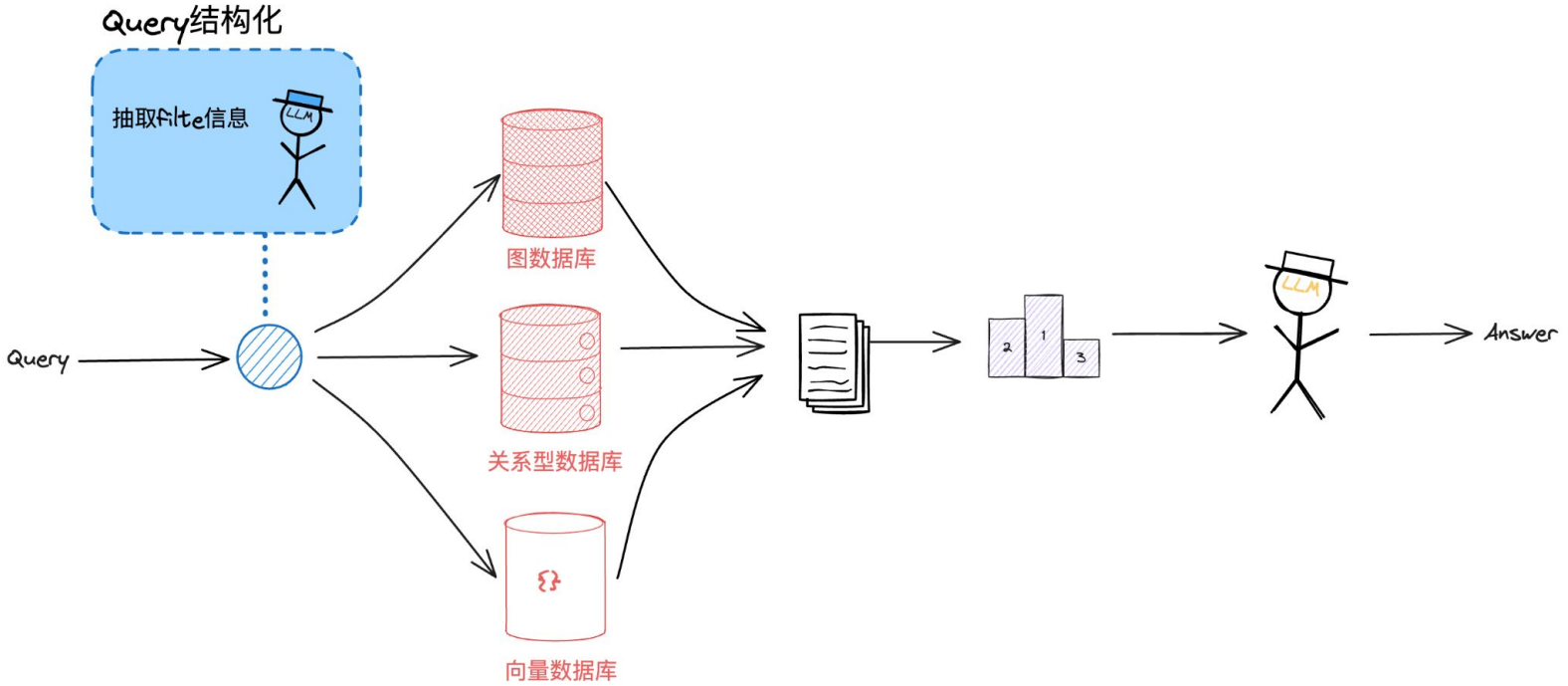
```
Table 1 provides a comparative summary of various
layer types used in neural network architectures,
focusing on three key metrics: complexity per
layer, sequential operations, and maximum path
length. Here is the summarized content of the
table:
…

- \( r \) represents the size of the neighborhood
in restricted self-attention.

The table highlights the differences in
computational complexity and efficiency across
various neural network layers, illustrating
trade-offs between computational resources and
performance characteristics.
```

Query结构化

- 使用LLMs抽取结构化信息
- 根据数据库的schema构建query



Query结构化

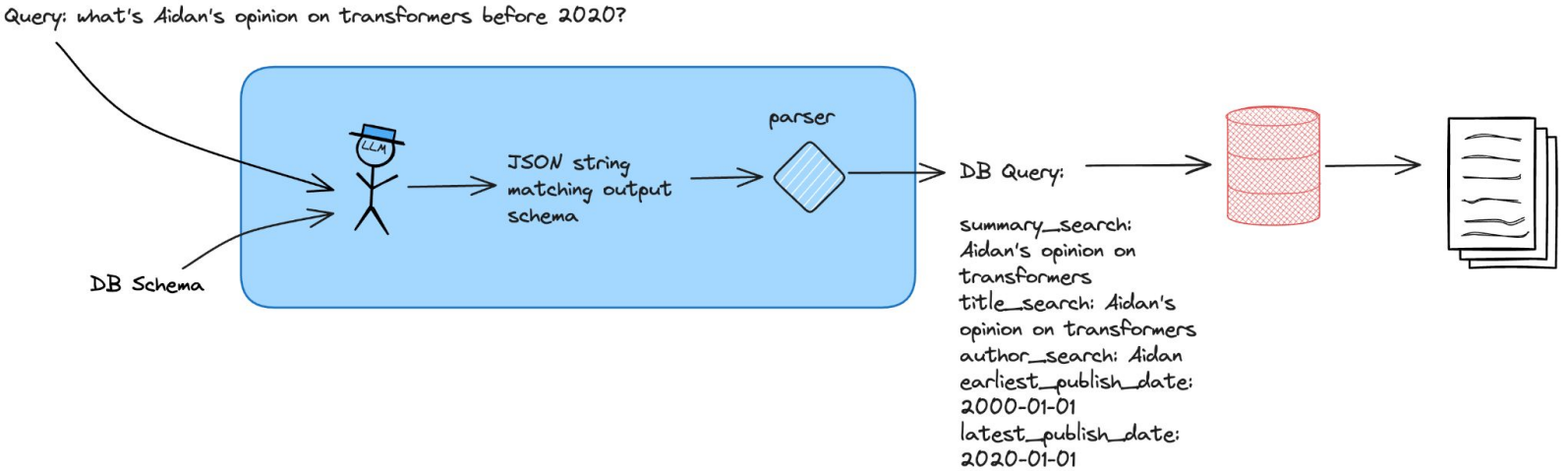

Query结构化
抽取Filte信息
LLM
图数据库
Query
关系型数据库
{}
向量数据库
2
1
3
LLM
Answer

Query结构化

Query: What's Aidan's opinion on transformers before 2020?

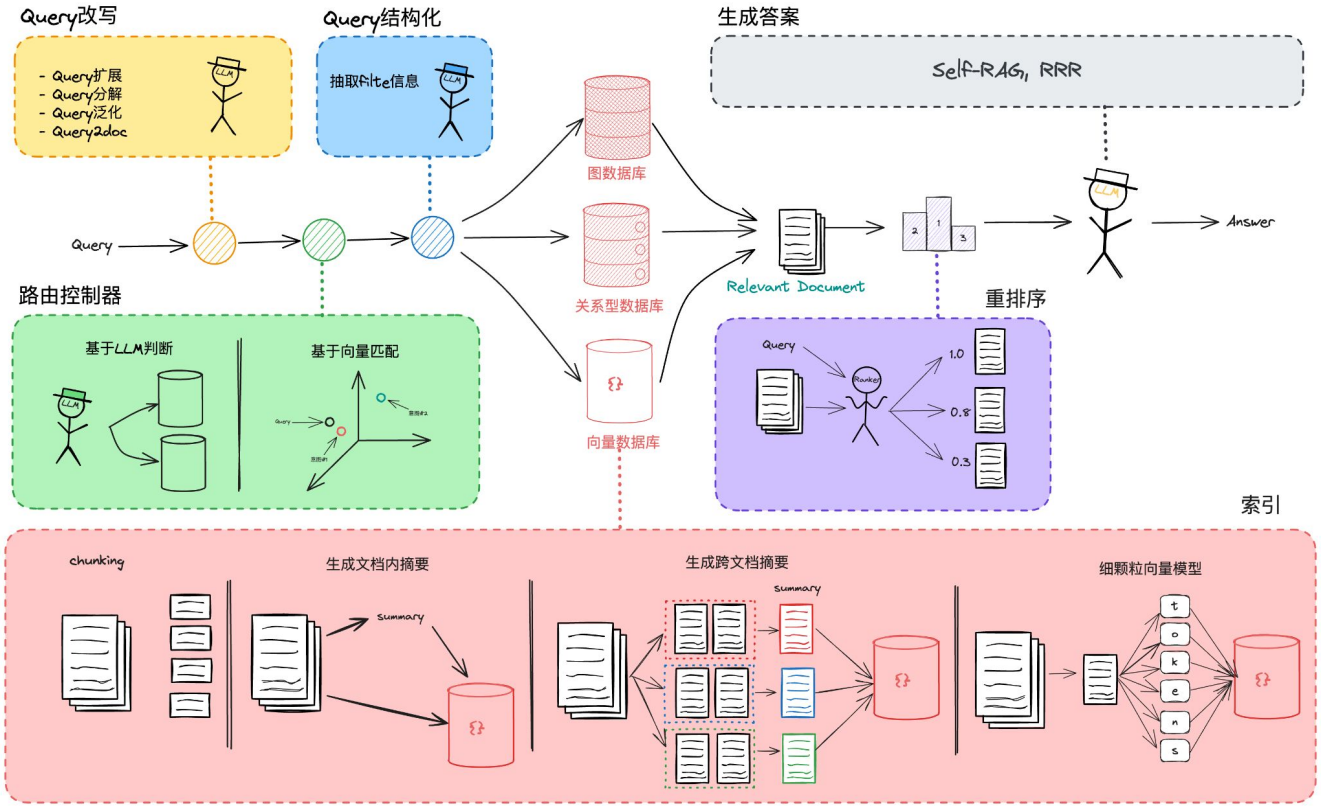
You are an expert at converting user questions into database queries. You have access to a database of tutorial videos about a software library for building LLM-powered applications. Given a question, return a database query optimized to retrieve the most relevant results.If there are acronyms or words you are not familiar with, do not try to rephrase them.

```
summary_search: Aidan's opinion on transformers
title_search: Aidan's opinion on transformers
author_search: Aidan
earliest_publish_date: 2000-01-01
latest_publish_date: 2020-01-01
```

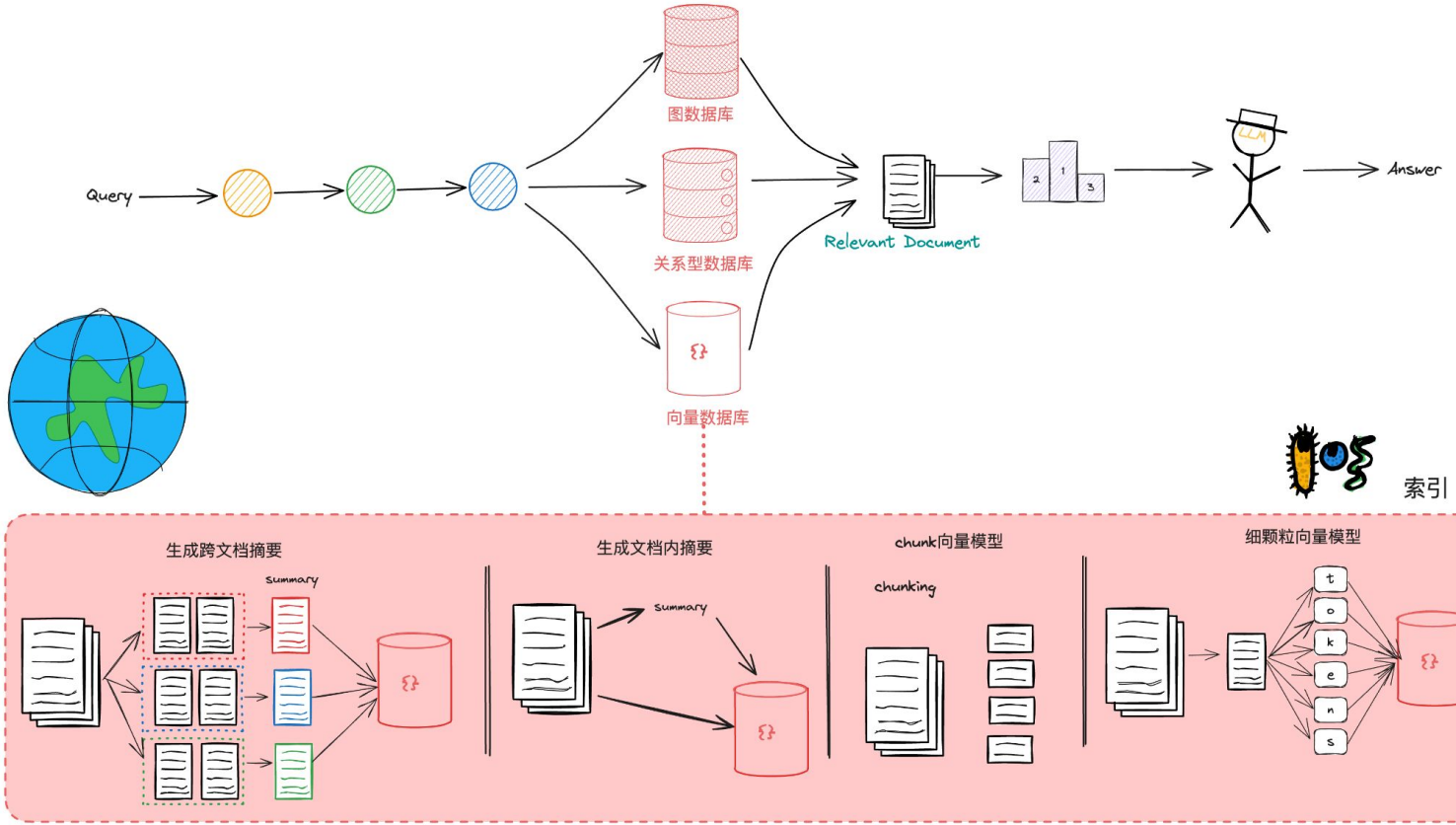


多向量表示

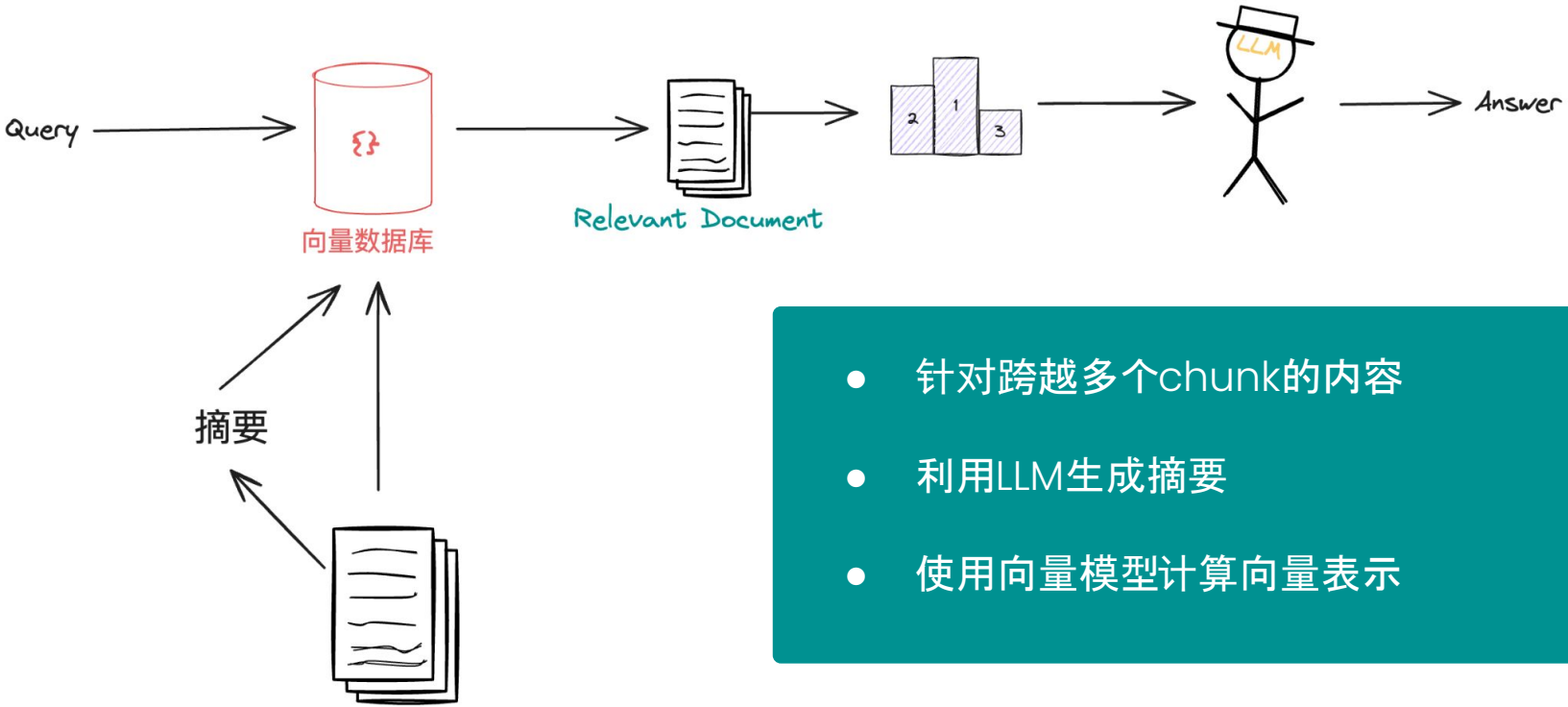
- 文档内摘要向量表示
- 文档间摘要向量表示
- 细颗粒度向量表示



多向量表示


Query
图数据库
关系型数据库
向量数据库
Relevant Document
Answer
索引
生成跨文档摘要
summary
生成文档内摘要
summary
chunk向量模型
chunking
细颗粒向量模型
token

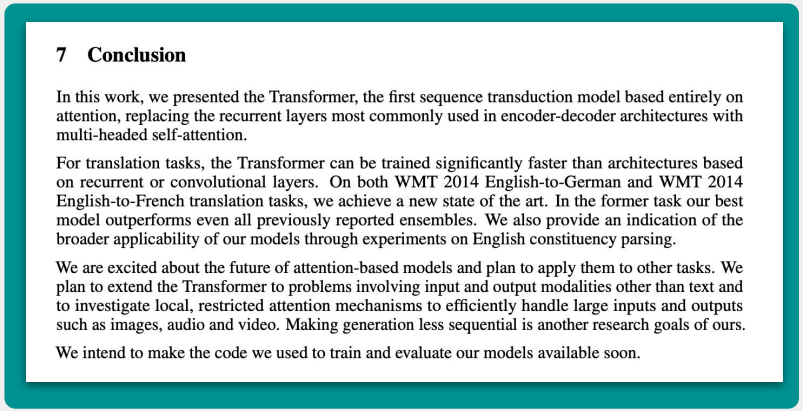
文档内生成摘要



- 针对跨越多个chunk的内容
- 利用LLM生成摘要
- 使用向量模型计算向量表示

文档内生成摘要

Query: What is the conclusion of this paper?



7 Conclusion

In this work, we presented the Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention.

For translation tasks, the Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers. On both WMT 2014 English-to-German and WMT 2014 English-to-French translation tasks, we achieve a new state of the art. In the former task our best model outperforms even all previously reported ensembles. We also provide an indication of the broader applicability of our models through experiments on English constituency parsing.

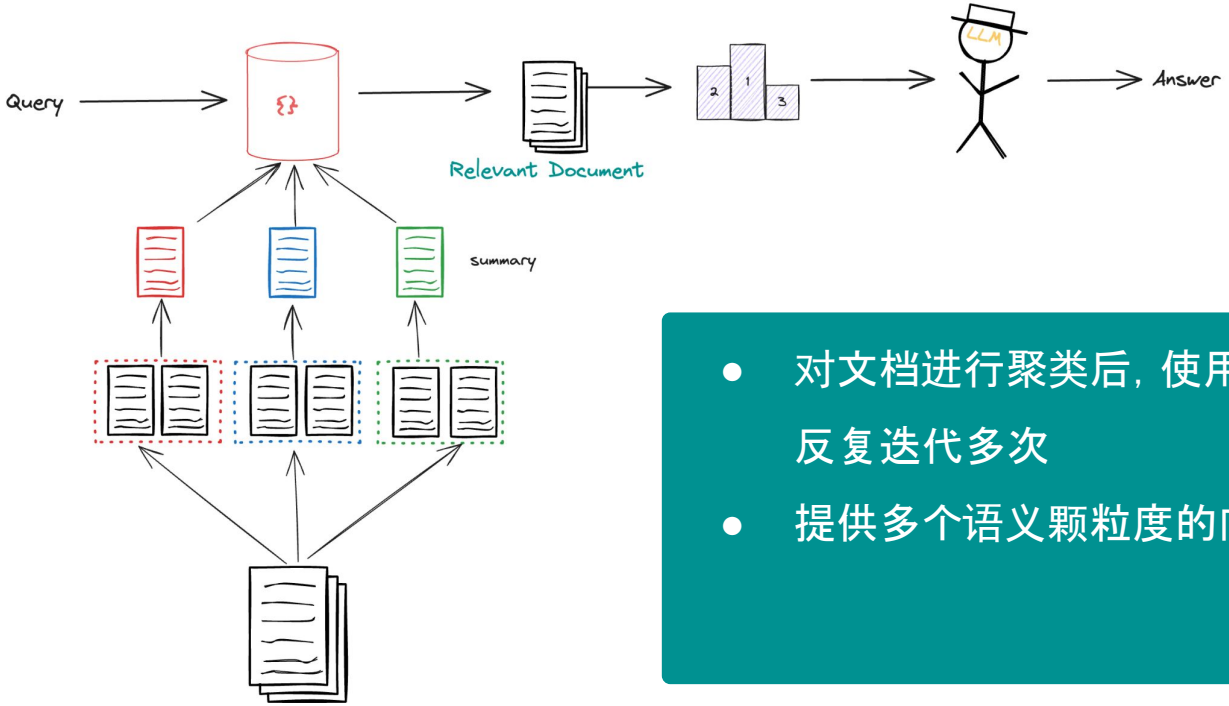
We are excited about the future of attention-based models and plan to apply them to other tasks. We plan to extend the Transformer to problems involving input and output modalities other than text and to investigate local, restricted attention mechanisms to efficiently handle large inputs and outputs such as images, audio and video. Making generation less sequential is another research goals of ours.

We intend to make the code we used to train and evaluate our models available soon.

Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top: Full attentions for head 5. Bottom: Isolated attentions from just the word ‘its’ for attention heads 5
and 6. Note that the attentions are very sharp for this word.

The authors summarize the introduction and success of the Transformer model, which is the first sequence transduction model based solely on attention mechanisms, eliminating the need for recurrent layers typically used in encoder-decoder architectures. The Transformer demonstrates faster training times and achieves state-of-the-art results on translation tasks, notably outperforming previous models on the WMT 2014 English-to-German and English-to-French translation tasks.

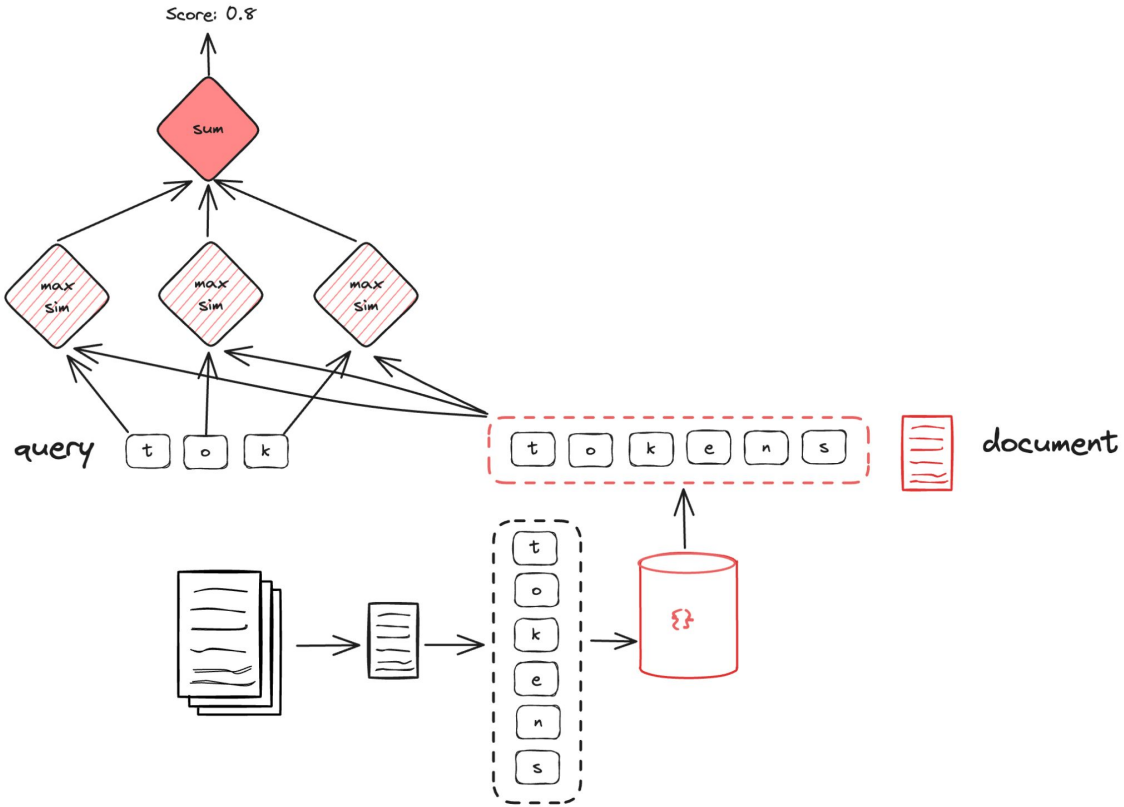
文档间生成摘要：RAPTOR



- 对文档进行聚类后，使用LLM生成总结。反复迭代多次
- 提供多个语义颗粒度的向量表示

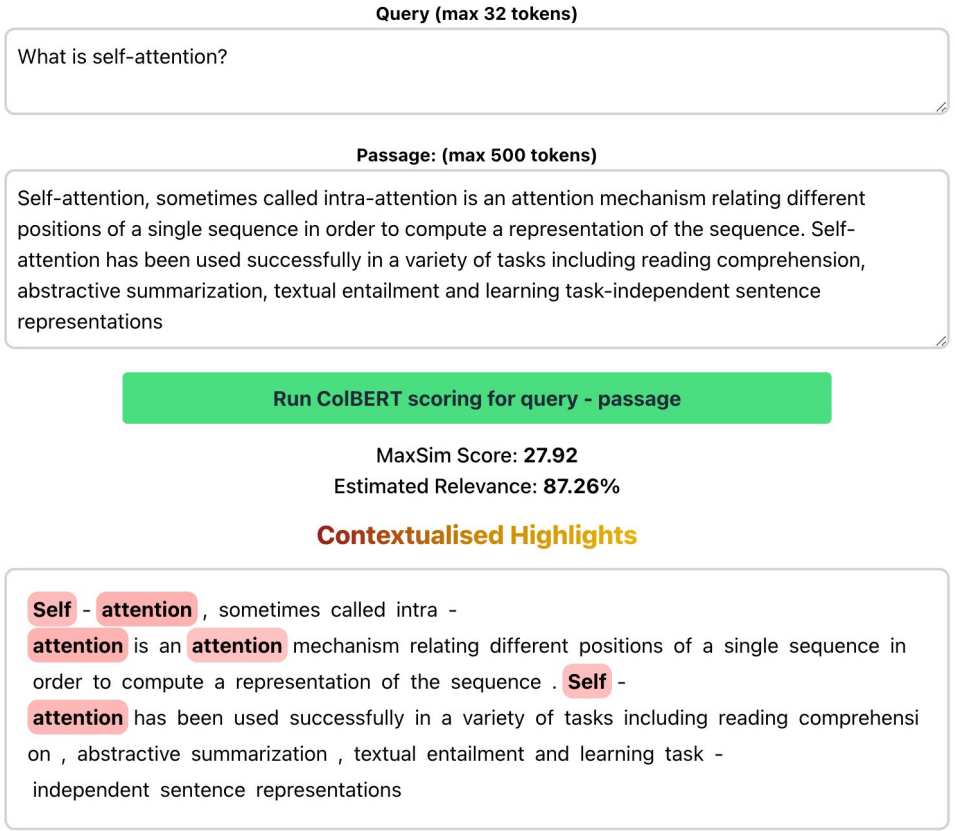
细颗粒度表示-ColBERT

- 存储token颗粒度的向量
- 查询时计算query的每个token的向量表示
- 与每个Document中token的向量表示计算Max-Sum，作为相似度

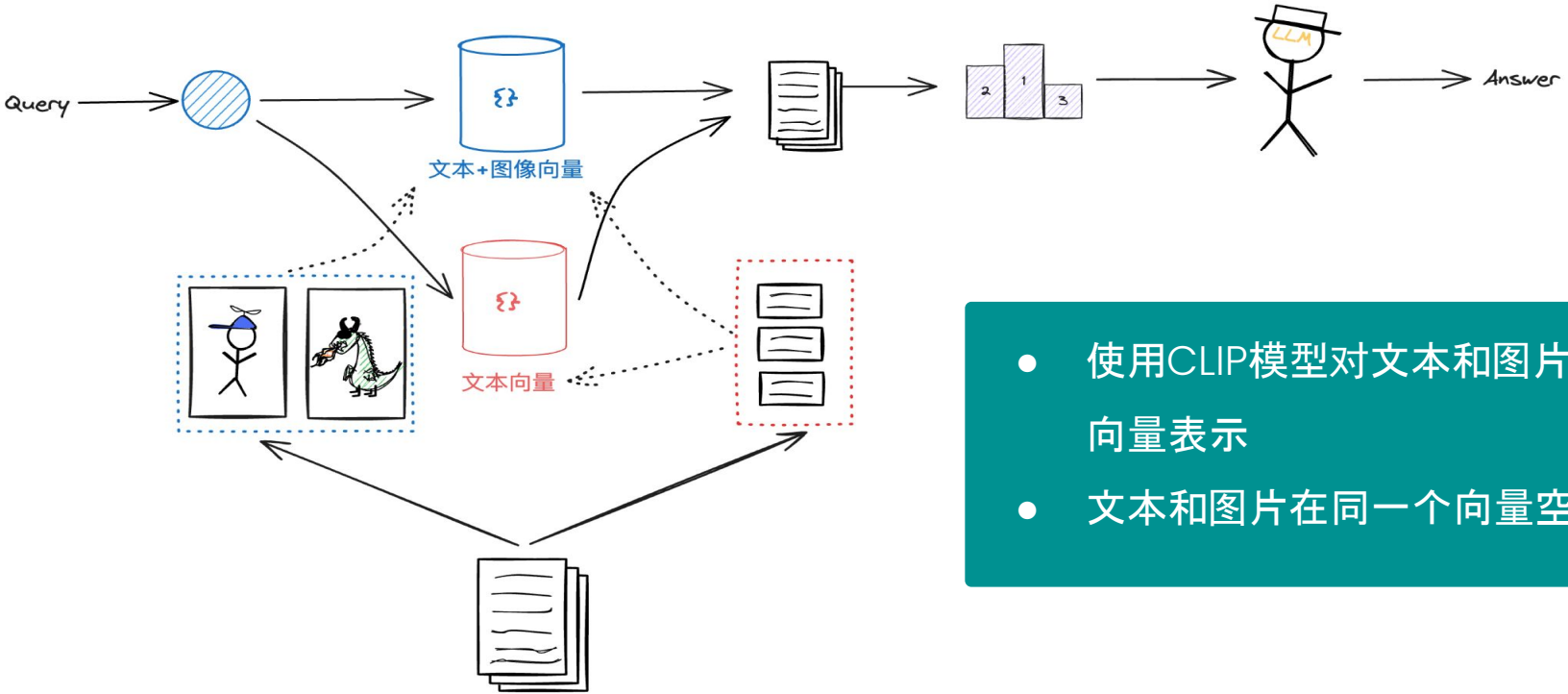


ColBERT举例

- 能够表示token颗粒度的语义
- 向量存储量大

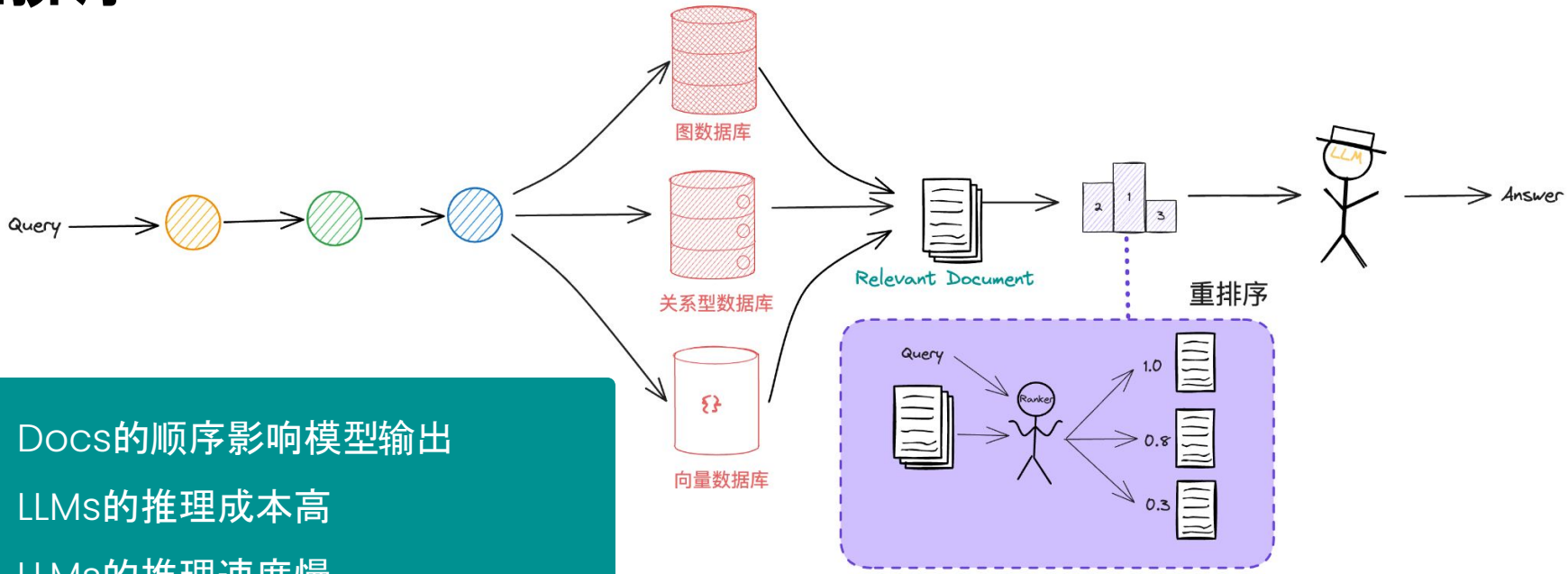


多模态(图片)向量表示



- 使用CLIP模型对文本和图片进行向量表示
- 文本和图片在同一个向量空间

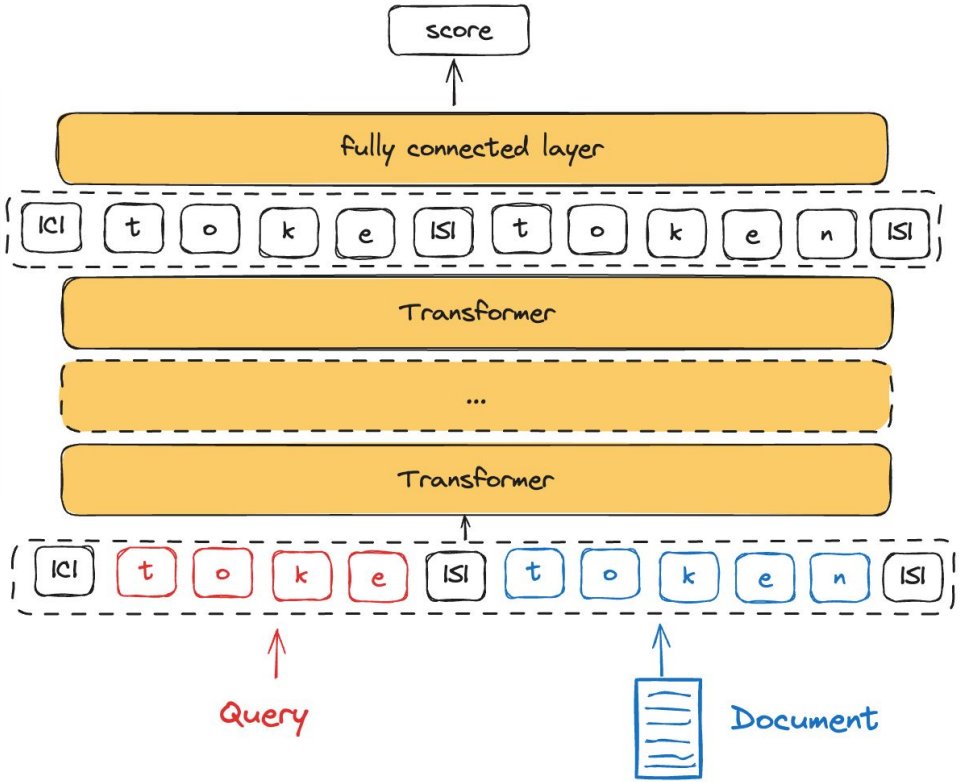
重排序



- Docs的顺序影响模型输出
- LLMs的推理成本高
- LLMs的推理速度慢
- LLMs的context length限制

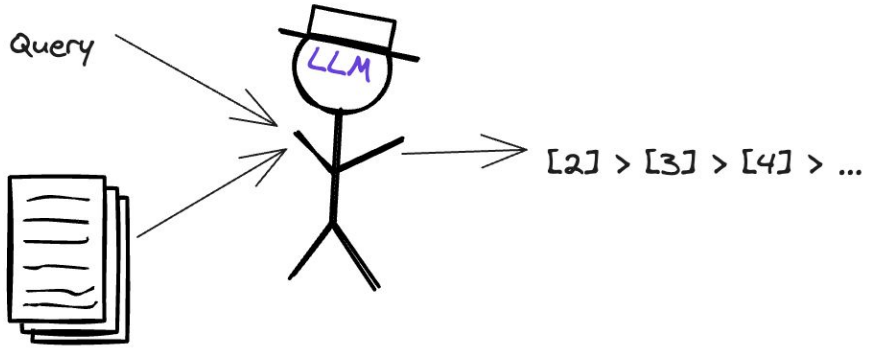
重排序-cross encoder

- 准确率高
- 计算量介于向量模型和LLMs之间
- 捕捉query和文本间细颗粒度语义相似

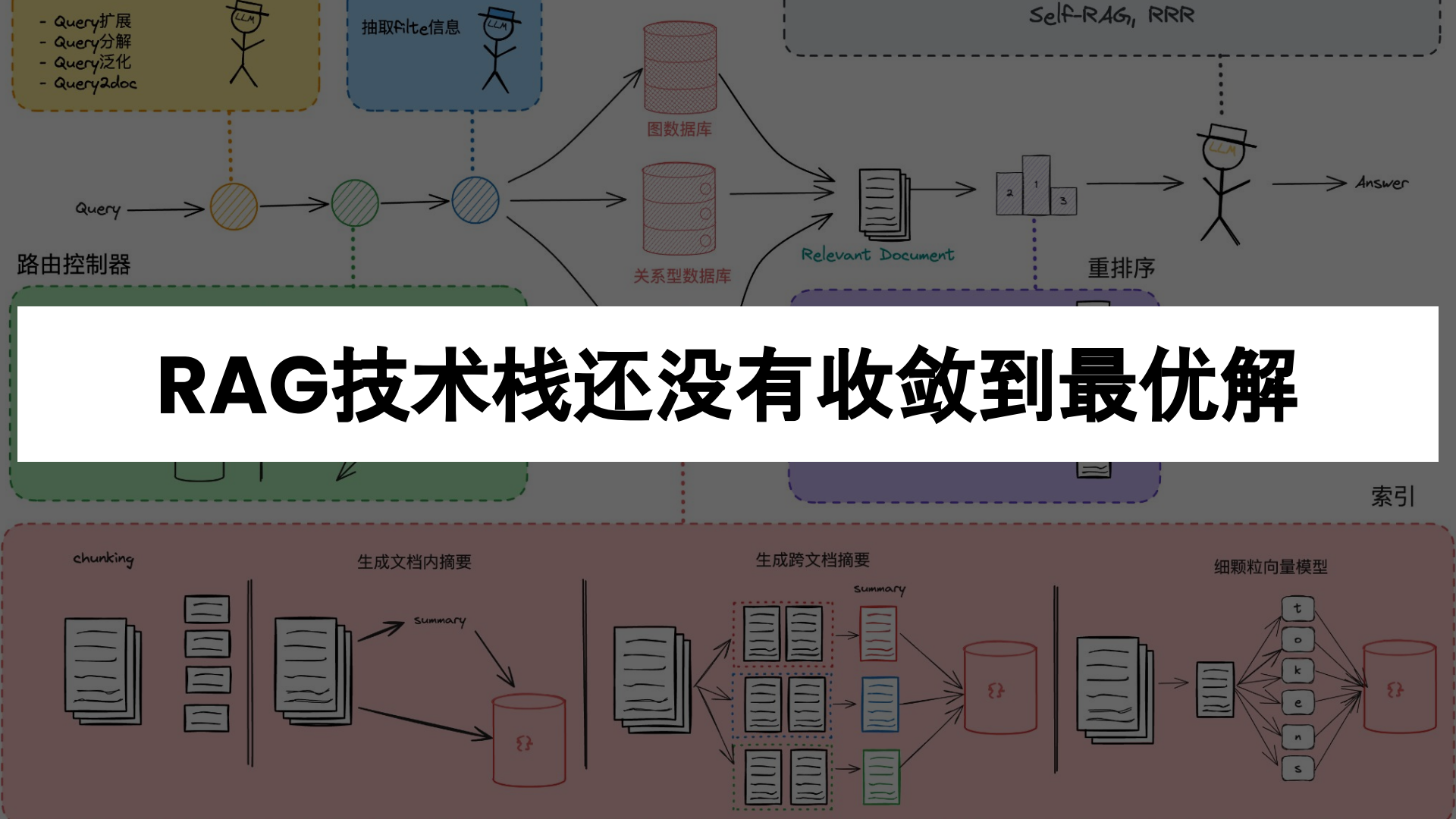


重排序-RankGPT

- 准确率高
- 计算量大
- 计算速度慢



The following are passages related to query {{query}}.
[1] {{passage_1}}
[2] {{passage_2}}
...
Rank these passages based on their relevance to the query.

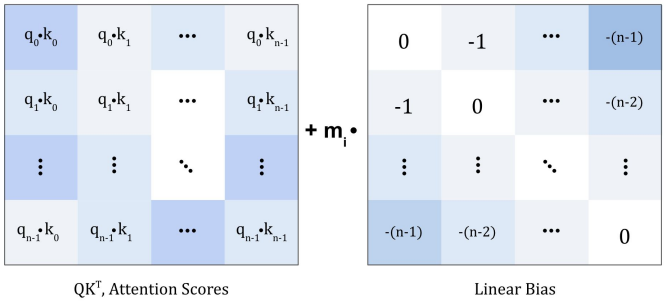


RAG技术栈还没有收敛到最优解

Jina AI的RAG工具

jina-embeddings-v2

- 2023年10月发布，全球第一个支持8k输入长度的开源向量模型
- 融合ALiBi，使用 750Gb 语料，预训练 jina-bert-v2

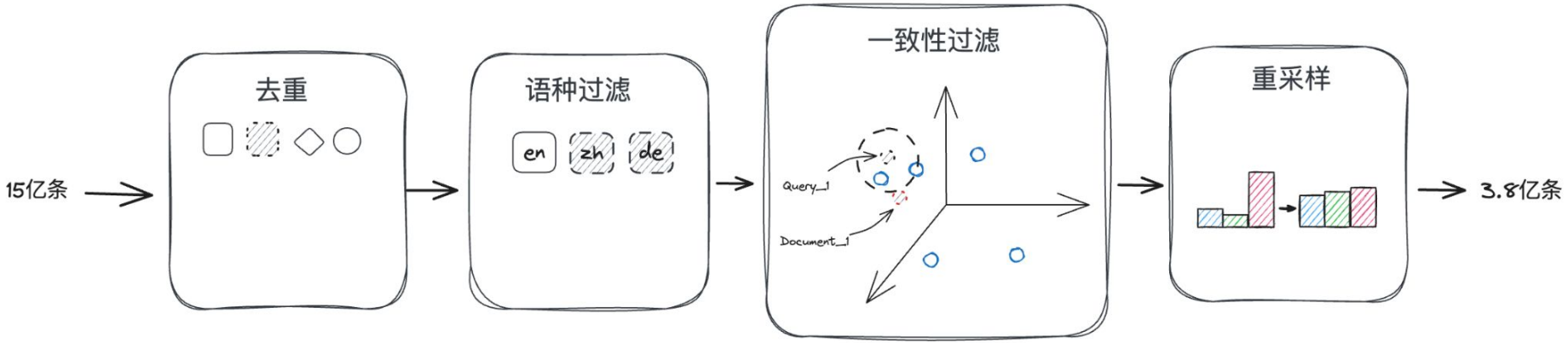


JINA EMBEDDINGS 2: 8192-Token General-Purpose Text Embeddings for Long Documents

Michael Günther, Jackmin Ong, Isabelle Mohr, Alaeddine Abdessalem, Tanguy Abel, Mohammad Kalim Akram, Susana Guzman, Georgios Mastrapas, Saba Sturua, Bo Wang, Maximilian Werk, Nan Wang and **Han Xiao**
Jina AI GmbH, Ohlauer Str. 43, 10999 Berlin, Germany
{michael.guenther, jackmin.ong, isabelle.mohr alaeddine.abdessalem,

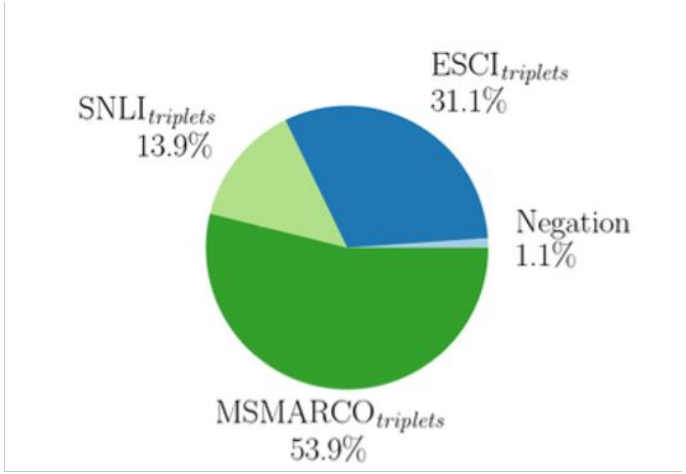
jina-embeddings-v2 弱监督学习

- 收集40+个开源数据源
  - 共15亿条文本对
- 3阶段数据清理
  - 得到3.8亿条高质量文本对，1700亿个token



jina-embeddings-v2 有监督学习

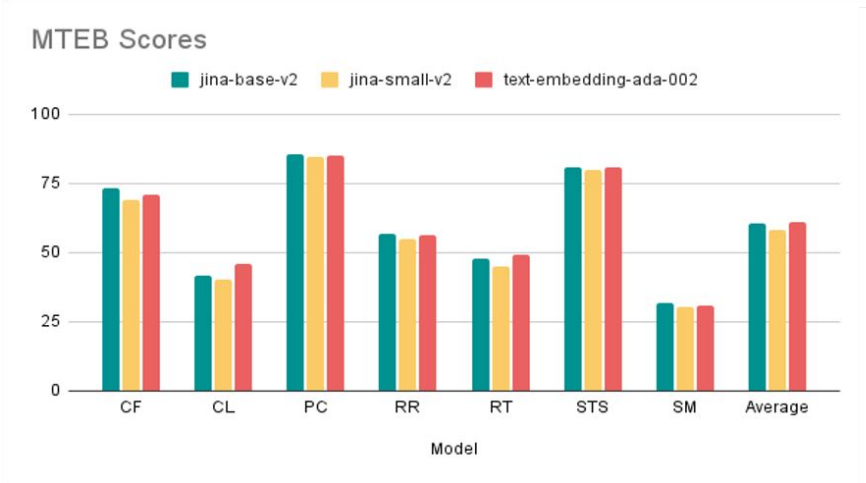
- 收集MSMarco, Natural Questions，NLI，fever, ESCI(EN)数据集
- 构造高质量正负样本三元组共300万条
  - 针对检索任务，使用Hard negative mining
  - (anchor, positive, negative_1, ..., negative_15)
- 尽可能增大batch size
  - 使用混合精度
  - 使用activation checkpoint
  - 基于DeepSpeed
  - 使用MiniBatch
  - 使用gradient caching

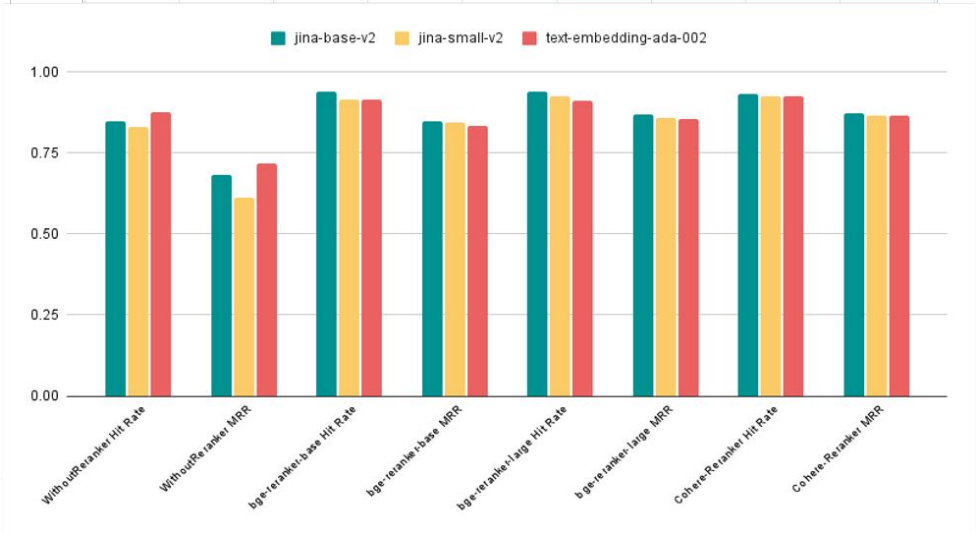


jina-embeddings-v2性能齐平ada-002

- 性能与ada-002齐平

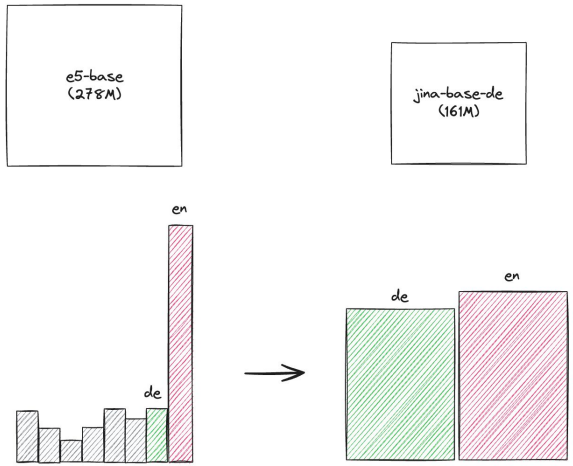
Embedding	WithoutReranker		bge-reranker-base		bge-reranker-large		Cohere-Reranker	
	Hit Rate	MRR	Hit Rate	MRR	Hit Rate	MRR	Hit Rate	MRR
OpenAI	0.876404	0.718165	0.91573	0.832584	0.910112	0.855805	0.926966	0.86573
bge-large	0.752809	0.597191	0.859551	0.805243	0.865169	0.816011	0.876404	0.822753
llm-embedder	0.814607	0.587266	0.870787	0.80309	0.876404	0.824625	0.882022	0.830243
Cohere-v2	0.780899	0.570506	0.876404	0.798127	0.876404	0.825281	0.876404	0.815543
Cohere-v3	0.825843	0.624532	0.882022	0.806086	0.882022	0.834644	0.88764	0.836049
Voyage	0.831461	0.68736	0.926966	0.837172	0.91573	0.858614	0.91573	0.851217
JinaAI-Small	0.831461	0.614045	0.91573	0.843071	0.926966	0.857303	0.926966	0.868633
JinaAI-Base	0.848315	0.68221	0.938202	0.846348	0.938202	0.868539	0.932584	0.873689
Google-PaLM	0.865169	0.719476	0.910112	0.833708	0.910112	0.85309	0.910112	0.855712





Jina-embeddings-v2双语模型

- 避免英文语料的偏差
- 避免多语言模型过大的词表
- 针对不同任务使用不同的损失函数
- 目标语种效果优于多语言模型



Multi-Task Contrastive Learning for 8192-Token Bilingual Text Embeddings

Isabelle Mohr, Markus Krimmel, Saba Sturua, Mohammad Kalim Akram, Andreas Koukounas, Michael Günther, Georgios Mastrapas, Vinit Ravishankar, Joan Fontanals Martínez, Feng Wang, Qi Liu, Ziniu Yu, Jie Fu, Saahil Ognawala, Susana Guzman, Bo Wang, Maximilian Werk, Nan Wang and **Han Xiao**
Jina AI GmbH, Ohlauer Str. 43, 10999 Berlin, Germany
research@jina.ai

Abstract

We introduce a novel suite of state-of-the-art bilingual text embedding models that are designed to support English and another target language. These models are capable of processing lengthy text inputs with up to 8192 tokens, making them highly versatile for a range of natural language processing tasks such as text retrieval, clustering, and semantic textual similarity (STS) calculations.

support tens of languages. This is achieved by fine-tuning pre-existing multilingual backbones like XLM-RoBERTa [Conneau et al., 2020] on mainly English data.[1] Alternatively, multilingual knowledge distillation can be applied to align embedding models across various languages using parallel data [Reimers and Gurevych, 2020] to cope with the scarcity of high-quality semantic pairs or triplets in the target languages. Despite these efforts, training data with such an extremely un-

Lang.	Model	CF	CL	PC	RR	RT	STS**	SM
en	jina-de-base	0.688	0.403	0.835	0.549	0.441	0.820	**0.318***
	jina-es-base	0.690	**0.405**	**0.846**	**0.553**	0.464	**0.835**	0.299*
	multilingual-e5-base	**0.730**	0.400	0.836	0.548	**0.489**	0.803	0.301*
de	jina-de-base	0.665	0.299	**0.583***	0.639	**0.387**	**0.714**	–
	multilingual-e5-base	**0.687**	**0.328**	0.541*	**0.648**	0.342	0.674	–
es	jina-es-base	0.671	**0.440**	**0.583***	**0.739**	**0.511**	**0.788**	–
	multilingual-e5-base	**0.685**	0.413	0.541*	0.736	0.478	0.783	–

CF: Classification Accuracy CL: Clustering $\mathcal{V}$ measure PC: Pair Classification Average Precision
RR: Reranking MAP RT: Retrieval nDCG@10 STS: Sentence Similarity Spearman Correlation
SM: Summarization Spearman Correlation

jina-colbert-v1

- 第一款支持8k长度的ColBERT模型

长文本上效果优于ColBERTv2

Model	Used context length	Model max context length	Avg. NDCG@10
ColBERTv2	512	512	74.3
Jina-ColBERT-v1 (truncated)	512*	8192	75.5
Jina-ColBERT-v1	8192	8192	83.7
Jina-embeddings-v2-base-en	8192	8192	**85.4**

短文本MSMARCO，与ColBERTv2齐平

dataset	ColBERTv2	Jina-ColBERT-v1
ArguAna	46.5	49.4
ClimateFEVER	18.1	19.6
DBPedia	45.2	41.3
FEVER	78.8	79.5
FiQA	35.4	36.8
HotPotQA	67.5	65.6
NFCorpus	33.7	33.8
NQ	56.1	54.9
Quora	85.5	82.3
SCIDOCS	15.4	16.9
SciFact	68.9	70.1
TREC-COVID	72.6	75.0
Webis-touché2020	26.0	27.0
Average	50.0	50.2

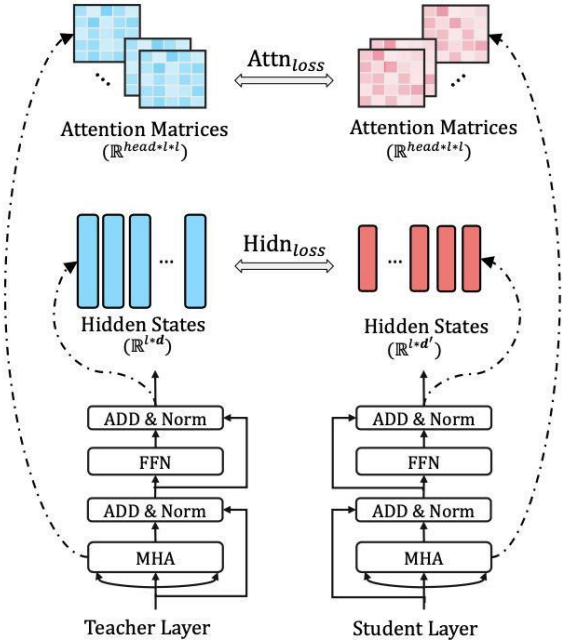
jina-reranker-v1

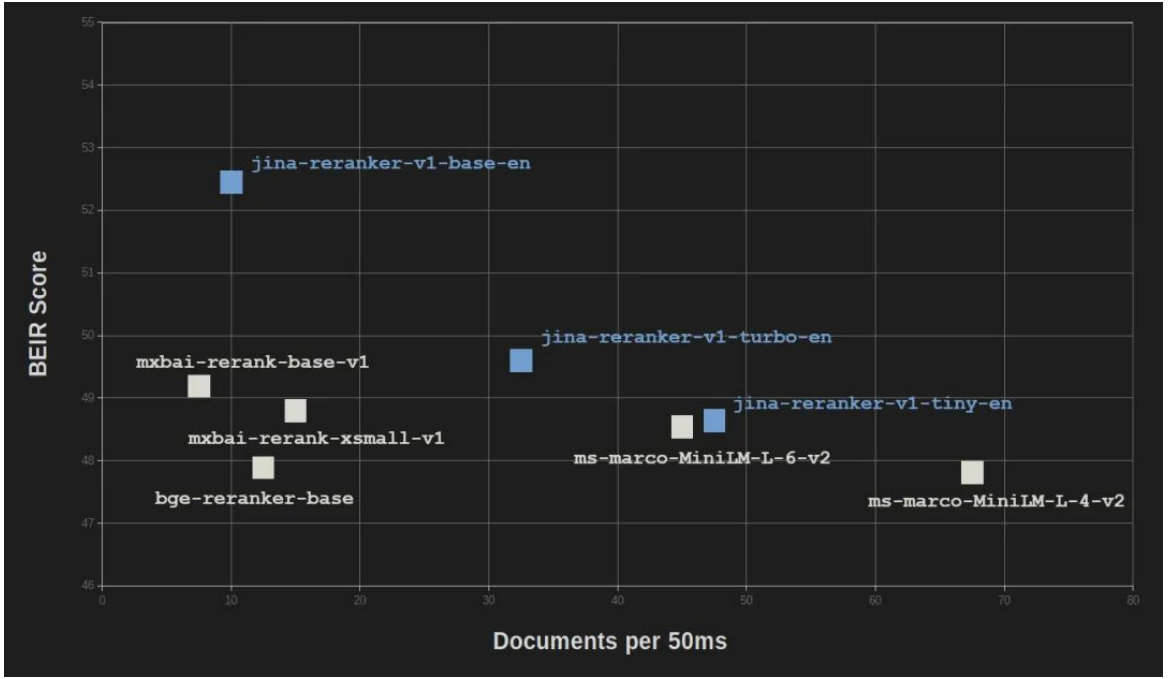
- 预训练：基于Jina BERT v2，支持 8K 上下文输入；
- 分阶段训练：逐步提升模型排序能力；
- 迁移学习：将Embedding模型学习到的知识迁移到Reranker模型；
- 训练数据：使用和Embedding模型相同来源的训练数据；

	No Reranker		jina-reranker		bge-reranker-base		bce-reranker-base_v1		cohere-reranker	
Embedding model	Hit Rate	MRR	Hit Rate	MRR	Hit Rate	MRR	Hit Rate	MRR	Hit Rate	MRR
jina-embeddings-v2-base-en	0.8053	0.5156	0.8737	**0.7229**	0.8368	0.6568	0.8737	0.7007	**0.8842**	0.7008
bge-base-en-v1.5	0.7842	0.5183	**0.8368**	**0.6895**	0.8158	0.6586	0.8316	0.6843	0.8368	0.6739
bce-embedding-base_v1	0.8526	0.5988	0.8895	0.7346	0.8684	0.6927	0.9157	**0.7379**	**0.9158**	0.7296
CohereV3-en	0.7211	0.4900	0.8211	**0.6894**	0.8000	0.6285	0.8263	0.6855	**0.8316**	0.6710
Average	0.7908	0.5307	0.8553	**0.7091**	0.8303	0.6592	0.8618	0.7021	**0.8671**	0.6938

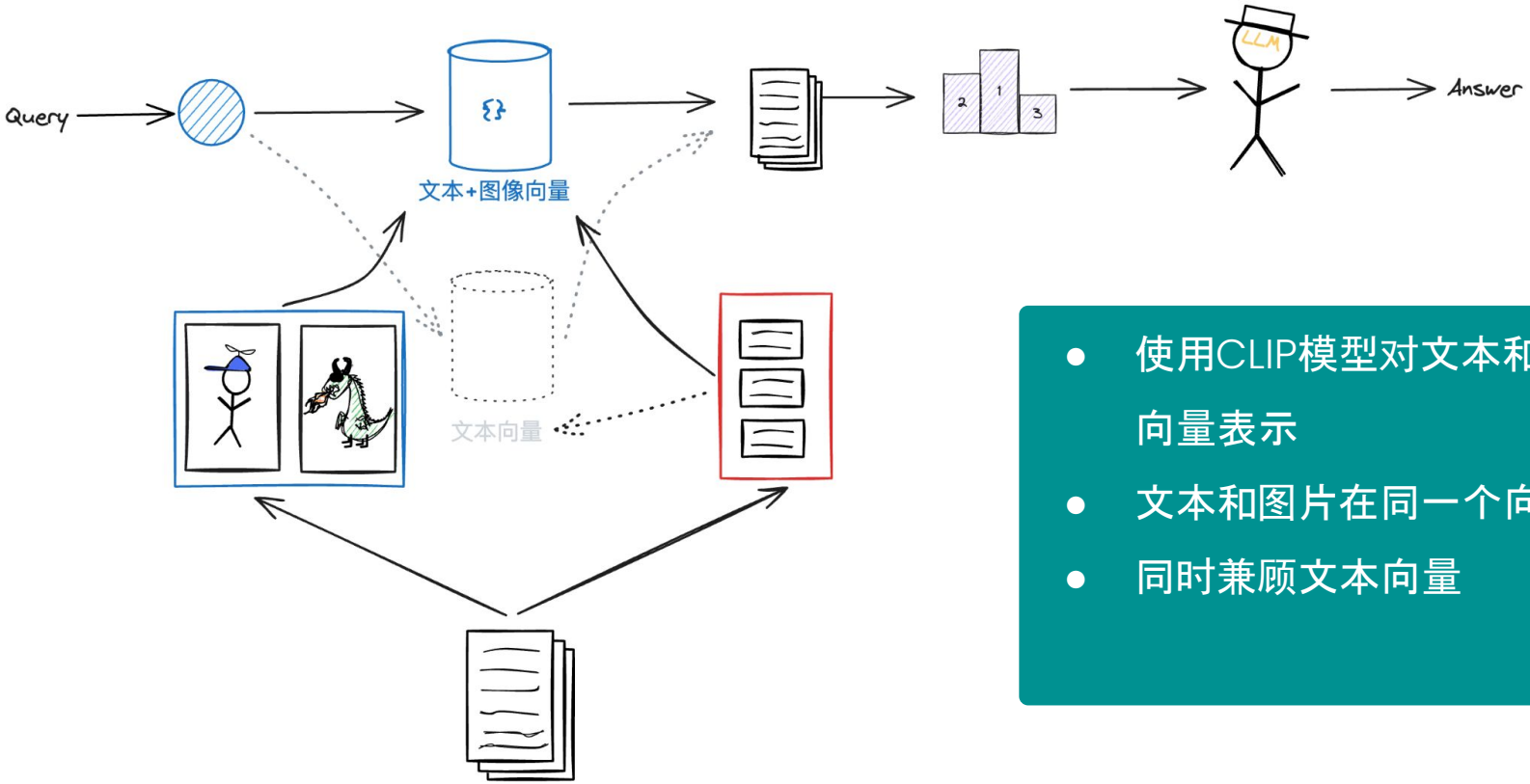
jina-reranker-v1-turbo/tiny

- 使用模型蒸馏技术，平衡准确率和推理速度



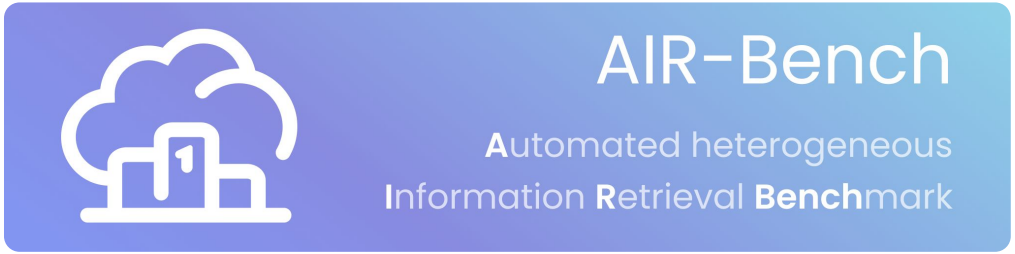


jina-CLIP-v1 (准备发布中...)



- 使用CLIP模型对文本和图片进行向量表示
- 文本和图片在同一个向量空间
- 同时兼顾文本向量

AIR-Bench：自动化的多样性信息检索评测基准



https://huggingface.co/spaces/AIR-Bench/leaderboard

标注成本高

- 使用LLMs生成测试数据
- 通过向量模型、排序模型、LLMs组合进行质量把控
- 覆盖多领域和多语言

自动生成评估数据

面向多种任务

- 首次提出长文档内检索
- 专注问答任务

针对RAG场景设计

数据泄露

- 基于真实世界的语料库进行生产
- 定期进行更新，满足社区不断变化的评测需求

便于更新和扩展

我们是否还需要RAG?


RAG
GPT-4
幻觉
知识更新难
私有知识注入难
推理成本高
GPT
-42
如果LLMs可以精确地召回记忆
如果LLMs可以有无限长的context
如果LLMs可以在本地完成推理，
私有数据不会离开本地
如果LLMs的推理速度比人类阅读速度
快且成本低

模型 != 产品

模型 != 产品

产品需要解决用户实际痛点

- 刷榜!=用户认可
- Storytelling很重要

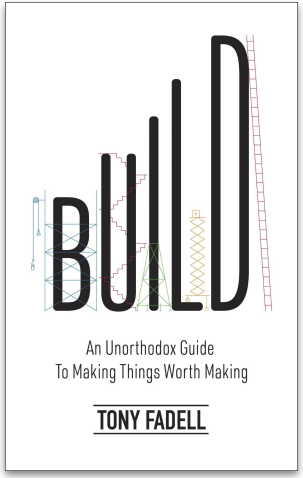
模型只是Jina AI提供的一种工具

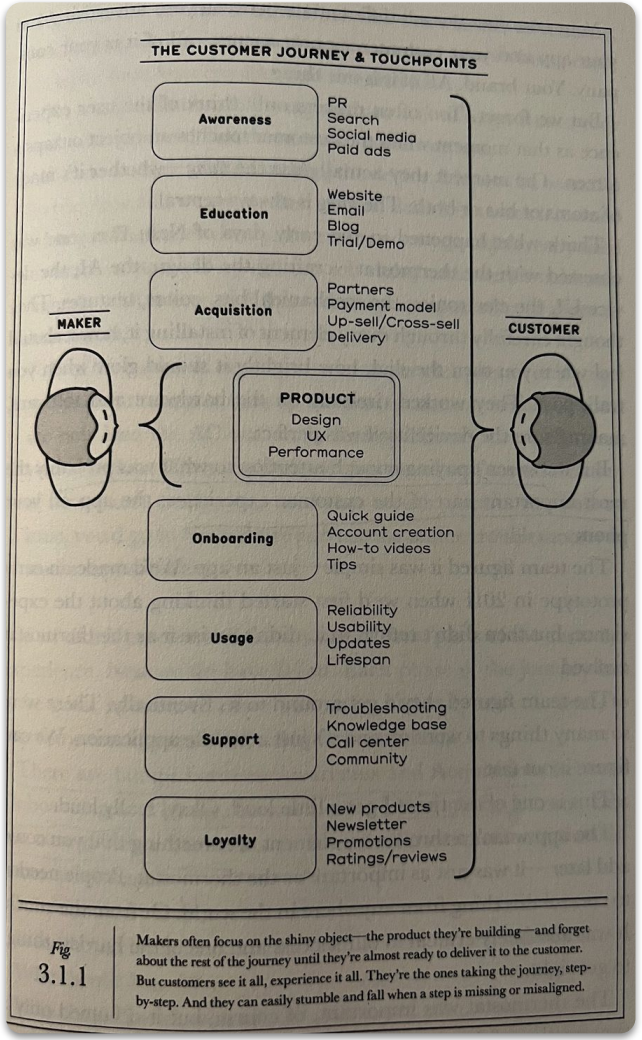
魔鬼隐藏在细节

- 思考每一步用户体验

模型需要迭代

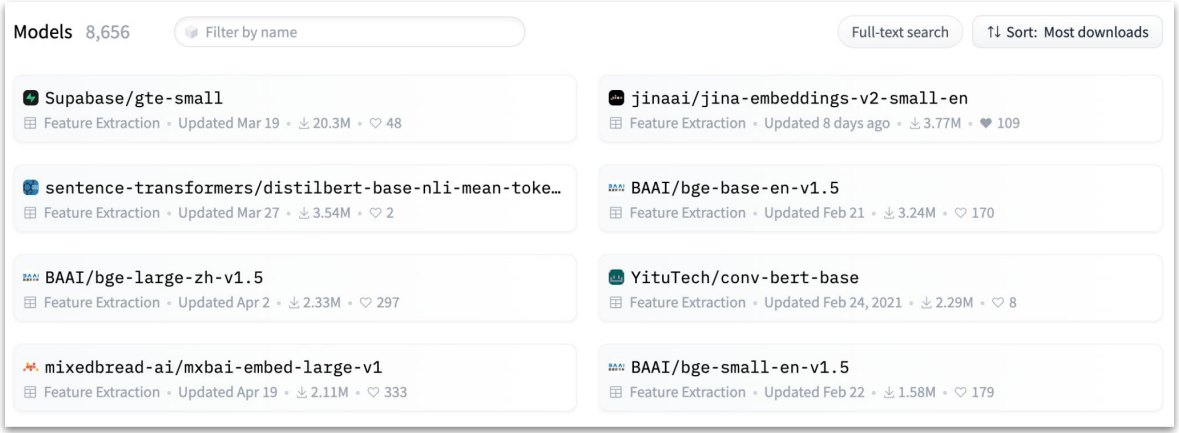
- 需求不断变化





Jina AI的模型产品

- HF下载量
  - 10个月，1000万下载量
- 社区反馈
  - HackerNews当日第一
  - llamaIndex评测第一



Hello Jina Team,

I see that you exposed the Jina base embedding models on Ollama. That is very good. Unfortunately, I would like to use the small version of it and most importantly the *jina-embeddings-v2-small-en* one.

I would like to use Meilisearch and the dataset from this article to do a great search demo. The embedding model used for the Hackernews articles is the small one.

Do you think you could upload this model, please?

Have a good week,
kero, CTO of Meilisearch

搞AI也必须尊重商业逻辑

不能忽略的成本

Scaling Laws

- 投入多少钱
- 花多长时间
- 预期什么效果

FLOPs

- 预估模型训练时间
- 合理规划计算资源
- 排查故障

Head-first Estimation

$$T \approx \frac{6 * t * (P - H * C)}{n * X} * (1 + \frac{S}{6 * H} + \frac{C}{12 * L * H})$$

where t is the total number of tokens in the training corpus. P is the total number of parameters in the XLM-R model.

Training time estimation for XLM-RoBERTa

Take our current training XLM-R-base on 8*A100 as an example,

		XLM-RoBERTa-base
# of parameters	P	270M
sequence length	S (we used in our settings)	8192
# of hidden states	H	768
# of layers	L	12
size of the vocabulary	C	250k
# of training tokens	t	420B
# of GPUs	n	8
FLOPS of GPU	X	312 T * 57% = 180T

X: 312 T FLOPs * 57% = 180T FLOPs. Due to the memory and network limitation, 57% utilization ratio is commonly observed.

$$\frac{6 * 420 * 10^9 * (270 * 10^6 - 768 * 250 * 10^3)}{8 * 312 * 10^{12} * 0.57} * (1 + \frac{8192}{6 * 768} + \frac{250 * 10^3}{12 * 12 * 768})$$
$$= 696086\ seconds \approx 193.4\ hours$$

.jina

不要高估6个月后的变化，
不要低估18个月后的变化。
在AI浪潮中，找到自己不变的价值。