

RAG for LLMs: A Survey.

Abstract.

SLMs 缺陷：变幻莫测、知识时效性、不透明、推导过程不可追溯。
RAG：将外部数据库中的实时数据 → LLM，提升准确性和可信度。
对于知识密集型任务，可以使知识实时更新，并且接入领域知识。

- 本文：1. RAG 的进化：朴素 naive RAG → 中进阶 advanced RAG → Modular.
2. RAG 系统三个基础模块的方法：
① 检索、选
② 生成器
③ 增强方法。
④ 未来方向：① 未来挑战 ② 多模态 ③ 技术栈与生态。

1. Introduction.

LLM short comings. [Kandpal et al., 2023] 虚构事实
= 专业领域知识、高度专业化化的查询。
= Large language models struggle to learn long-tail knowledge.

↓ 在实际生产环境利用生成式 AI，黑盒 LLM 不可直接用。

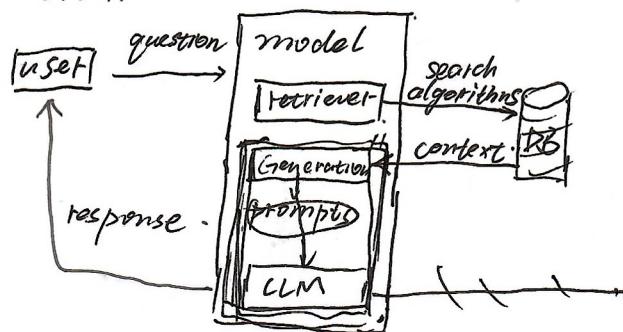
RAG 动态检索，外部知识，使用外部知识，作为参考生成回答。
what: tokens → phrases → entities → chunks.
when: single retrieval, adaptive retrieval, multiple retrievals.
how: input layer, middle layer, output layer.

① RAG 框架

② 技术

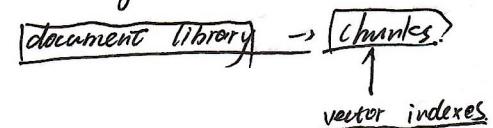
③ 评估。 RAG v.s. FT，未来展望。

2. Definition.

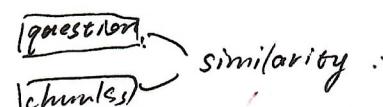


3步！新词检索！

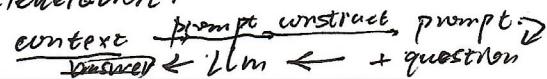
1. Indexing:



2. Retrieval:



3. Generation:



3. RAG Framework.

3.1. Naive RAG 朴素 RAG.

传统的3个过程：indexing. retrieval. generation.
("Retrieve" - "Read" framework)

- Indexing.

1. Data indexing: 数据清洗、抽取。处理不同形式数据到文本？
2. chunking: loaded text → smaller chunks.
LLM输入窗口有长度限制。

3. Embedding and Creating Index:

使用语言模型将文本编码为向量（用于检索以及计算相似度）

embedding model: 要求 high inference speed. 能够在用户提问时实时对该数据库和问题进行编码。参数量不可过大。

- ① embedding 为 vector → 原语料 - embedding
- ② create index. 构建 k-v 对，句子之后的检索。

- Retrieve

- ① 使用相同的 encoding model. 将问题是 query → vector
- ② 系统计算 query 与之相关的语料的相似度
- ③ 前 k 个文档 block. (相似度最高？)
- ④ 作为当前 query 的上下文 context.

- Generation.

question
documents → fresh prompt.

根据不同的需求，LLM 可以利用其内部知识，或只使用给定信息。
还可利用对话历史，使用 prompt 结合对话历史，形成多轮对话。

Drawbacks in Naive RAG 朴素 RAG 缺点

- ① 检索质量. {准确性低，检索的结果无法与 query 完整地对应。

low recall 低召回率。不是所有的相关结果都被检索到。

过时信息、冗余信息。

- ② 生成质量. { 幻觉. hallucination

不相关 irrelevance

有害 toxicity bias 偏见。

- ③ 对话过程 { 有效集成检索的阶段及生成阶段 → 缺乏相关性。

augmentation process | 冗余和重复。→ 重复内容。

②

④ 多篇文章检索时，重要性与相关性，需要有效权衡文章的价值，各文章的不同风格，以保证输出的连贯性

⑤ 生成模型过于依赖记忆信息， \rightarrow 不重新生成记忆，^{提供新的值或综合信息}

3.2 Advanced RAG 增强 RAG

针对检索、生成 ^{预处理}: pre-retrieval, post-retrieval.

针对 indexing 索引：sliding windows, fine-grained segmentation.
~~more data~~ metadata

优化检索过程：pipeline, end-to-end.

Pre-Retrieval Process 检索前

· Optimizing Data Indexing. 优化数据索引。

目标：增强建立索引的内容的质量。

① 增强数据粒度 Enhancing Data Granularity. 一致性。

pre-indexing 优化 — 提升文章的标准性，~~连贯性~~，保证事实准确性。丰富上下文，来确保 RAG 系统的性能。

文章标准性：移除无关信息，特殊字符，增加检索效率。
一致性：实体、术语消歧，去冗余

保证事实准确性：保证数据各部分的真实性。

维持上下文：保证系统的交互上下文了真实场景匹配。

— 通过特定领域的标注，添加一个新的话题下文的层，通过用户反馈来继续更新。

— 通过机制更新过时数据。

优化数据：明确，有上下文，正确，以保证系统有效可靠。

② 优化索引结构 Optimizing Index Structures.

(1) 调整 chunk 大小，映射到路径，~~跨~~来自图结构的信息。

↳ (1): 从小到大调整 chunk 块大小 \rightarrow 在最小化碎片同时同时捕捉相关信息
chunk 的大小在 RAG 系统中是一个重要的参数。

↳ (2): 通过多索引路径进行查询。与元数据后筛选和嵌入式相关联。
~~同时~~通过不同索引进行查询

· 构造索引用于特定的查询

· 独立的索引用于搜索及筛选，基于元数据关键词

↳ (3): 使用图结构，增加检索的相关性。

③ 增加元数据信息 \rightarrow 一体化利用元数据。

通过元数据进行初筛可以大幅提升检索的效率和相关性。

④ 对齐优化，对齐问题与文档之间的差异。"hypothetical questions"
假设性问题引入文档中。

⑤ 混合检索，利用不同的检索技术，如：基于关键词的搜索、语义搜索、向量搜索。
混合检索，适应不同的查询类型与信息需求，提升整体检索性能。

Embedding

Fine-tuning Embedding. 微调嵌入

目的：增强检索内容与查询的相关性

| 调整模型参数，

| 优化检索步骤。

对于一些专业领域，定制的 embedding 方法，可以增强检索的相关性。

[BAAI, 2023] Flagembedding. BGE 模型。

使用 GPT-3.5 编写基于文档的格式化问题及回答，用于微调

Dynamic Embedding.

| 静态：每个单词一个向量。

| 动态：基于单词的上下文调整。如 BERT.

OpenAI. embedding-ada-02. 比静态的更精确。| embedding 需包含尽可能多的上下文

Post-Retrieval Process. 检索后。

将检索到的上下文放到输入中给 LLM。

1. 受 LLM 上下文窗口限制。2. 连接大量相关文档作为一个长 prompt 效率低，引入噪音，影响 LLM 对句子关键信息。

• ReRank：重新定位最相关信息。① 放到 prompt 前 / 后。

LlamaIndex. Langchain. Haystack.

计算相关文本与查询之间的语义相似度。

• Prompt compression：压缩 prompt。检索文档中的噪音对 RAG 效果影响很大。
① 压缩无关上下文。② 强调关键段落。③ 减少整体上下文长度。

| [Litmat et al., 2020] Selective Context) 使用小模型计算 prompt 的共有信息

| [Anderson et al., 2022] LLM Lingua. | 处理复杂度，估计元素重要性。
但是，在 RAG 的长上下文场景下可能会丢失关键信息。

[Xu et al., 2023a] 在不同粒度下训练压缩器。解决冗余上下文，分离和压缩。

[Xu et al., 2023b] Long Context. 解决扩展现上下文，分离和压缩 “Walking in the Memory Maze”

[Chen et al., 2023a] 分级摘要树。→ 强化 LLM 的关键信息感知能力。

“Walking in the Memory Maze”

RAG Pipeline Optimization. RAG 流程优化。

提升 RAG 系统效率和质量。
① 融合集成不同的搜索技术 ② 优化检索步骤
③ 引入认知回溯。④ 灵活采用不同查询策略。⑤ 利用嵌入相似度。

- Exploring Hybrid search. 混合搜索.
 - 基于关键词. 语义检索. 向量检索.
 - 满足 ~~基于~~ 不同查询和所需信息. 保证对相关信息的连续检索.
- Recursive Retrieval and Query Engine. 递归的检索和查询引擎.
 - 利用之前已经查询到的上下文. 更少的检索查询代价也满足上下文需求.
- StepBack-prompt 让 LLM 不只专用于某个实例, 而是更广泛的概念或原则.
 - Sub-queries. 基于不同的场景使用不同的查询策略 [Zheng, 2023].
 - LlamaIndex 的查询框架. 平衡树查询. 利用向量查询. 使用更多简单的序列表.
 - Hyde. 在嵌入空间中, 成对的答案可能比直接查询更接近.
 - 使用 LLM, Hyde 创建一个假设的文档(答案)来响应查询, 嵌入此文档, 并使用生成的嵌入来检索与假设文档类似的真文档.
 - 不是根据查询来计算嵌入相似性, 而是比较一个答案到另一个答案的 embedding.
 - 模型不熟悉主题时, 会产生更加错误的结果]. X

3.3 Modular RAG 模块化 RAG

由只包含 indexing, retrieval, generation 的传统 RAG 扩展.

更加灵活, 集成多种方式来强化功能模块.(如搜索模块用于相似性检索、微调方法同向检索器)[Lin et al., 2023] 整合 RAG. (Restructured RAG, [Xu, 2022]) 迭代方法.[Shao, 2023.]

New Modules. 新模块.

· Search module. 搜索模块

{朴素强化 RAG: query 和 corpora 语料库 之间以相似性检索.

搜索模块: 特定场景. 利用 LLM 生成的代码, 在过程中集成;

· 对语料库进行直接搜索, 或利用其他工具.

数据源: 搜索引擎、表格数据、文本、KG. [Wang, 2023d]

· Memory module 记忆模块.

利用大模型本身的记忆能力来指导检索.

Selfmen[Cheng, 2023b] 迭代地根据检索增强的生成器来制作一个无限的记忆池. 检索增强的生成模型 (retrieval-enhanced generative model) 可以利用自己的输出来增强自身.

模型不使用训练数据, 而是自己的输出 [Wang, 2023a]

· Extra Generation Module. 额外的生成模型.

冗余和噪声. 阻碍 内容检索.

使用 LLM 生成必要的上下文, 可能比直接检索更能够包含相关信息.

任务适应模块

Task Adaptable Module.

使 RAG 适应各种下游任务。

Alignment Module 对齐模块

query 查询子 text 之间之词对齐

在 retriever 中加一个 trainable Adapter module. 可以有效缓解 对齐问题
强化学习.

Validation Module 验证模块

不是所有检索到的和都有问题。[Yu, 2023a]

New Pattern 新范式

RAG 中，针对特定问题，重组或代替 RAG 过程中的方法。

两个范式：{增加或替换模块

调整模块间的数据形式 (flow)

Adding or Replacing Modules 增替模块

引入或替换模块，保持 Retrieval-Read 过程，集成额外的模块来提升某些能力。

RRR [Ma, 2023a] Rewrite-Retrieve-Read

使用 ~~大模型~~ Language learning model 语言学习模型作为 rewriting 模块的强化学习的激励。利用 rewriter 对于 retrieval queries 检索查询微调，增强 reader 的下游任务的表现。

Generate-Read [Yu, 2022] 模块在方法中选择性地执行

Recite-Read [Sun, 2022] 改变外部检索为检索模型的权重，大模型初始化语义相关信息。

Adjusting the Flow between Modules 调整模块间的数据流。

调整语言模型与检索模块的交互。

DSP [Khattab, 2022] Demonstrate-Search-Predict

将上下文学习系统作为一个易于理解的程序，而非一个最终的信号 in prompt.

ITER-RETGEN [Shao, 2023]

使用生成和上下文指导检索，“迭代地实现” retrieval-enhanced generation “从“ generation-enhanced retrieval ””

通过 Retrieved-Read-Retrieve-Read 循环。

Self-RAG [Asai, 2023]

使用 Decide-Retrieve-Reflect-Read 过程，增加决策模块

4 Retriever

- 主要问题：1) 如何实现准确的语义表示？
2) 利用什么方式对齐查询 queries 和文档 documents 的语义空间？
3) 检索器的输出如何与大模型的表现对齐？

4.1 Better Semantic Representation. 更好的语义表示。

2部分. 语义体优化 chunk optimization / 微调嵌入模型 Fine-tuning Embedding Models

Chunk optimization.

对于外部文档，第一步是将其分为更小的块来抽取更细粒度的信息，这些块会被嵌入以表达其语义。

- 调整语料库中文档的分块大小。 → 提升检索效率和准确性。
 - 索引的原始内容 nature content.
 - 嵌入模型及其理想块大小。
 - 阅读查询的期望长度及复杂度。
 - 检索结果的特定应用。

在实际应用中，适应地应用不同的分块策略，不存在最好的全局策略，而要基于给定上下文选择最合适。

sliding window. merge 全局信息 用于多个检索过程。

- small 2 big 初始搜索阶段 使用小文本块 (text block). 然后使用更大的相关文本块
- abstract embedding 在文档摘要中使用 top K 检索
- metadata filtering 利用文档的元数据进行过滤过程。
- graph indexing 利用图结构，在多跳问题中可明显提升相关性。

以上方法相结合。

Fine-tuning Embedding Models.

决定分块大小之后，接下来的关键问题就是将这些块和查询嵌入到语义空间中。

embedding model UAT, Voyage, BGZ. 使用额外的语料库预训练。但对某一领域而言效果不太好。无法准确获取领域知识！
利用 embedding model.
↓
task-specific fine-tuning of Embedding model

模型理解用户的查询

有两个初始的方案 (embedding fine-tuning methods)
{ Domain knowledge Fine-tuning
Fine-tuning of downstream tasks }

- Domain knowledge Fine-tuning.

创造一个用于微调的领域数据集

微调 embedding model 与微调标准中语言模型的过程不同，主要基于使用该数据集上。

数据集普遍包含 3 个主要成分：查询 queries、语料 corpus、相关文档 relevant docs。

embedding model 基于查询 query 从语料库 corpus 中搜索相关文档 relevant docs，检索的文档必须是独立的部分，作为评估模型性能的标准。

布剑（造数据集、微调）以及评估的过程中有很多挑战。LlamaIndex 定义了一系列类和方法来简化这个过程。

- Fine-tuning of downstream tasks.

使 embedding model 适应下游任务。

利用大模型的能力来微调 embedding model。PROMPTAGATOR [Dai, 2022] 用大模型作为 few-shot 生成类 来生成 query？ \rightarrow 监督微调

LLM-Embedder [Zheng, 2023a] 从不同下游任务中，用 LLM 生成奖励值。

通过整合领域知识，以及针对下游任务的微调增强语义表示。但通过这些方式训练的检索器不能完全匹配某个特定的大模型。因此一些方法通过 LLM 的反馈直接同于嵌入模型的监督微调。

4.2 Align Query and Documents.

2 个核心技术

Query Rewrite · 重写查询

{· Query2Doc [Wang, 2023c] 利用大模型生成 pseudo-document 伪文档，用于指导语言模型（原始查询形式新查询）。

· HyDE [Gao, 2022] 使用 textual cue 构建查询向量，用于生成“假设 hypothetical”之高。

· RRR [Ma, 2023a] 重写查询。使用 LLM 生成查询，使用网络蜘蛛搜索引擎检索上下文。使用语言模型作为一个训练的 retriever 补足 LLM。

· STEPBACK PROMPTING [Zheng, 2023] LLM 通过抽象的推理，抽取高层次的概念和原则。基于这些抽象执行检索，最后执行名^o查询、检索方法。使用 LLM 生成多搜索查询。

Embedding Transformation

· LlamaIndex [Lin, 2023] 在查询编码器 query encoder 后加一个适配器 adapter。通过微调优化 query embedding 表示。将 embedding 映射到 latent space 来更好地对齐特定任务。将查询与结构化的外部文档对齐很关键。

· SANTA [Li, 2023a] 引入两个预训练的方法，增强 retriever 对于结构化信息的认识：

1. 利用结构化与非结构化数据的天然对齐关系，用于 structured-aware pretraining 的对比学习；
2. Masked entity prediction。实体导向的掩码策略，让语言模型填充掩码实体。

4.3. Align Retriever and LLM

检索的文档有可能与 LLM 的潜在需求不对齐。

2 个方法，对齐 retriever 的输出以及 LLM 的偏好偏好 preference。

LLM supervised training

利用LLM的反馈信号优化嵌入模型。

- AAR [Yu, 2023b] 使用一个encoder-decoder架构的LM，利用信号预训练检索器。
- REPLUG [Shi, 2023] 使用检索器和一个LLM来计算检索到文档的概率分布。然后通过计算KL散度来监督微调。
- UPRISE [cheng, 2023a] 使用冻结的LLM微调prompt retriever。LLM和retriever都使用Prompt-Input对作为输入。利用LLM的输出来监督 retriever 的训练。
- Atlas [Izacard, 2022] 4个方法监督微调 embedding model。
 1. 游戏：LLM产生的 cross-attention 分数蒸馏模型的输出。
 2. EMDR2：使用 Expectation-Maximization algorithm 将检索到文档作为潜在变量 latent variables 对模型进行训练。
 3. Perplexity distillation 蒸馏：利用生成的 token 的复杂度作为指示直接蒸馏。
 4. CROP：基于删除文档对LLM的影响设计损失函数。

提高LLM和retriever的协同作用。

Plug in an adapter

- PRCA [Yang, 2023b] 通过上下文抽取阶段和奖励驱动来训练 adapter。使用 token-based 方向策略优化 retriever 的输出。
- token filtering [Bachanay, 2023] cross-attention 分数来过滤 token，只选择高分的 token。
- RECOMP [Xu, 2023a] 抽取和生成的 compressor 用于摘要生成。选择相关句子或综合文档信息，创建适合多文档查询的摘要。
- PKG [Luo, 2023] 通过一个指示器，将知识结合到蒸馏模型中。retriever 被一个直接基于 query 生成出相关文档的模块。

5. Generator

RAG生成器可以利用检索到的数据提升准确性和相关性。

5.1 Post-retrieval with Frozen LLM.

GPT-4等大模型性能最好，但受上下文长度限制以及易受冗余信息影响。

Post-retrieval 方法目标是提升检索结果的质量，将检索结果与用户需求或后续任务对齐。在检索阶段重新处理获取的文档。

information compression
result reranking,

Information compression 信息压缩

the reduction of noise 减少噪声。
addressing context length restrictions. 解决上下文长度限制。
augmenting generation effects. 提升生成效果。

- 压缩
- PRCA [Yang, 2023b] 训练信息抽取器，从输入文档 Simce 中压缩上下文，输出序列 *Compressed*. 该过程是渐进的，*Extracted* 和 *Compressed* 之间的差异。
 - R2COMP [Xu, 2023a] 通过对比学习训练信息压缩器 *information condenser*，减少冗余。“Filter-Reranker” [Ma, 2023b] SLM small language model 作为过滤器。LLM 作为 reordering agents.

Re-ranking. 重排序。

将重新组合检索结果，将最相关的放到前面。

解决上下文窗口不足问题，提升检索效率和响应能力 *responsiveness*.

[Zhuang, 2023].

5.2 Fine-tuning LLM for RAG.

generator 更自然有效地利用检索到的文档，更好地适应用户的查询的需要。

普通的 LLM 生成任务输入：查询

RAG 的输入：查询、检索到的各种（结构化/非结构化）的文档。

RAG 输入的文档会很大程度地影响模型的误解，特别是些小模型。X

微调 LLM 应该包含上下文文档和查询的输入。

General Optimization Process. 一般优化过程。

Self-Mem [Cheng, 2023b] 给定输入 query 和检索的文档，模型生成 y.

两个 fine-tuning 的方式：Joint-Encoder & Dual-Encoder.

- Joint-Encoder：使用标准的 encoder-decoder 模型。encoder 对输入编码，decoder 通过注意力机制，以自回归的形式对 encoder 的结果生成 tokens.
- Dual-Encoder：设计了两个独立的 encoder。一个对 (query, context) 编码，一个对文档编码 (document). 通过 decoder 的双向交叉注意力过程生成输出 bidirectional cross-attention. 利用 transformer 作为基础结构。

Utilizing Contrastive Learning. 利用对比学习。

在准备训练数据的时候，生成正输入-输出对作为训练数据，此时模型受限，因为只有唯一真值输出。“exposure bias” X 模型仅暴露在唯一真值，反馈 true feedback，无法 access 其他 token. 这影响模型在 real-world 应用时的表现。“as it may overly adapt to specific feedback in the training data without effectively generalizing to diverse scenarios”.

SURGE [Kong, 2023] graph-text 对比学习。利用对比学习，模型训练，生成多种合理的响应（在训练集之外）。

SANTA [Li, 2023d] 3阶段的训练过程，理解性地捕捉结构性信息和语义信息，利用对比学习优化 queries 和 documents 的 embedding representation.

- mask 実际，模型利用上下文（从上下文抽取必要信息）恢复 masked 的实际。
使模型理解结构数据的结构性质语义，对于 structured data 中的相关实际。⑩

6. Augmentation in RAG.

3个核心方面：增强阶段、增强数据来源、增强过程。

6.1 RAG in Augmentation Stages.

Pre-training Stage. 预训练阶段

使用检索的方法增强预训练语言模型在开放域上的问答。

- REALM [Arona, 2023] 提出了结构化的可解释的指示嵌入方法，将预训练和微调结合作 retrieve-then-predict 的过程。基于一个 masked LM 的范式。
- RETRO [Borgeaud, 2022] 利用检索增强做大规模预训练。RETRO encoder 对外部数据库和邻居序列的特征进行编码。

检索阶段

- Atlas [Izacard, 2022] 利用相似度的方法，将 T5 模型结合了检索机制。
- COG [Lan, 2022]
- RETRO++ [Wang, 2023b] 在 RETRO 的基础上提升了参数量。

挑战：预训练数据和所需资源随模型增大而增加，提升生成质量、准确率，但有更新速度慢（困难）
完整性、下游任务...

Fine-tuning Stage. 微调阶段。

REPIUG [Shi, 2023] 将 LM 作为黑盒，通过从 LM 反馈的监督信号增强可调整的检索模型。

- UPRISE [Zhang, 2023] 通过训练不同的任务集，制作轻量级的任务驱动型 retriever。
- Self-Mem [Zhang, 2023] 利用内存池中的 example 挑剔 generator。
- RA-DIT [Lin, 2023] 最大化 retrieval-enhanced 命令的回答的正确率来微调 generator 和 retriever。

SUGIRE [Kang, 2023] 对比学习，训练端地微调 retriever 和 generator。
通过使用基于 GNN 的上下文感知的子图检索器，从 KG 中抽取相关的知识，来支撑一个对话进行的对话，确保生成的回复准确且反映检索到的知识。

总结：微调 LM 或者 retriever，都可以更适应特定任务。

(优) 通过适应不同的下游任务，使模型更多样化。

更好地适应不同数据格式，如 KG。

(劣势) 需要额外的数据集和计算资源。

Inference Stage. 推理阶段。

朴素 RAG 就是通过检索上下文在推理阶段结合。

- DSP [Khattab, 2023] 采用复杂的流水线来使 frozen LM 和 retriever 之间交替来提升生成质量。

- PKG : knowledge-guided module. 在不改变 LLM 参数的情况下，可以获取相关信息。
- CREA-ICL [Li, 2023b] 利用跨语言的知识同步检索，来获取更多信息。
- ITRG [Feng, 2023] 通过反复检索查询正确的推理路径来提升对多步推理任务的适应性。

· TIER-RETGN [Shao, 2023] 提出“retrieval-enhanced generation”和“generation-enhanced retrieval”。

· RGRA [Guo, 2023] 提出一个两阶段架构用于非知识密集型 non-knowledge-intensive. ~~PKI~~ NKI 任务：① 固化各不可知 task-agnostic 的检索并选择有效的候选证据。② 由 prompt-guided reranker 优先考虑相关的证据。

· IRCoT [Trivedi, 2022] 基于 RAG 和 CoT，利用交替的 CoT 指导的检索，来和检索结果提升 CoT.

总结：轻量、cost-effective 方式，不依赖外训练。
 (CB) ✓
 (CoT) ✓

需要额外的数据处理和优化。
 受基础模型能力限制。

通常，这种方法通过逐步处理、迭代推理和自适应检索等过程优化技术相结合，有效地满足不同的任务要求。

6.2. Augmentation Data Source 增强数据源。

不同粒度和维度的数据。

分成 3 类：非结构化数据、结构化数据、LLM 生成的内容。

Augmented with Unstructured Data 非结构化数据。

在粒度层面：普通的 chunks (块) 一般为句子。除此之外还有 tokens, phrases 短语、document paragraphs. 文档段落。

粒度细：更高的样式以及领域外知识，但开销更大。

Augmented with Structured Data 结构化数据

KG. 提供高质量上下文，减少幻觉的可能性。

RET-LLMS [Modarressi, 2023] 通过在历史对话中抽取三元组，构建 personalized KG 内存。

SUGRE [Kang, 2023] 使用 GNN，从 KG 中检索子图，并防止模型生成无关内容。使用了一个图的编码方式，将图结构的信息嵌入到 PLM 的表示空间中，并通过多模型对比学习，在图子文字模式中，确保可以连续地检索和生成事实。

knowledgeGPT [Wang, 2023d] 从知识库中查询（生成查询代码）。

LLM-Generated Content in RAG 大模型生成的内容。

利用大模型生成的内容来检索。

· SKR [Wang, 2023e] 对数据集打标签，大模型对 unknown 的输入进行检索，其他直接回答。

· Gen-Read. [Yu, 2022] 用 LLM generator 代替 retriever.

· Selfmen [Cheng, 2023b] 使用检索增强的生成器，制作一个无限的内存池。使用 memory selector 为后续的生成选择 memory (之前已经输出)。

6.3 Augmentation Process. 增强过程。

只用一个 retrieval & generation 的过程。可能会有冗余，以及“lost-in-the-middle”影响。
冗余会模糊关键信息，或包含与实际不符的信息，影响生成效果 [Yoran, 2023].
在涉及多步推理的问题中受限。

Iterative Retrieval / 迭代检索。

在原生的 query 和生成的文本的基础上，再检索相关信息，为 LLM 提供更多的参考，可以提高鲁棒性，~~且可能~~ 以及支撑后续问题(子问题)的生成。
但可能会在语义上不连续，以及引入噪声和冗余信息。

· Recursive retrieval ~~递归检索~~。先在结构化的索引上检索，再一级级向下检索。

· Multi-hop retrieval 多跳检索。更深入挖掘信息或图结构数据。[Li, 2023c]

· IR retrieval ~~从 generation 到过程中迭代~~。[ITER-RETGEN. Shaw, 2023]

[IR-COT Trivedi, 2022]

Adaptive Retrieval ~~适应性检索~~。

Flare [Jiang, 2023b]. Self-RAG [Asai, 2023] 优化 RAG 检索过程，让 LLM 自适应地判断检索的时机和内容。

Graph-Toolformer [Zhang, 2023]. LLM 通过使用检索器 retriever, Self-Ask, DSP, 来从 Agent.

WebGPT [Jiang, 2023b] 使用搜索引擎。

Flare. [Jiang, 2023b] 基于生成物的质量，自动判断检索的时机。

Self-RAG [Asai, 2023]

7. RAG Evaluation. 评估。

一般在特定下游任务。如：

QA: exact match (EM)、F1
事实检测 fact check：准确率 accuracy.

7.1 Evaluation Targets. 评估目标。

检索质量. Hit Rate, MRR, NDCG.

生成质量. 可信度
相关性
有效性。

手动评估
自动评估。

7.2 Evaluation Aspects 评估方面

Quality Scores. 质量分数。

检索 \leftarrow 1. Content Relevance 上下文相关性。上下文之间相关性。

生成 \leftarrow 2. Answer Fairfulness 回答可信度。给定上下文，model 生成必须可信。

3. Answer Relevance 回答相关性。回答与 query 相关。

Required Abilities 需要的能力

- 1. Noise Robustness. 噪声鲁棒性. 模型应对噪声的能力.
- 2. Negative Reception 报道负面信息. 上下文中的回答问题时需要信息, 模型应飞鸽传书回答.
- 3. Information Intergration. 信息整合. 整合不同文档中的信息. 回答复杂问题.
- 4. Counterfactual Robustness 非事实鲁棒性. 模型识别文档中的错误信息. (模型可直接回答的问题, 但上下文中提供了错误信息.)

7.3 Evaluation Framework: Benchmark and Tools.

8. Discussion.

8.1. Optimization techniques of LLMs

8.2 RAG vs. Fine-Tuning.

9. Future & Prospects

- Long context 长上下文.
- Robustness 鲁棒性.
- Hybrid (RAG+FT)
- Role of LLMs. 检索、生成评估, ?.
- Scaling laws for RAG.
- Production-ready RAG.

Modality Extension of RAG 多模态

图像

音频视频

9.2 Ecosystem of RAG.

Downstream Task and Evaluation.

Technical Stack.

10 Conclusion.