AiDD AI+ 软件研发数字峰会
AI+ software Development Digital summit

AI驱动软件研发
全面进入数字化时代

中国·深圳 11.24-25

AI+
software
Development
Digital
summit

知识增强大模型：垂域落地的最后一公里

演讲人 王昊奋 同济大学

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

KEYLINK ing

K+峰会

K+峰会 深圳站

K+ 全球软件研发行业创新峰会

会议时间：2024.05.24-25

K+峰会 上海站

K+ 全球软件研发行业创新峰会

会议时间：2024.09.20-21

AiDD 峰会

AiDD峰会 深圳站

AI+ 软件研发数字峰会

会议时间：2023.11.24-25

AiDD峰会 北京站

AI+ 软件研发数字峰会

会议时间：2024.07.19-20

AiDD峰会 深圳站

AI+ 软件研发数字峰会

会议时间：2024.11.15-16

演讲嘉宾

王昊奋

同济大学特聘研究员 / OpenKG发起人之一

中国计算机学会（CCF）技术前沿委员会知识图谱SIG主席、自然语言处理专委会秘书长；中国中文信息学会（CIPS）理事、语言与知识计算专业委员会副秘书长；上海市计算机学会青年工作委员会副主任。

研究方向：知识图谱、自然语言处理、智能内容生成业。 腾讯云最具价值专家TVP，中国中文信息学会理事，畅销书《知识图谱方法、实践与应用》的作者，曾作为2家AI独角兽企业的CTO； 具有超过16年的知识图谱研发和技术管理经验。

目录

CONTENTS

PART 01
知识检索增强的基本概述

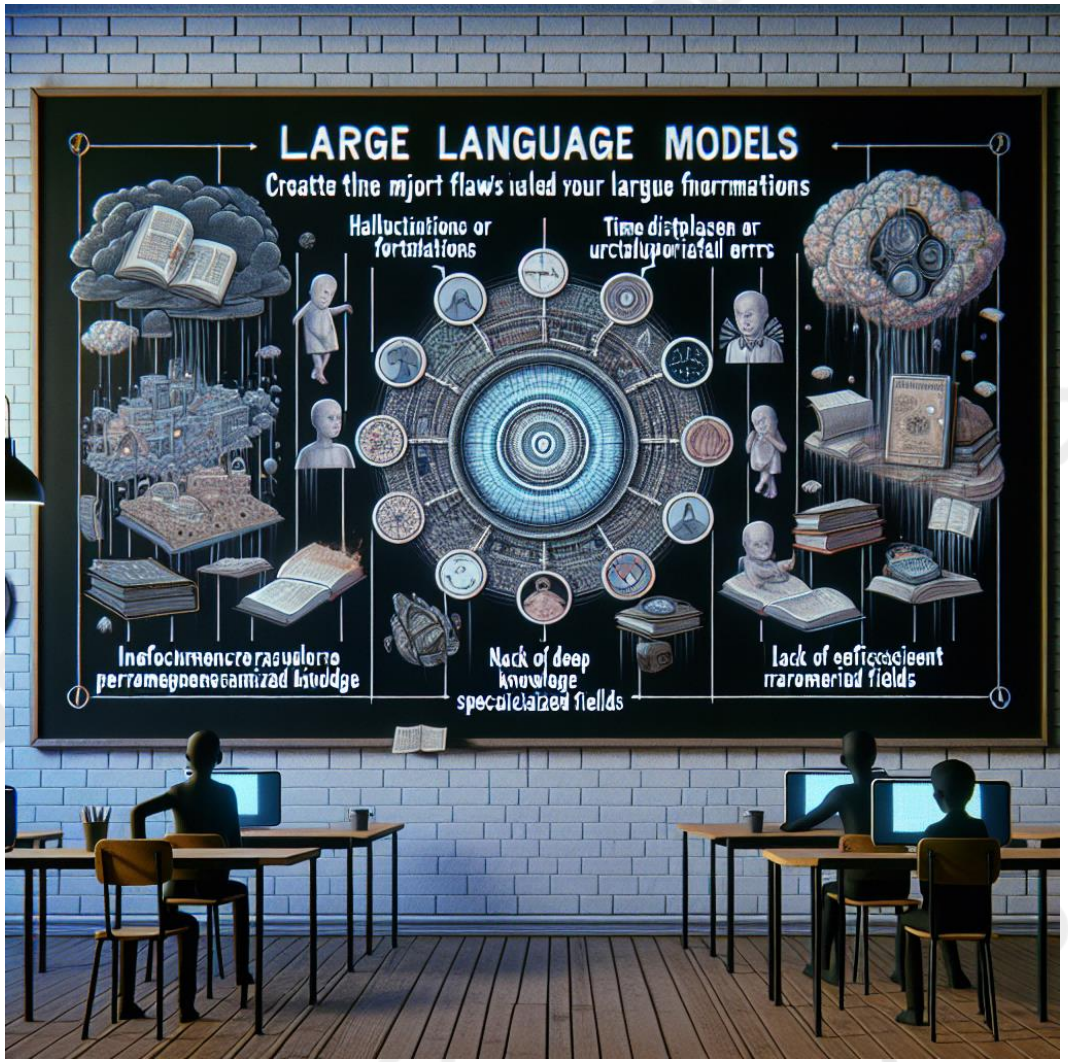
知识检索增强技术提出的背景

LLM的缺陷

- 幻觉
- 信息过时
- 参数化知识效率低
- 缺乏专业领域的深度知识
- 推理能力弱

实际应用的需求

- 领域精准问答
- 数据频繁更新
- 生成内容可解释可溯源
- 成本可控
- 数据隐私保护



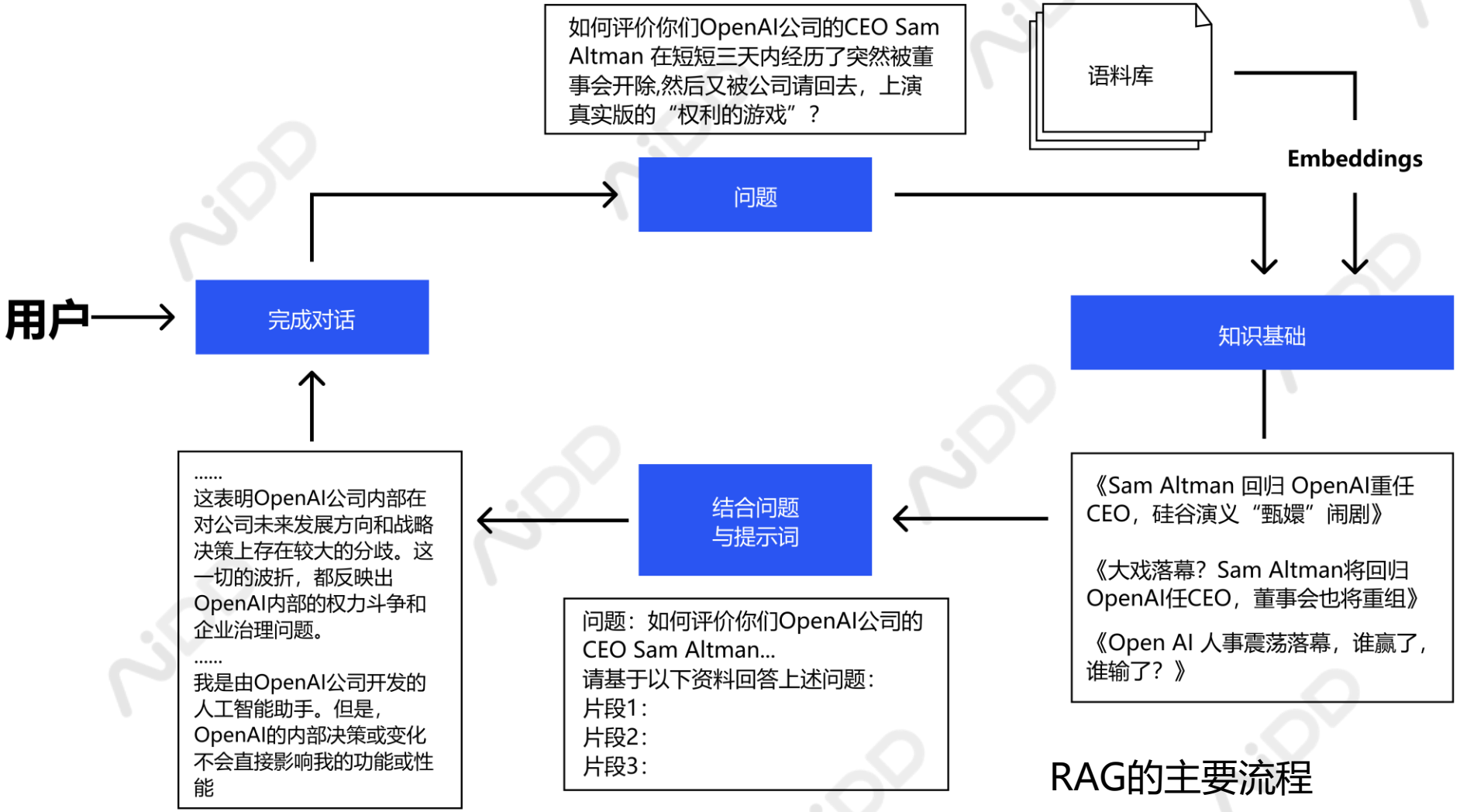
由 OpenAI DALL·E 3 生成

检索增强生成（Retrieval-Augmented Generation，RAG）

LLM 在回答问题或生成文本时，先会从大量文档中检索出相关的信息，然后基于这些信息来生成回答。

RAG 方法使得不必为每一个特定的任务重新训练整个大模型，只需要外挂知识库。

RAG模型尤其适合知识密集型的任务。



RAG的主要流程

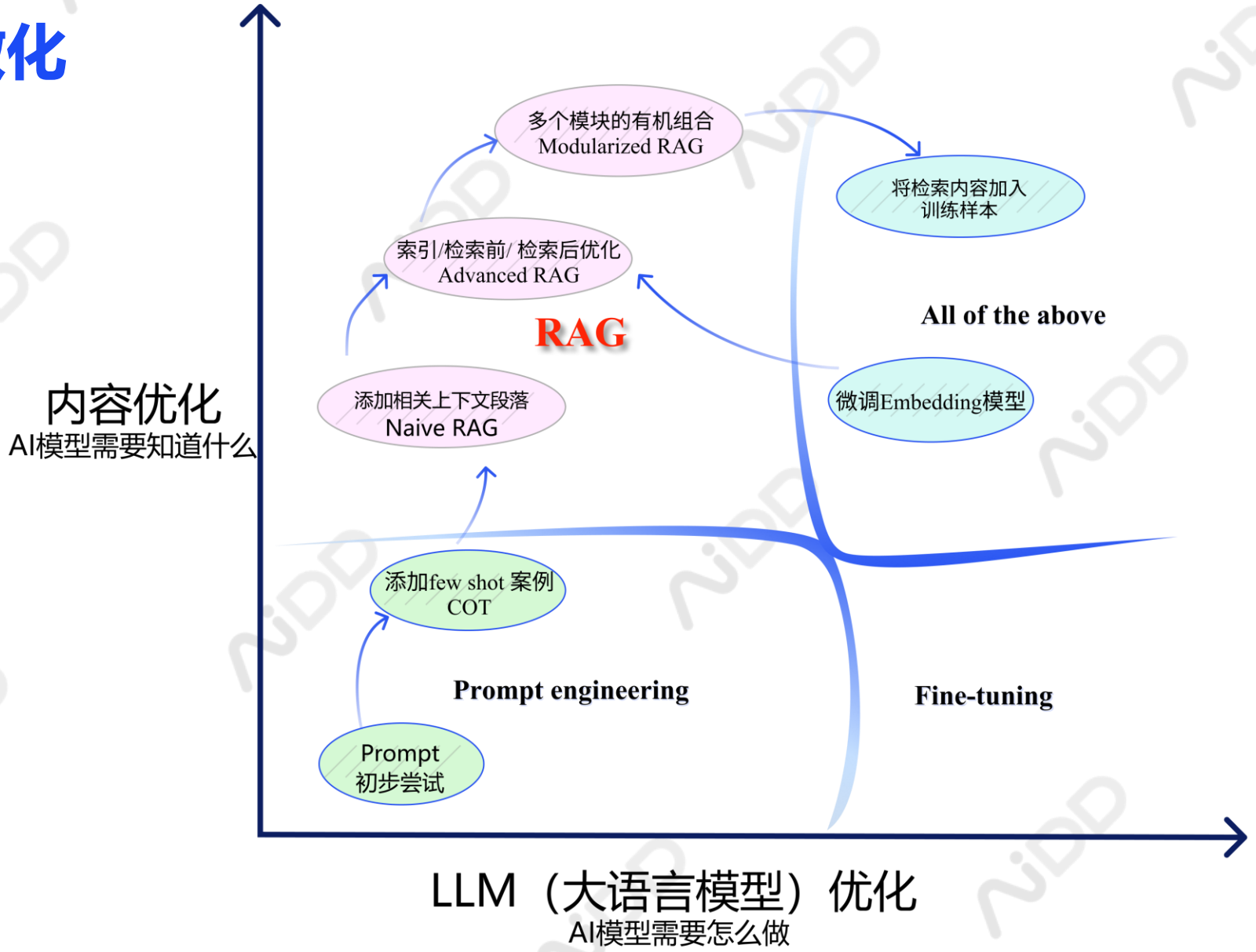
外挂知识库 vs 知识参数化

大模型优化的方式

提示工程
Prompt Engineering

检索增强
Retrieval-Augmented Generation

（指令）微调
Instruct / Fine-tuning



RAG vs Fine-tuning

	RAG	Fine-tuning
知识更新	直接更新检索知识库，适合动态数据环境	重新微调训练，保持更新需要大量资源
训练数据的要求	对数据加工和处理的要求低	微调依赖高质量数据集，有限的数据集可能不会产生显著性能改善
可解释性（可溯源性）	通常可以追溯到特定数据源的答案，从而提供更高等级的可解释性和可溯源性	微调就像黑匣子，并不总是清楚模型为何会做出这样的反应，相对较低的可解释性
可扩展性	高，可以动态衔接不同的数据源	低，扩展新知识需要重新微调训练
耗时	由于（多次）数据检索可能会有更高延迟	经过微调的 LLM 无需检索即可响应
外部知识利用	擅长利用外部资源，适合文档或其他结构化/非结构化数据库	需要构造监督数据集以内化外部知识，不适用频繁更改的数据源

知识检索增强的使用场景

RAG适用的情况：

- 数据长尾分布
- 知识更新频繁
- 回答需要验证追溯
- 领域专业化知识
- 数据隐私保护

问答
RETRO (Borgeaud et al2021)
REALM (Gu et al, 2020)
ATLAS (lzacard et al, 2023)

事实验证
RAG (Lewis et al, 2020)
ATLAS (lzacard et al, 2022)
Evi. Generator (Asai et al, 2022a）

对话
BlenderBot3 (Shuster et al.2022)
Internet-augmented generation
(Komeili et a., 2022)

总结
FLARE (Jiang et al, 2023)

机器翻译
kNN-MT (Khandelwal et al., 2020)TRIME-MT (Zhong et al., 2022)

代码生成
DocPrompting (Zhou et al., 2023
Natural ProverWelleck et al., 2022)

自然语言推理
kNN-Prompt (Shi et al., 2022)
NPM (Min et al., 2023)

情感分析
kNN-Prompt (Shi et al., 2022)NPM (Min et al., 2023)

常识推理
Raco (Yu et al, 2022)

PART 02

知识检索增强技术的主要范式与发展历程

RAG的典型范式（Naive RAG）

步骤1：构建数据索引：

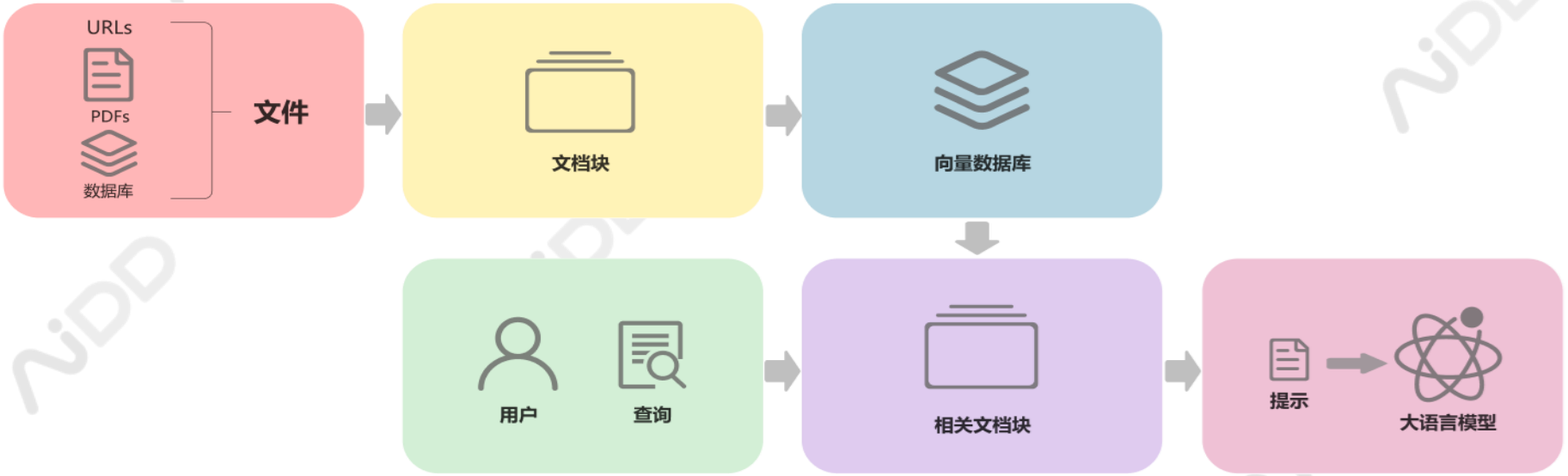
1. 将文档分割成均匀的块。每个块是一段原始文本。
2. 利用编码模型为每个文本块生成Embedding
3. 将每个块的Embedding存储到向量数据库中。

步骤2：检索

通过向量相似度检索和问题最相关的K个文档。

步骤3：生成

原始Query与检索得到的文本组合起来输入打语言模型，得到最终的回答。



朴素RAG
Naive RAG

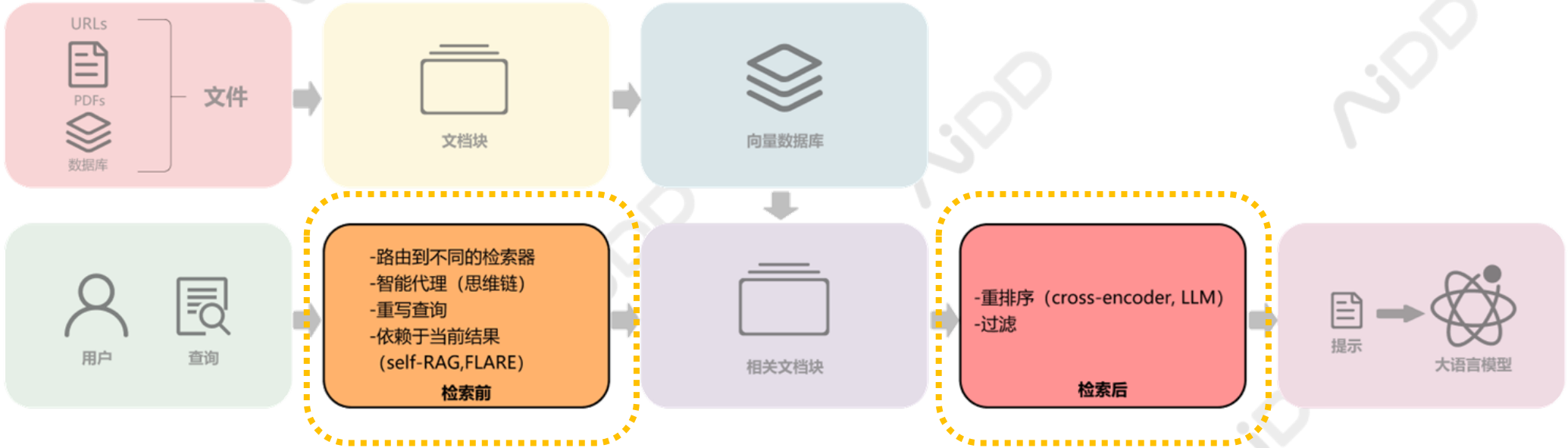
RAG进阶
Advanced RAG

模块化RAG
Modularized RAG

RAG的典型范式（Dynamic/Advanced RAG）

索引优化 -> **前检索** -> 检索 -> **后检索** -> 生成

- **索引优化**：滑动窗口、细粒度分割、元数据
- **前检索模块**：检索路由、摘要、重写、置信度判断
- **后检索模块**：重排序、检索内容过滤

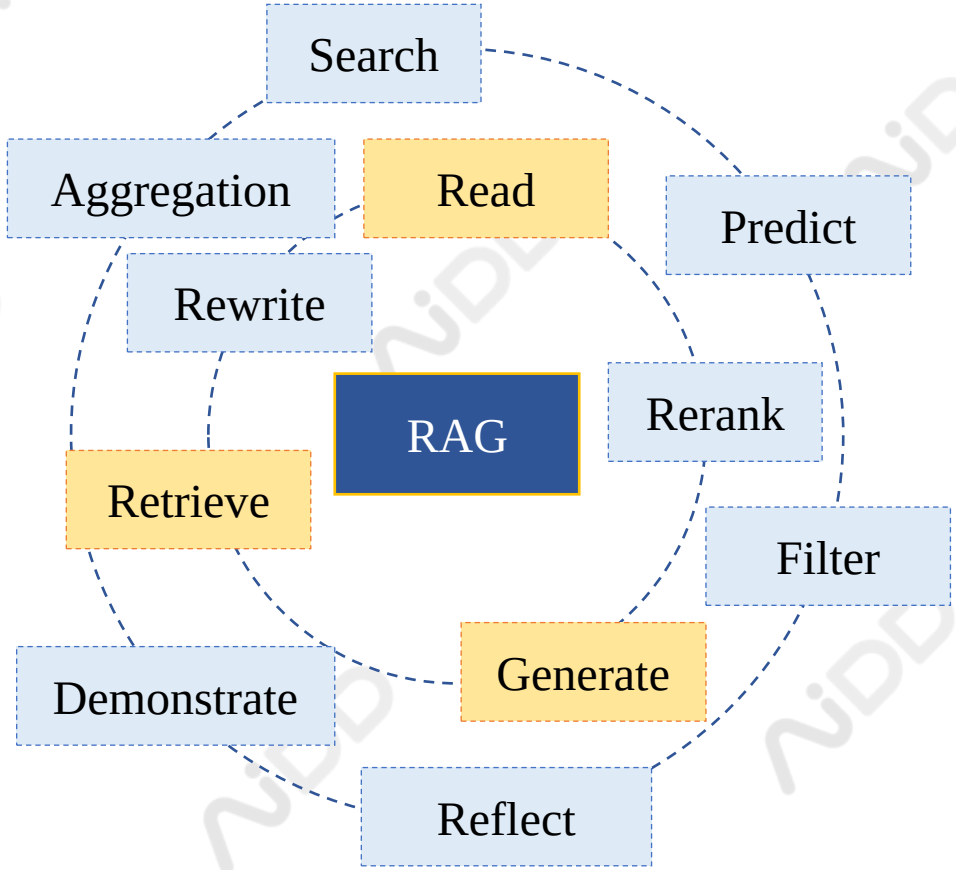


朴素RAG
Naive RAG

RAG进阶
Advanced RAG

模块化RAG
Modularized RAG

模块化RAG（Modularized RAG）



典型Pattern			
Naive RAG	Read	Retrieve	Generate
DSP (2022)	Demonstrate	Search	Predict	Generate
Rewrite-Retrieve-Read (2023)	Rewrite	Retrieve	Read
Retrieve-then-read (2023)	Retrieve	Read	Generate

。。。。。。

朴素RAG
Naive RAG

RAG进阶
Advanced RAG

模块化RAG
Modularized RAG

RAG的三大灵魂拷问

检索什么?

- 词元
- 词组
- 句子
- 段落
- 实体
- 知识图谱

什么时候检索?

- 单次检索
- 每个Token
- 每 N个Token (Phrase)
- 自适应检索

怎么使用检索的结果?

- 输入/数据层
- 模型/中间层
- 输出/预测层

其他问题

在什么阶段增强?

- 预训练
- （指令）微调
- 推理

检索器选择?

- BERT
- Roberta
- BGE
- .....

模型协同

规模选择

生成器选择?

- GPT
- Llama
- T5
- .....

RAG的关键问题——检索什么?

粗

检索粒度

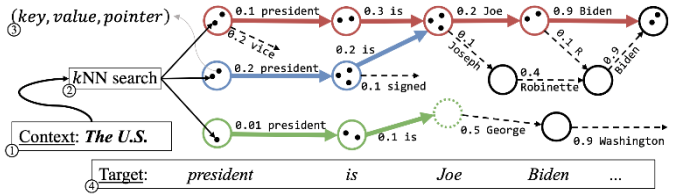
细

低

结构化程度

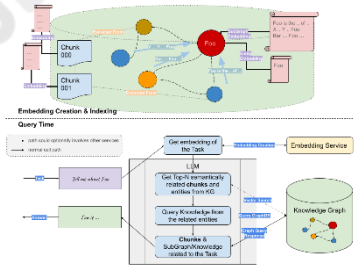
高

Chunk | In-Context RAG 2023



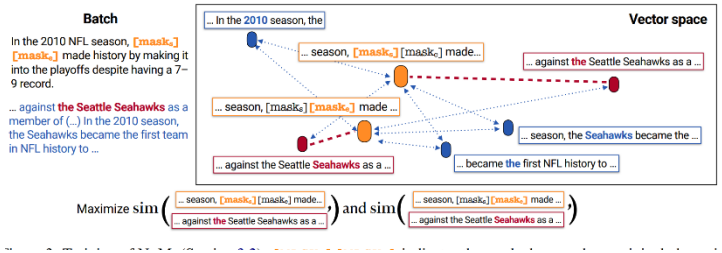
检索粒度粗，召回信息量大，精确度低，覆盖率高，冗余信息多

Knowledge Graph | 2023

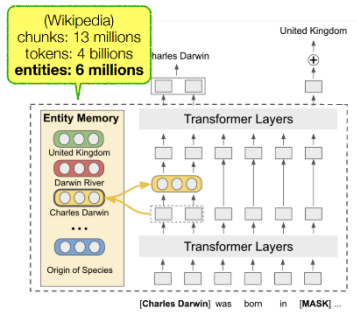


提供更丰富语义和结构化信息，检索效率更低，受限于KG质量

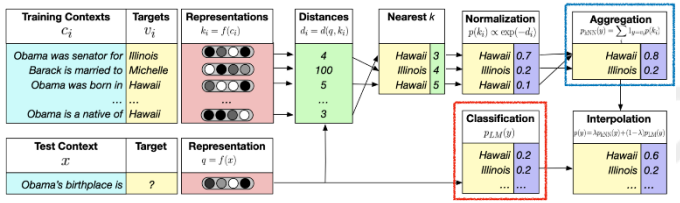
Phrase | NPM 2023



Entity | EasE 2022



Token | KNN-LMM 2019

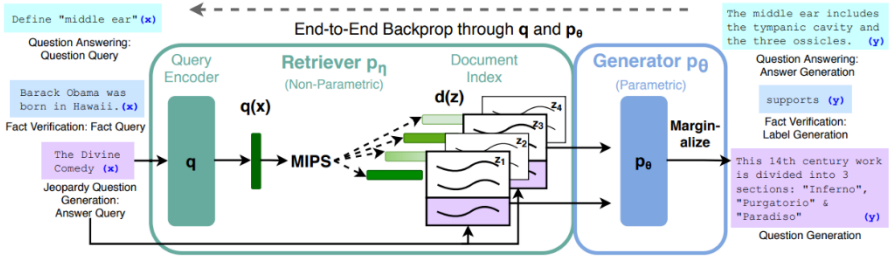


在长尾问题、跨领域问题上更有优势，计算效率高，存储消耗大

RAG的关键问题——如何使用检索内容

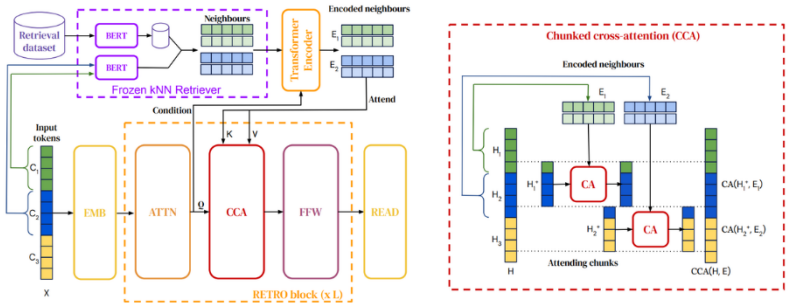
在推理过程中，集成检索到的信息到生成模型的不同层级中

集成检索位置



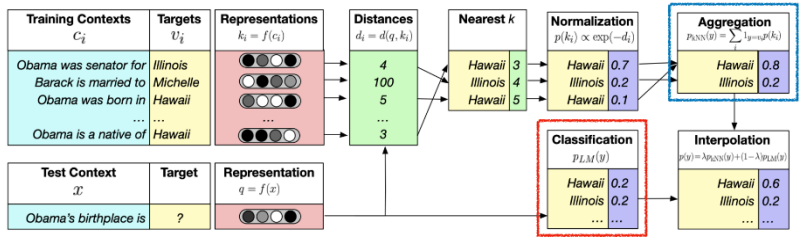
输入/数据层

使用简单，但无法支持检索更多的知识块，且优化空间有限



模型/中间层

支持输入更多的知识块检索，但引入额外的复杂度，且必须训练



输出/预测层

保证输出结果与检索内容高度相关但效率低

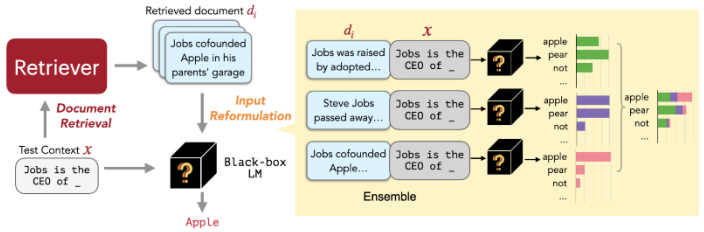
RAG的关键问题——什么时候检索

效率高，检索到的文档相关度低

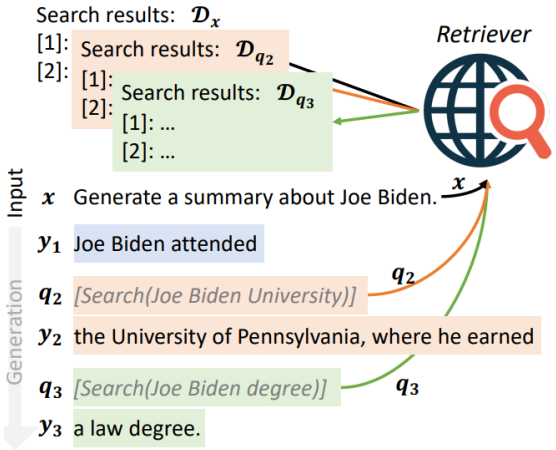
平衡效率和信息的矛盾
可能非最优解

检索到的信息量大，但效率低，冗余信息多

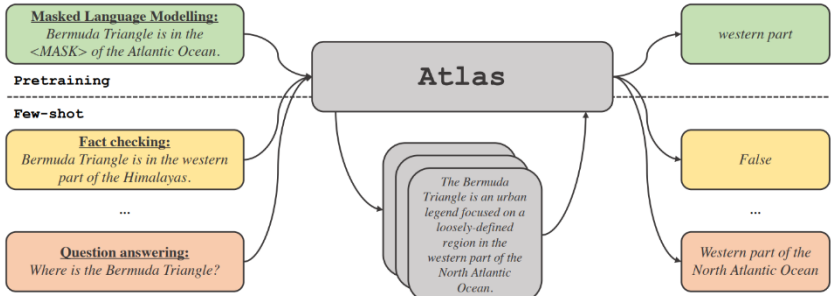
Once | Replug 2023



Adaptive | Flare 2023



Every N Tokens | Atlas 2023



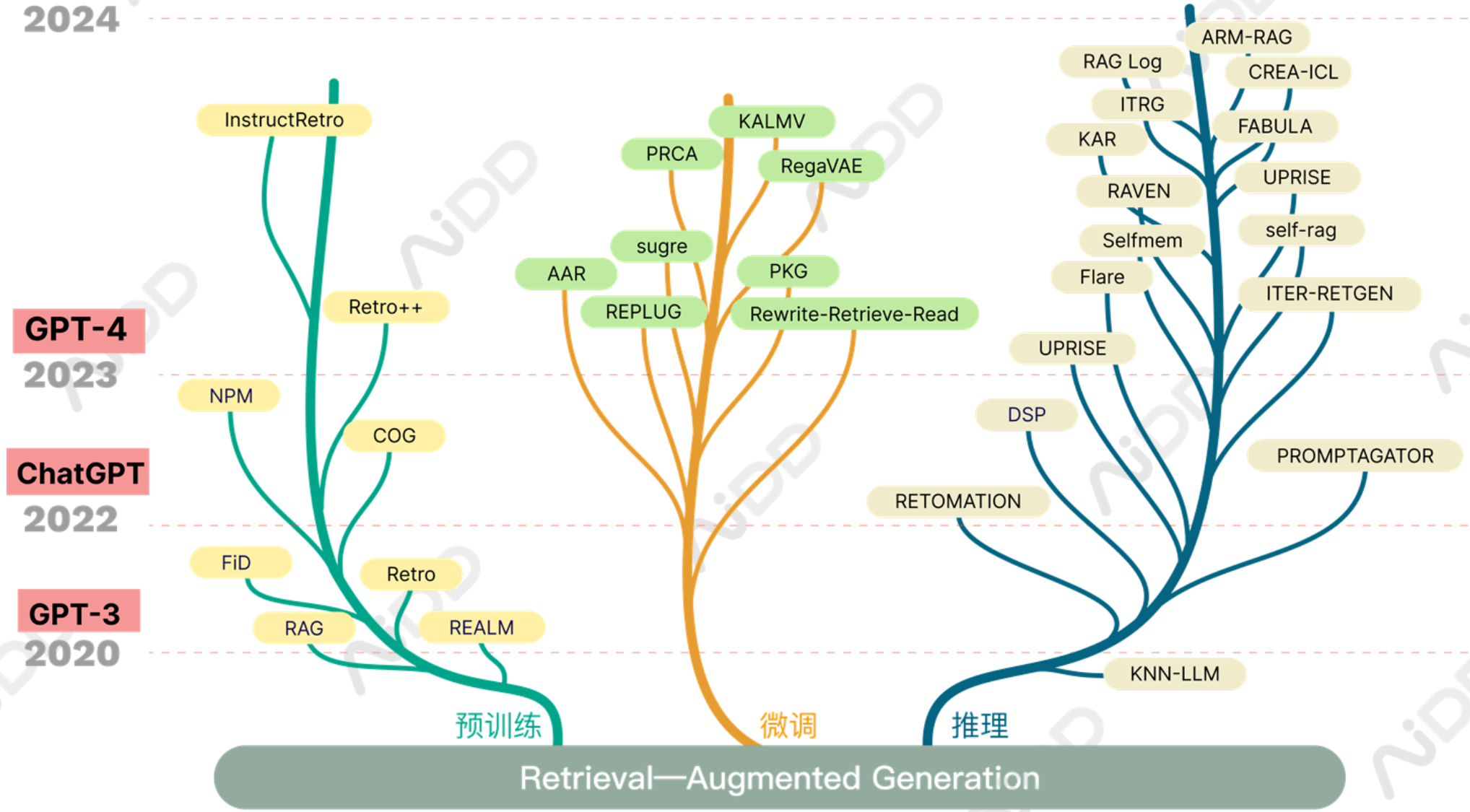
在推理中仅进行一次检索

自适应地进行检索

每生成N个Tokens去检索一次

低 —— 检索频率 —— 高

RAG发展历程总览


2024
GPT-4
2023
ChatGPT
2022
GPT-3
2020
InstructRetro
Retro++
NPM
COG
FiD
Retro
RAG
REALM
KALMV
PRCA
RegaVAE
sugre
AAR
PKG
REPLUG
Rewrite-Retrieve-Read
ARM-RAG
RAG Log
CREA-ICL
ITRG
FABULA
KAR
UPRISE
RAVEN
Selfmem
self-rag
Flare
ITER-RETGEN
UPRISE
DSP
PROMPTAGATOR
RETOMATION
KNN-LLM
预训练
微调
推理
Retrieval—Augmented Generation

PART 03

知识检索增强的关键技术与效果评估

Techniques for Better RAG ——检索内容优化

索引优化

Small-2-Big

在句子级别嵌入文本，然后在LLM生成过程中扩大窗口

滑动窗口

滑动Chunk覆盖全文，避免语义割裂

摘要

通过摘要嵌入更大的文档。通过摘要检索文档，再从文档中检索文本块

添加元数据

示例　页码　时间　类型　文档标题

元数据筛选/扩充

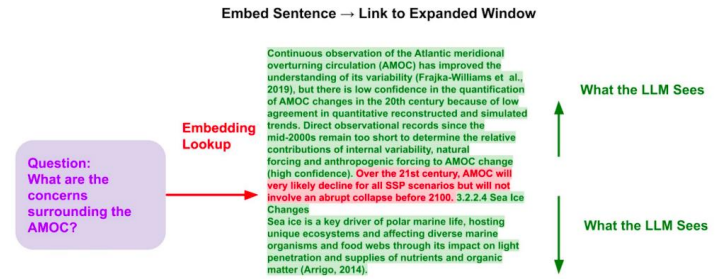
伪元数据生成

通过为传入的查询生成一个假设性的文档来增强检索，并生成该文本块可以回答的问题

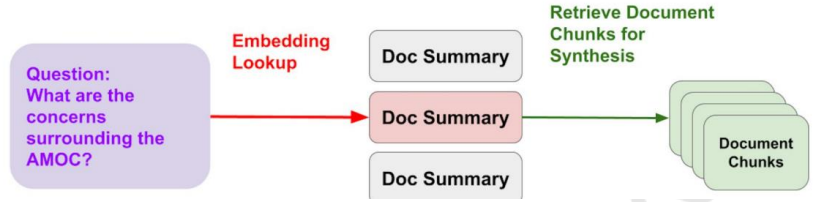
元数据过滤器

对文档进行分离和标记。查询期间，除了语义查询之外，并推断元数据过滤器

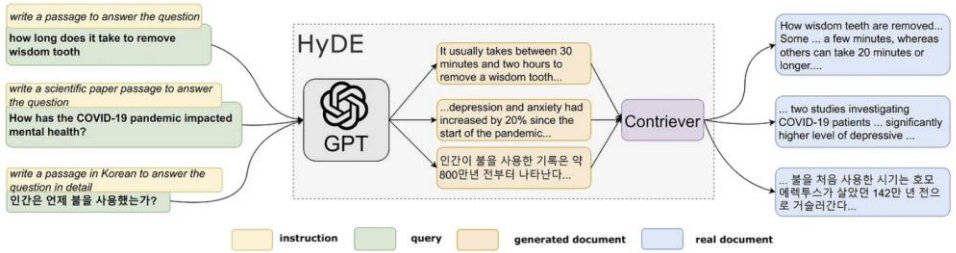
Small-2-Big



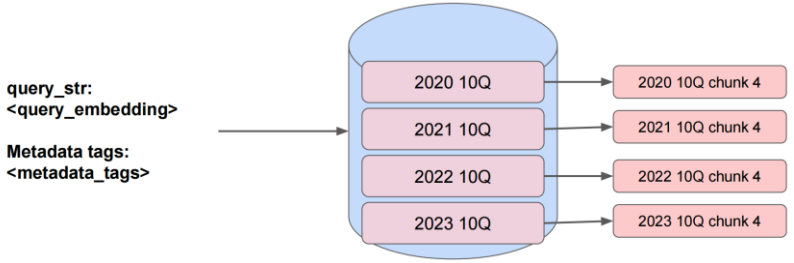
摘要



伪元数据



元数据过滤



Techniques for Better RAG ——结构化语料

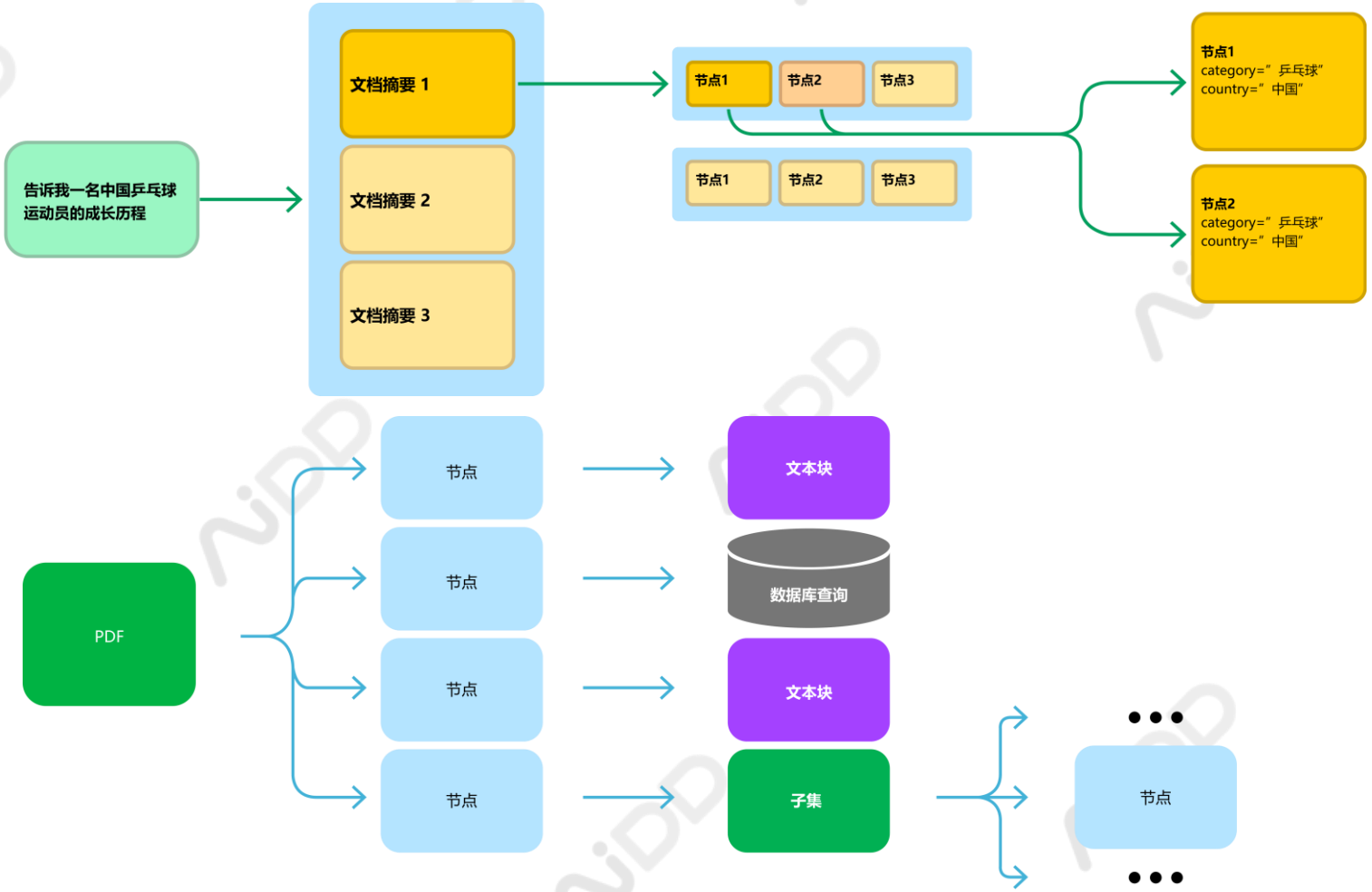
分层组织检索语料库

- 摘要 → 文档

用摘要检索代替文档检索，不仅检索直接最相关的节点，还会探索节点相关联的额外节点

- 文档 → 嵌入对象

文档中嵌入了对象（如表、图），先检索实体引用对象，再查询底层对象，例如文档块、数据库、子节点

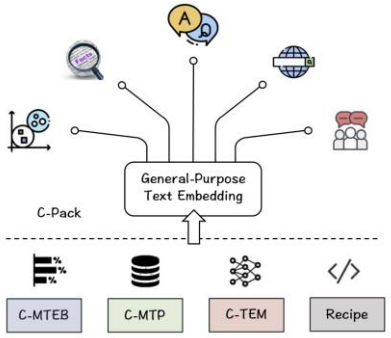


Techniques for Better RAG ——Embedding优化

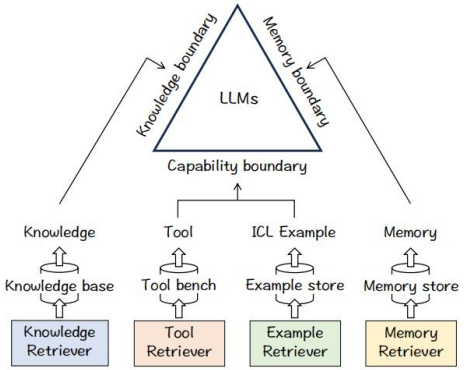
选择更合适的Embedding供应商

cohere　M3E　VOYAGE AI

OpenAI　llmrails　BAAI 智源研究院

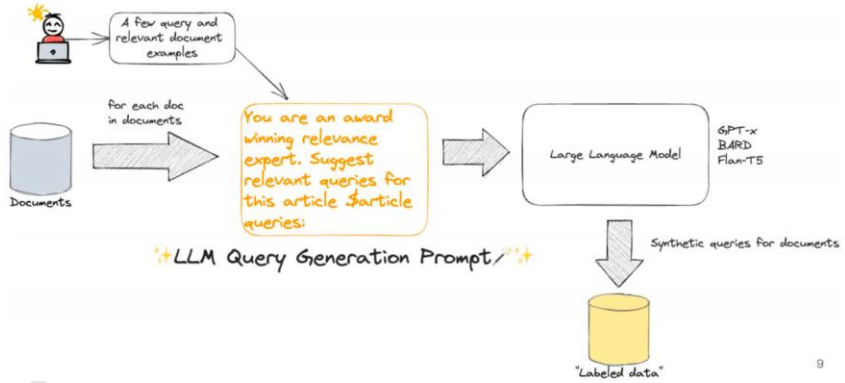


BAAI-General-Embedding (BGE)

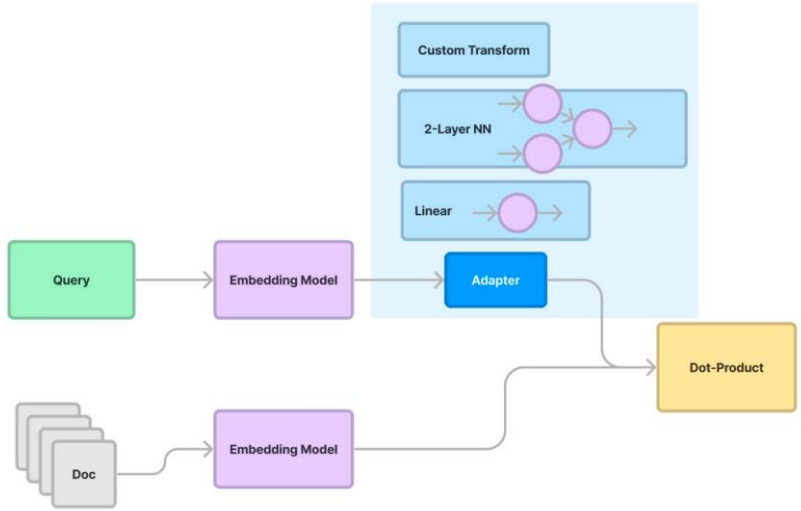


LLM-Embedder (BGE2)

微调Embedding模型



根据领域检索库和下游任务微调



微调Adapter模块，对齐Embedding模型和检索库

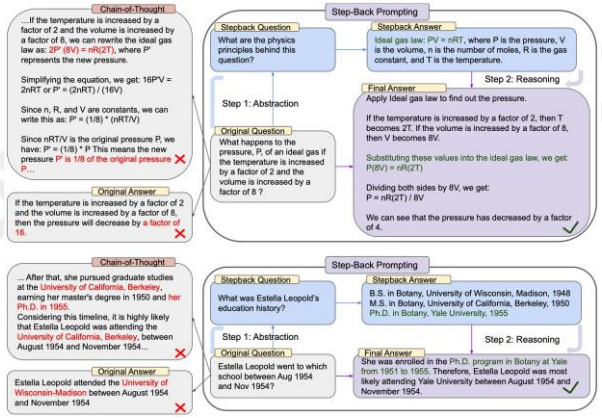
Techniques for Better RAG ——流程优化

Iterative

迭代的检索语料库，不断获取更细更深入的知识

ITER

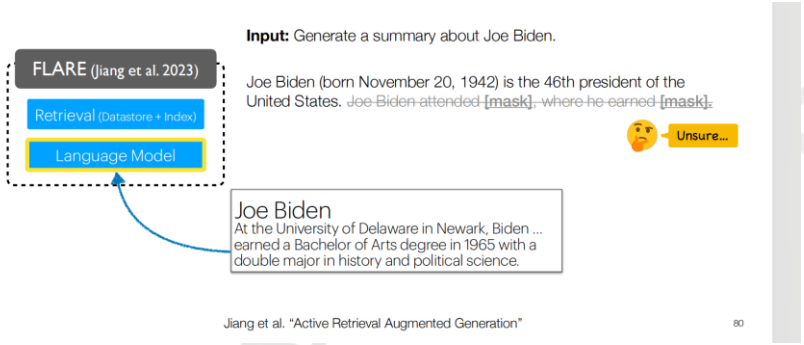
Step-Back Prompting



Adaptive

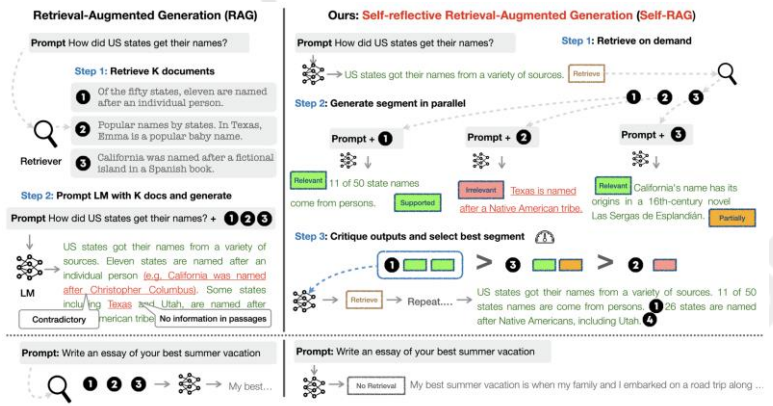
由LLM动态的判断检索的时机和范围

FLARE



Jiang et al. "Active Retrieval Augmented Generation" 80

Self-RAG



如何评估RAG的效果

评估方式

独立评估

检索评估
(Retriever Evaluator)

评估查询检索到的文本块的质量。
输出：与查询相关的“真实”文档
(MRR, Precision, NDCG)

生成评估
(Generation Evaluator)

分块外部知识库，使用LLM从每个（或一组）文本块生成问题。形成（问题，文本块）评估对

端到端评估

- **无标签评估指标（相关性，无害性）**
- **有标签评估（准确性，EM）**
- **人工 / GPT评估**

评价体系

RAGAS

生成 LLM对问题的回答如何	**检索** 检索内容与问题的相关程度
正确率 生成答案的事实准确性如何	**检索准确率** 检索语境的信噪比
答案相关性 生成的答案与问题的相关性如何	**检索召回率** 能否检索到问题所需的所有相关信息

RGB

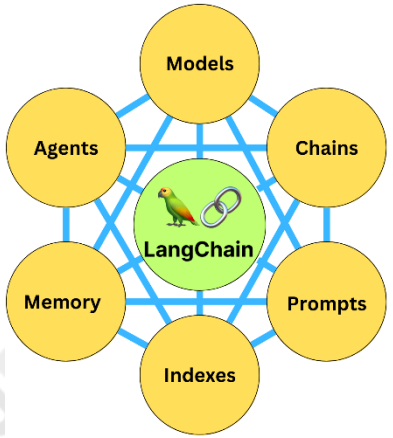
- 信息整合
- 噪声鲁棒性
- 反事实鲁棒性
- 否定拒绝

PART 04

知识检索增强技术栈与行业实践浅析

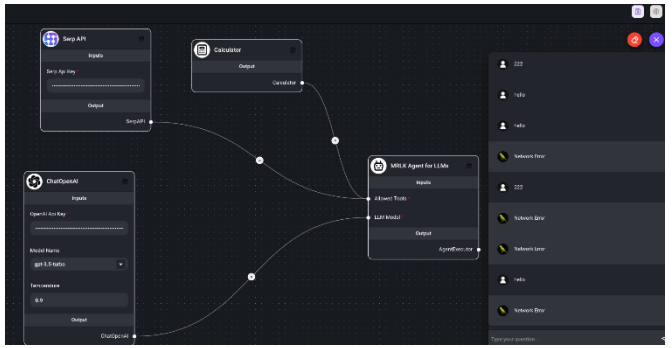
RAG 现有技术栈选择

名称	优点	缺点
LangChain	模块化，功能全面	行为不一致并且隐藏细节 API复杂，灵活度低
LlamaIndex	专注知识检索	需组合使用，定制化程度低
FlowiseAI	上手简单，流程可视化	功能单一，不支持复杂场景
AutoGen	适配多智能体的场景	效率低，需要多轮对话

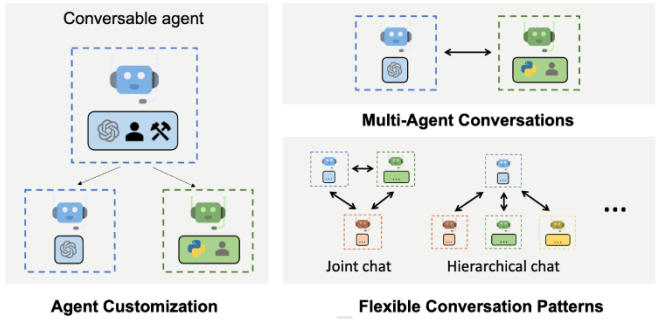


LangChain

LlamaIndex



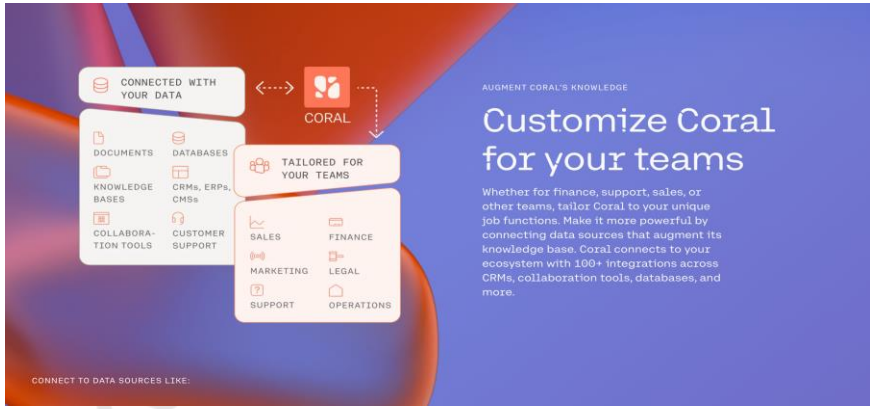
FlowiseAI



AutoGen

RAG 行业应用实践

网易有数 - ChatBI

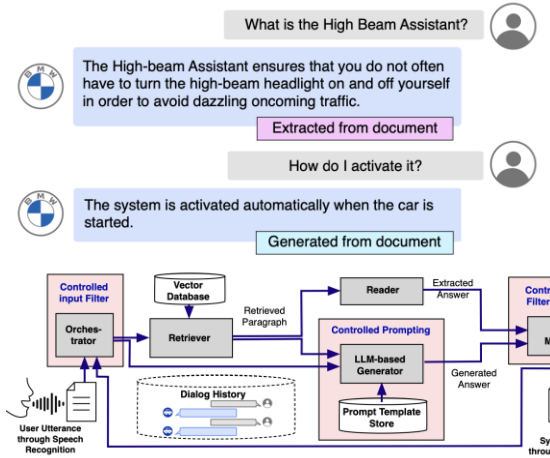


Cohere - Coral

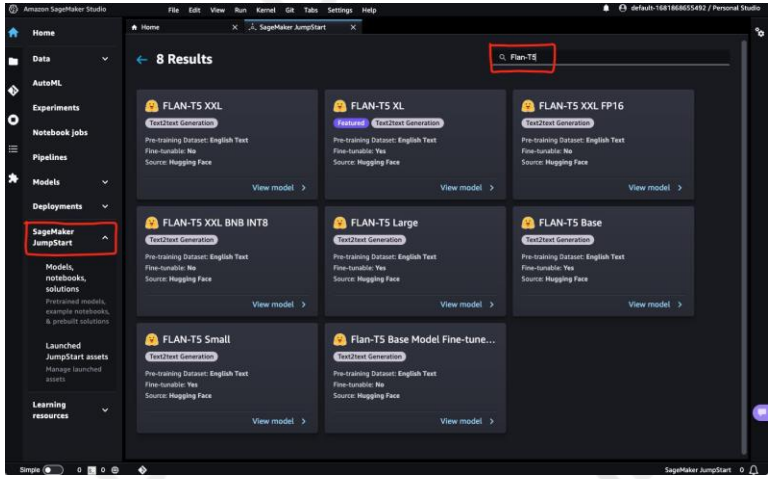
传统行业的智能化升级

RAG

AI工具链提升



BMW - CarExpert



Amazon - Kendra

PART 05
总结和展望

总结

RAG 技术栈

Langchain | LlamaIndex | FlowiseAI | AutoGen

RAG范式演变

朴素RAG | RAG进阶 | 模块化RAG

RAG的优化技巧

检索内容优化 | 结构化语料 | Embedding优化 | 流程优化

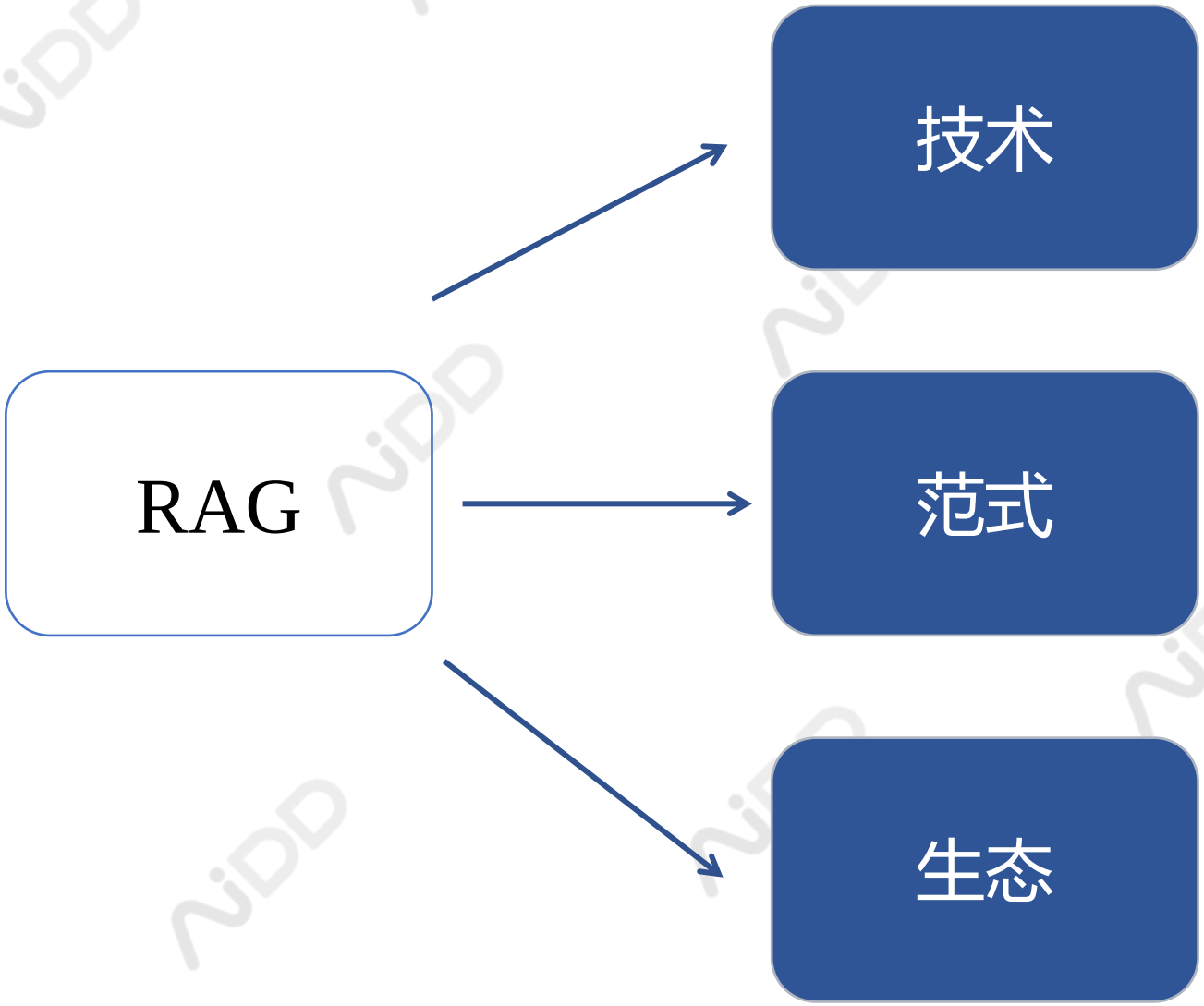
RAG的关键问题

检索什么 | 什么时候检索 | 怎么用检索的内容

RAG的评测

- 忠实性
- 答案相关性
- 上下文召回率
- 上下文精确性
- 噪声鲁棒性
- 否定拒绝
- 信息整合
- 反事实鲁棒性

展望



- RAG模型的Scaling Law规律
- 如何提升检索大规模数据的效率
- 长上下文场景下的遗忘缓解
- 多模态的检索增强

- 模块化将成为主流
- 模块组织待凝练模式
- 评测体系需与时俱进完善

- 工具链技术栈初步形成
- 一站式平台仍需打磨
- 企业级应用井喷

AI+软件研发数字峰会
AI+ software Development Digital summit

THANKS

OpenKG公众号

知识图谱与大模型技术算法、实战文章、行业案例分享