基于 Agentic RAG 的 AI教学辅助设计系统

信工xxx xxx（xxxxxxxx）

摘要：本研究旨在开发一种基于Agentic RAG技术的教学辅助系统，通过优化AI驱动的教学工具提升教学效率和个性化学习体验。随着信息技术的迅猛发展，人工智能在教育领域的应用日益广泛。然而，大语言模型（LLMs）在专业领域中常面临“幻觉”问题，即生成内容可能与事实不符，这限制了其在教学中的应用。为解决这一挑战，本项目引入了检索增强生成（RAG），该机制通过从外部源动态获取信息以增强生成过程的准确性和时效性，并且集成AI代理（Agent），系统能够执行更复杂的查询处理和多步骤推理，从而提供更加精准和个性化的教学支持。使用Agentic RAG并结合现代化的前后端设计及容器化部署策略，本项目实现面向专业领域的教学大纲、创新性实验内容和教学案例的自动生成，提高教学内容生成的质量和效率，减轻教师的工作负担。

关键词：大语言模型，检索增强生成，教学辅助

# 1 研究背景

随着信息技术的飞速发展，人工智能技术逐渐渗透到教育领域的各个方面，成为推动教育信息化和现代化的重要力量。近年来，AI技术在教育中的应用从早期的辅助工具逐渐发展为智能化教学系统的核心组成部分，例如，AI驱动的个性化学习平台能够根据学生的学习行为和能力水平，动态调整学习内容和难度，从而提高学习效率。此外，AI技术还被广泛应用于教学资源管理、学习行为分析和教学效果评估等领域，为教育工作者提供了强有力的技术支持。

传统的教学辅助工具往往依赖于教师的经验和手动操作，难以满足现代教育对个性化和高效化的需求。教师在教学过程中面临诸多挑战，例如如何快速生成高质量的教学内容、如何设计符合学生需求的个性化教学方案，以及如何在有限的时间内完成复杂的教学任务。此外，随着教育信息化的推进，教师需要处理的教学数据量急剧增加，传统工具已无法有效应对这些挑战。因此，开发一种基于AI技术的教学辅助系统，成为解决这些问题的关键方向。

2024 年是大语言模型飞速发展的一年，国内外涌现出许多生成式AI教学辅助工具。然而新出现的工具大多依赖语言模型本身的生成能力，在专业特定领域仍然无法避免LLMs“幻觉”、“说瞎话”等问题，极大阻挠了大模型在教学领域方面的应用。

本课题旨在设计并实现一种基于Agentic RAG的教学辅助系统，通过RAG实现面向专业领域的教学大纲、创新性实验内容和教学案例自动生成，减少AI“幻觉”的出现；通过 Agent工作流优化模型输出，为教师提供全方位的教学支持。该系统的应用将显著提高教学效率，减轻教师的工作负担，同时为学生提供更加个性化和高效的学习体验。此外，本课题的研究成果还将为教育信息化和智能化提供理论支持和实践参考，推动教育领域的创新发展。

# 2 文献综述

2.1 **检索增强生成（RAG）**

2.1.1大模型在教学内容生成方面的不足

虽然人工智能在自然语言处理方面表现出了卓越的能力，但它们对静态预训练数据的依赖往往会导致各种各样的问题。

由于模型训练数据集权重的综合考虑，包括对于版权的担忧，很多领域专业知识可能并不会被包含到模型训练数据集里。这会导致大模型无法直接生成正确的领域专业性内容。由于大模型部署所耗费的GPU与存储资源非常惊人，一般会通过量化[1]、知识蒸馏[2]等方式减小模型的大小与部署成本。这也会导致模型边缘知识的缺失。

大语言模型本质上是一个知识的压缩器[3]，模型能够根据从训练数据中学习到的模式生成听起来合理的文本，即使生成的内容与现实不符，从而导致输出逻辑与事实错误的内容，误导使用者，这就是幻觉。幻觉可能是无意的，可能由各种因素造成，包括训练数据中的偏差、模型无法获取实时或最新信息，或者模型在理解和生成符合语境的准确响应方面的固有局限性[4]。

教师在教育过程中肩负着传播知识的重要责任，其核心任务是确保学生能够获取准确、可靠的信息。“幻觉”在日常对话或创意写作中或许可以被容忍，但在教学领域却带来了无法承受的代价。因为教育不仅关乎知识的传递，更关乎学生认知结构的构建和批判性思维的培养，一旦错误信息被当作事实接受，可能对学生的学习效果和知识体系产生长期的负面影响。研究表明，LLMs生成的不准确内容可能误导用户，尤其是在缺乏外部验证的情况下[5]。因此，在日常教育场景中，教师必须对LLMs的输出保持高度警惕，确保其辅助作用不会因“幻觉”而背离教学的根本目标。

RAG 通过从外部源动态检索相关信息并将其纳入生成过程来解决这一限制，从而实现上下文准确且最新的输出。

2.1.2 RAG的原理与应用

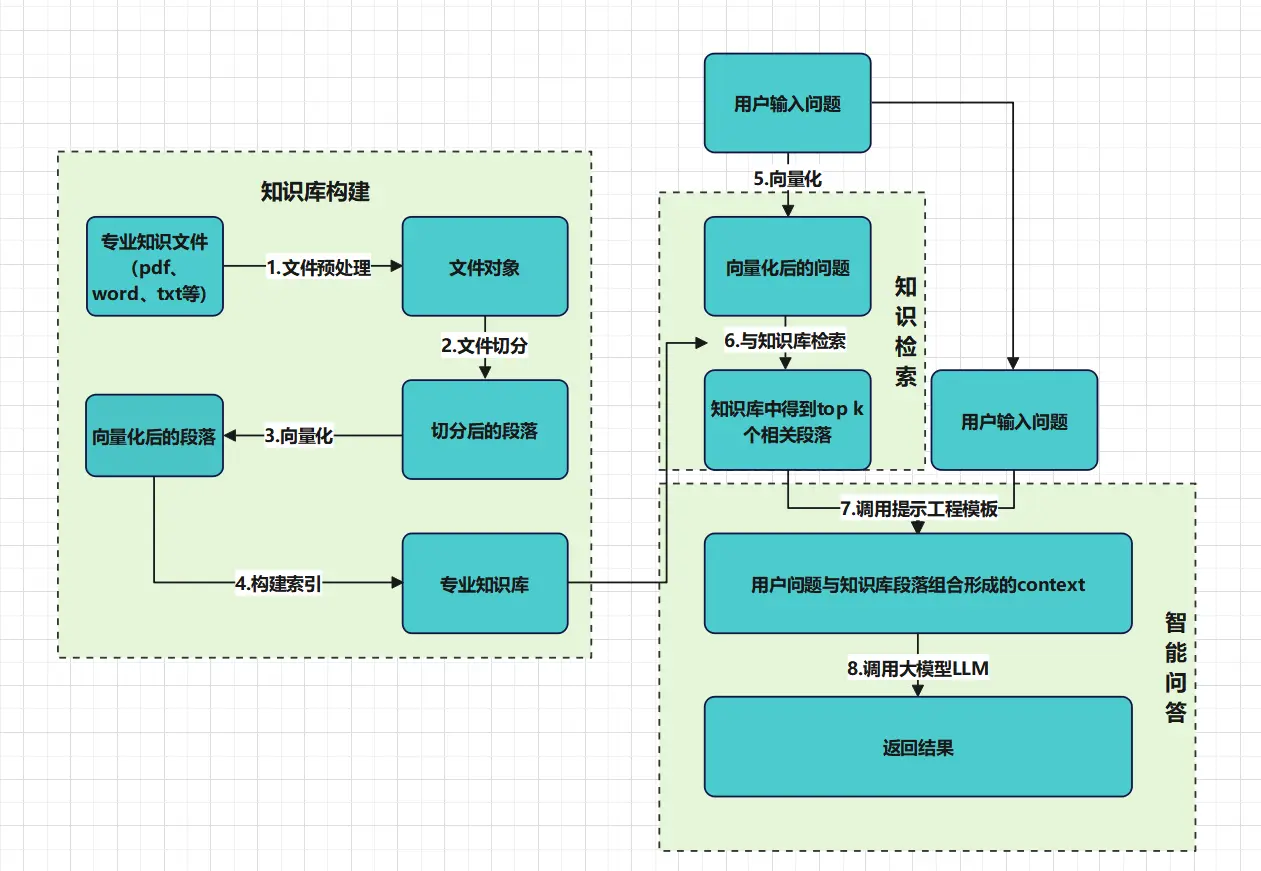
检索增强生成（Retrieval-Augmented Generation，RAG）是当前辅助LLMs生成更准确、更丰富文本的主流技术。RAG通过将外部检索到的相关信息作为有效线索整合到模型的提示中来增强LLMs的文本生成，有效减少了知识密集型任务中的事实错误[6]。通过此方式，RAG增强了模型处理知识密集型任务的能力，如问答、文本摘要、内容生成等。

RAG模型由Facebook Al Research团队于2020年首次提出，并迅速成为大模型应用中的热门方案。RAG 的含义是：

第一步：检索（Retrieval）。检索是RAG流程的第一步，从预先建立的知识库中检索与问题相关的信息。这一步的目的是为后续的生成过程提供有用的上下文信息和知识支撑。

第二步：增强（Augmented）。增强是将检索到的信息用作生成模型（即大语言模型）的上下文输入，以增强模型对特定问题的理解和回答能力。这一步的目的是将外部知识融入生成过程中，使生成的文本内容更加丰富、准确和符合用户需求。通过增强步骤，LLM能够充分利用外部知识库中的信息。

第三步：生成（Generation）。生成是RAG流程的最后一步。这一步的目的是结合LLM生成符合用户需求的回答，生成器会利用检索到的信息作为上下文输入，并结合大语言模型来生成文本内容。



**图1 RAG 技术架构总览**

应用RAG技术的LLM可有效解决原生LLM的幻觉问题、时效性问题与数据安全问题[7]。

2.1.3 RAG的改进

传统RAG（Native RAG）系统在面对复杂任务时仍存在局限性，如语义理解不足、输出相关性不高以及对动态环境的适应性有限。为此，研究者提出了多种改进方法，包括GraphRAG、Self-RAG、DF-RAG和Agentic RAG等。这些方法分别从知识结构、自我反思、动态细粒度和代理增强的角度优化了RAG的性能。

* 1. **GraphRAG**：GraphRAG是由Microsoft Research开发的一种方法，通过将知识图谱引入RAG框架，显著增强了语义理解和上下文关联性。知识图谱是一种结构化数据表示，捕捉实体及其关系，例如“城市-国家”或“作者-书籍”，这使得GraphRAG能够更好地处理需要多跳推理的复杂查询。GraphRAG的工作流程包括两个阶段：首先，LLM从文本数据中提取实体和关系，构建知识图谱；然后，利用Leiden社区检测算法将图谱分层为语义社区，并生成社区摘要。在查询时，系统通过map-reduce方法结合社区摘要生成全局答案。研究在播客和新闻数据集（约100万token）上测试了125个语料库特定的感官问题，结果显示GraphRAG在全面性和多样性上显著优于传统向量RAG[8]。例如，对于“数据集的主要主题是什么”这类全局问题，GraphRAG能够提供更结构化的回答。

1. **Self-RAG**：Self-RAG是一种通过自我反思提升RAG输出质量和事实性的框架，特别强调在生成后检查输出与问题的相关性。Self-RAG训练一个单一的任意LM，适应性地按需检索段落，并在生成和反思中使用特殊标记（reflection tokens）。这种方法允许LM在推理阶段可控地调整行为，以适应多样化的任务需求。研究显示，Self-RAG（7B和13B参数）在知识密集型任务上显著优于Native RAG系统[9]。
2. **DF-RAG**： Dynamic Fine-grained RAG是一种通过动态调整检索细粒度并利用熵值计算和相关性排序来提升输出有效性的方法。DF-RAG涉及生成多个查询并通过倒数排名融合（Reciprocal Rank Fusion, RRF）提升检索精度。该方法的工作流程包括：(i) LLM生成多个查询以扩展用户输入；(ii)矢量搜索为每个查询检索文档；(iii) RRF算法根据相关性重新排名文档；(iiii) LLM综合所有查询和文档生成响应。研究在四个开放域问答数据集上，与四种不同类型基准模型对比实验结果显示，DF-RAG 在多数情况下处理语义复杂查询的性能更优，且验证了反馈标签、重写模型的有效性[10]。然而，当衍生查询与原始查询相关性不足时，可能会导致回答偏离主题[11]。
3. **Agentic RAG**：Agentic RAG是一种将AI代理集成到RAG系统中的方法，通过赋予系统自主决策和多步骤推理能力，提升信息检索和生成的灵活性[12]。Agentic RAG超越了传统RAG的静态工作流，嵌入自主AI代理以处理多步骤推理和复杂任务管理。

本项目综合考虑了实现难度、应用效果、工程实践优越性后，选用了Agentic RAG作为RAG的扩展方向。

2.2 Agentic RAG的应用

Agentic RAG是对传统 RAG的进一步优化，其核心在于将 AI 代理（Agent） 整合到检索与生成流程中，使系统具备更强的自主决策和动态适应能力。与传统 RAG 相比，Agentic RAG 不仅依赖外部知识库增强 LLMs的生成能力，还引入了智能代理机制，使系统能够自主感知任务需求、规划检索策略并动态优化输出结果。

Agent是一种能够感知环境、自主决策并执行任务的智能软件实体，其运作方式类似于人类的问题解决过程——分析需求、制定策略并采取行动。这种能力使得 Agentic RAG 特别适用于动态、复杂的应用场景，例如个性化教育、实时数据分析等。在传统的 RAG 中，检索和生成过程通常是静态的，而 Agentic RAG 则允许系统根据上下文动态调整检索范围，甚至执行多轮查询优化，从而提供更精准、相关且个性化的输出。

在教学辅助系统中，Agentic RAG 的工作流程可以概括为：(i)代理接收用户查询并初始检索相关数据；(ii)代理评估这些数据是否足够，如果不够，它会制定子查询并进一步检索；(iii)当代理确定信息充足时，生成最终的回答。这种方法特别适合创建个性化的学习体验，例如作为导师评估学生的理解并调整教学策略，或生成定制的学习材料和测验。

此外，Agentic RAG进一步增强了系统的能力。Agentic RAG在时间序列分析任务上的预测、分类、异常检测和插补任务都具有良好的准确性和鲁棒性[13]；在教育领域，Agentic RAG 提供了改进的适应性、上下文感知生成和实时行动能力，可以辅助复杂任务如个性化辅导和持续学习支持[14]。例如，它通过连续学习从用户互动和系统日志中改进表现，确保决策基于准确信息，减少错误和过时建议。

2.3 现存的AI教学生成案例分析

Teaching Assistant描述了一个基于 LLM 和 RAG 的教学助理系统，能够处理上传的书籍并为学生提供即时支持[15]，体现了 RAG 的信息检索优势。该项目在Github 上开源，如果其结合 Agent 技术，这一系统可进一步扩展为执行任务如安排复习会或更新学生记录。

国内的LLM平台中，豆包提供了较为友好的教案设计功能。该功能允许教师上传材料，并通过RAG增强输出的相关性。豆包还提供了内置提示词用于教案生成。不过，豆包并未使用向量数据库进行索引，这也导致其对用户上传的文档有着最大字符限制（50000字符）。

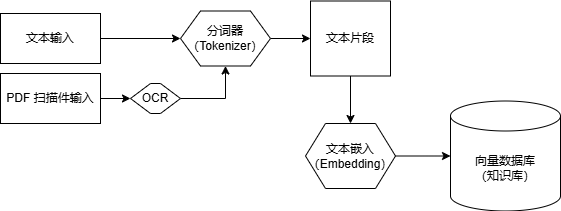
另一个例子是 Cogniti.ai，一个由教师设计的平台，允许教师创建自定义 AI 代理，结合特定资源和指令辅助学生学习，展示了 Agent 技术的个性化潜力。调查还指出，RAG AI 在教育中的应用通过定制信息交付革命化了学习体验[16]，而 Agent 的加入进一步提升了互动性和决策能力。

# 3 技术路线

根据 Agentic RAG 技术的原理，项目实现的核心步骤为**知识库构建**与**用户查询处理**。此外，项目还结合了优秀的**工程实践**，采用前后端分离式架构与容器化部署以降低上手与部署难度。

3.1 知识库构建

知识库构建是本项目的核心步骤之一，旨在将教师提供的教学材料或互联网上的可信数据转化为结构化的知识，存储于向量数据库中，以便后续查询和处理。知识库构建的流程如图 2 所示：



**图2 知识库构建流程**

由于本项目的主要应用场景是教学辅助，知识库的构建可能涉及从电子书、PPT、PDF等格式的教学材料中提取文本。文本预处理阶段主要包括以下几个步骤：

**（1）文本输入与OCR处理**：对于扫描件或图片格式的教学材料，应先使用OCR（光学字符识别）技术将图像中的文字转换为可编辑的文本格式。由于教学材料的特殊性，传统OCR使用的神经网络可能无法很好地完成公式、表格识别的工作。本系统采用RAGFlow团队的DeepDoc模型，该模型基于paddleOCR进行了微调，对公式、表格识别进行了特殊优化，使其在学术书籍、课件等相关资料的OCR中具有优秀的表现。

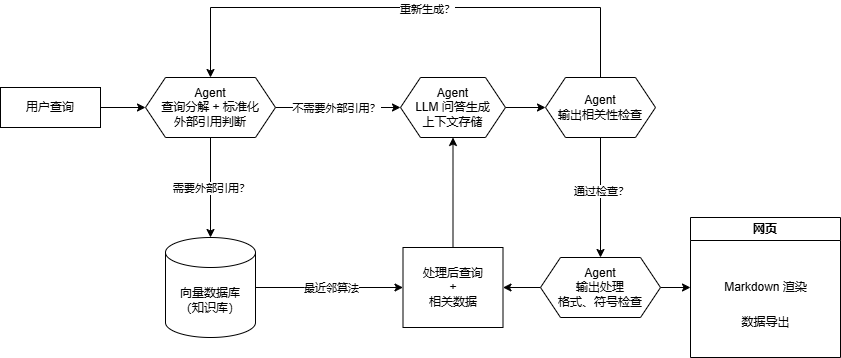
**（2）分词与文本片段生成**：使用分词器（Tokenizer）对文本进行分词处理，将大段文本分割为更小的文本片段。这些片段将作为后续嵌入处理的基础。常见的分词处理方式有字符数量切分、分隔符切分等。本系统使用chonkie库进行分词，该库包含了许多常见的基础分词模式以使用。

**（3）文本嵌入（Embedding）**：通过预训练的语言模型（如BERT、GPT等），将文本片段转化为高维向量表示。这些向量能够捕捉文本的语义信息，便于后续的相似度计算和检索。本系统使用基于RocketQA的rocketqa-zh-base-query-encoder模型进行文本向量化。

**（4）向量数据库构建**：将生成的文本通过嵌入模型转换为高维向量表示，并存储于专门的向量数据库中。向量数据库能够高效地管理和检索高维向量数据，其核心优势在于支持快速的近似最近邻搜索，从而确保在用户发起查询时，系统能够毫秒级响应并返回语义最相关的内容。本系统采用Facebook 的Faiss作为向量数据库解决方案，Faiss通过优化的索引结构和并行计算技术，能够在大规模向量数据集上实现亚秒级的相似性搜索，减少了流程中用于检索的时间。

3.2 用户查询处理

用户查询处理是将教师的查询请求转化为系统能够理解的形式，并从知识库中检索出相关内容，最终生成教学大纲或教学案例的过程。在此步骤中，主要难点是如何有效地混合应用RAG与Agent技术，通过合适的prompt编写，实现高质量的内容生成。用户查询处理的流程如图3所示。



**图3 用户查询流程**

**（1）用户查询与标准化**：教师通过系统界面输入查询请求，如“生成《机器学习》课程的教学大纲”或“设计一个关于深度学习的开放性讨论题目”。然后系统使用一个Agent对用户查询进行标准化、结构化的处理，补充额外信息，提高后续向量匹配和检索的精确度和相关性。

**（2）最近邻算法与数据检索**：使用最近邻算法（HSNW）在向量数据库中进行相似度匹配，找到与用户查询最相关的top-k文本片段或知识条目，作为生成教学内容的参考，输入进问答LLM的上下文。这里的文本片段数k会根据数据库内条目总数推荐一个合适的值，也会给用户暴露调整该参数的方法。

**（3）LLM问答生成**：利用大语言模型（LLM）对检索到的相关内容进行进一步处理，生成详细的教学大纲、教学案例或开放性讨论题目。为了增强生成产物的相关性，LLM需要能够记忆并根据上下文生成连贯且符合教学需求的内容。此外，还可以测试多种不同的prompt，选择具有最优输出的propmt进行使用。随着LLM的飞速发展，大模型的性能与输出质量也在日益提高。我们需要添加切换不同模型API的功能以适应大模型的推陈出新；并且也需要提供本地运行模型的接口，以便满足数据安全性的审计要求。

**（4）输出处理与格式检查**：这一步系统需要对生成的教学内容进行格式化和符号检查，确保输出的内容符合中国教学要求。例如，需要把大模型输出的公式符号统一为中国教学所需的标准。

**（5）Markdown渲染与数据导出**：将生成的教学内容渲染为Markdown格式，便于渲染到网页上。教师可以选取所需内容，插入到课件中。同时，系统支持将生成的内容导出为多种格式（如PDF、PPT等），方便教师在不同场景下使用。

3.3 其他**工程实践**

3.3.1 用户界面

除了大模型输出的核心功能之外，本系统在设计与开发过程中还充分融入了良好的工程实践，以提升用户体验和系统的可维护性。本项目的用户界面采用了现代化的前后端分离设计架构，前端部分基于React框架构建，并结合Radix UI组件库来实现高效、直观且美观的用户交互界面；后端则选用了Python语言结合FastAPI框架，以其高性能和异步处理能力为系统提供稳定而高效的服务支持。数据存储方面，对于非向量数据的存储需求，系统采用了轻量级的嵌入式数据库SQLite，其不仅能够满足基本的存储功能，还因其无需独立服务器支持的特性显著减小了系统的资源占用体积，使得系统整体更加轻便和易于部署。

前端所使用的React框架和后端的FastAPI框架在其他诸多实践项目中均得到了广泛应用，拥有成熟的生态系统和庞大的社区支持，这为本项目的开发和后续维护提供了坚实的技术保障。此外，Radix UI的使用进一步提升了前端开发的效率和一致性，而SQLite的选择则在保证数据管理能力的同时，兼顾了系统的便携性和低依赖性。这种技术选型的组合，旨在为用户提供流畅的操作体验，同时为开发者提供灵活的扩展空间。

3.3.2 容器化部署

Python项目的部署过程往往较为复杂，尤其是在面对众多依赖包的情况下，环境配置的差异可能导致部署过程中出现不可预见的错误。为了解决这一问题，本项目计划采用容器化技术对程序进行打包和部署，以确保其在不同运行环境中的一致性和稳定性。具体而言，项目将使用Docker作为容器化工具，通过编写Dockerfile将应用程序及其所有依赖项（如Python解释器、FastAPI框架等）封装到一个独立的容器镜像中。

这种方式不仅能够简化部署流程，还能有效避免因服务器环境差异导致的兼容性问题，使得项目既可以在高性能服务器上轻松运行，也可以在个人电脑上快速部署和测试。此外，容器化技术还带来了版本控制的便利，开发者可以通过容器镜像的版本管理实现功能的快速迭代与回滚，进一步提升开发效率。对于用户而言，容器化部署意味着他们无需手动配置复杂的运行环境，只需拉取预构建的镜像并运行即可，从而大幅降低了使用门槛。未来，随着项目规模的扩展，容器化还为引入容器编排工具（如Kubernetes）提供了可能性，以便在分布式系统中实现更高的可用性和扩展性。

# 4 进度安排

1～2周 查阅RAG与教育AI领域文献，完成关键技术综述翻译，搭建开发环境；

3～4周 完善文献综述，撰写开题报告，确定系统架构与技术选型；

5～6周 分析教育知识库构建流程，完成数据采集与预处理；

7～9周 实现RAG核心模块：向量化检索、生成模型微调、Agent决策逻辑开发；

10～13周 系统集成与多场景测试（如课件生成、智能答疑、学习路径规划）；

14～16周 撰写论文，优化实验指标；

17周 答辩准备：制作PPT、预答辩演练；

18周 正式答辩与系统演示。

# 5 参考文献

1. LIU Z, OGUZ B, ZHAO C, et al. LLM-QAT: Data-Free Quantization Aware Training for Large Language Models[Z].2023.
2. HINTON G, VINYALS O, DEAN J. Distilling the Knowledge in a Neural Network[Z].2015.
3. DELÉTANG G, RUOSS A, DUQUENNE P A, et al. Language Modeling Is Compression[Z].2024.
4. RAWTE V, SHETH A, DAS A. A Survey of Hallucination in Large Foundation Models[Z].2023.
5. GOODFELLOW I, BENGIO Y, COURVILLE A. Deep Learning[M]. Cambridge, MAMIT Press, 2016.
6. 张浩然, 郝文宁, 靳大尉, et al. DF-RAG:基于查询重写和知识选择的检索增强生成方法[J].计算机科学: 1-12.
7. 王博楷, 牛本, 刘玲丽. 基于 RAG 的供应链智能问答模型[J/OL].Operations Research and Fuzziology, 2024: 637-644. <http://www.hanspub.org/journal/PaperInformation.aspx?PaperID=>[102653](http://www.hanspub.org/journal/PaperInformation.aspx?PaperID=102653).DOI:[10.12677/orf.2024.146564](https://doi.org/10.12677/orf.2024.146564).
8. EDGE D, TRINH H, CHENG N, et al. From Local to Global: A Graph RAG Approach to Query-Focused Summarization[Z].2025.
9. ASAI A, WU Z, WANG Y, et al. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection[Z].2023.
10. 张浩然, 郝文宁, 靳大尉, et al. DF-RAG:基于查询重写和知识选择的检索增强生成方法[J].计算机科学: 1-12.
11. RACKAUCKAS Z. Rag-Fusion: A New Take on Retrieval Augmented Generation[J/ OL]. International Journal on Natural Language Computing, 2024, 13(1): 37-47. DOI:[10.5121/ijnlc.2024.13103](https://doi.org/10.5121/ijnlc.2024.13103).
12. SINGH A, EHTESHAM A, KUMAR S, et al. Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG[Z]. 2025.
13. RAVURU C, SAKHINANA S S, RUNKANA V. Agentic Retrieval-Augmented Generation for Time Series Analysis[Z]. 2024.
14. PAYONG A, MUKHERJEE S. RAG, AI Agents, and Agentic RAG: An In-Depth Review and Comparative Analysis[EB/OL]. DigitalOcean, 2024. <https://www.digitalocean.com/community/>[conceptual-articles/rag-ai-agents-agentic-rag-comparative-analysis](https://www.digitalocean.com/community/conceptual-articles/rag-ai-agents-agentic-rag-comparative-analysis).
15. REMOTE. Teaching Assistant - Powered by LLM and RAG[EB/OL]. Devpost, 2024. <https://devpost.com/software/teaching-assistant-powered-by-llm-and-rag>.
16. MYSCALE. Revolutionizing Education with RAG AI: Real Case Studies Revealed[EB/ OL]. MyScale, 2024. <https://myscale.com/blog/rag-ai-education-impact-case-studies/>.

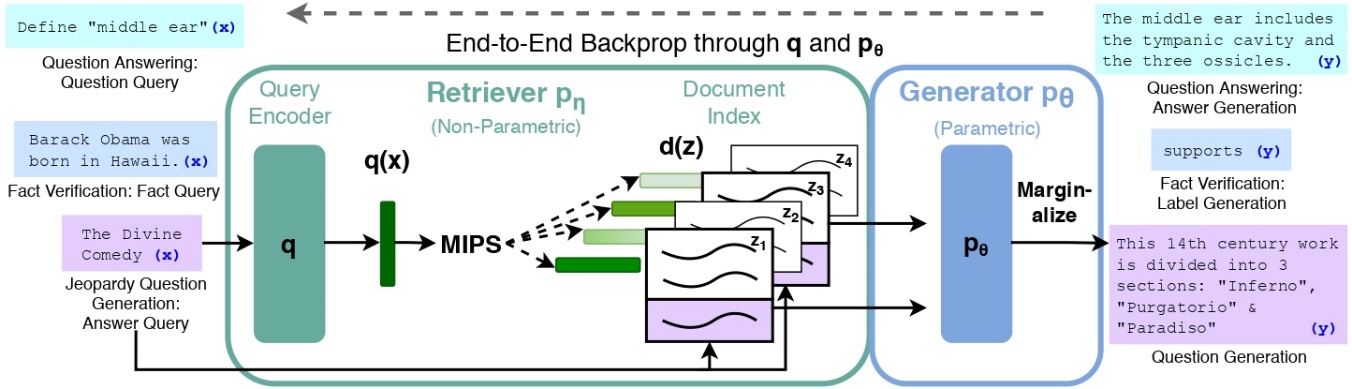
用于知识密集型NLP任务的检索增强生成

Patrick Lewis, Ethan Perez,  
Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, Douwe Kiela

摘要：大型预训练语言模型已被证明可以在其参数中存储事实知识，并在下游NLP任务中微调时达到最先进的结果。然而，它们访问和精确操控知识的能力仍然有限，因此在知识密集型任务中，其性能落后于特定任务的架构。此外，为其决策提供出处并更新其世界知识仍然是开放的研究问题。具有可微访问机制的预训练模型对显式非参数内存的研究迄今仅限于抽取式下游任务。我们探索了一种通用的检索增强生成（RAG）微调方法——这种模型结合了预训练的参数和非参数内存用于语言生成。我们引入了RAG模型，其中参数内存是一个预训练的seq2seq模型，非参数内存是一个维基百科的密集向量索引，通过预训练的神经检索器访问。我们比较了两种RAG形式，一种是在整个生成序列中基于相同的检索段落进行条件化，另一种是可以为每个标记使用不同的段落。我们在广泛的知识密集型NLP任务上微调和评估我们的模型，并在三个开放域QA任务上设定了新的技术标准，超越了参数seq2seq模型和特定任务的检索和抽取架构。对于语言生成任务，我们发现RAG模型生成的语言比最先进的仅参数seq2seq基线更具体、多样和现实。

# 1 绪论

内容预训练的神经语言模型已被证明可以从数据中学习大量深入的知识。它们可以在没有外部记忆访问的情况下做到这一点，作为参数化的隐式知识库。虽然这一发展令人兴奋，但此类模型确实存在缺点：它们不能轻易扩展或修改其记忆，不能直接提供对其预测的洞察，并且可能产生“幻觉”。结合参数记忆和非参数（即基于检索的）记忆的混合模型可以解决其中的一些问题，因为知识可以直接修改和扩展，并且可以检查和解释访问到的知识。REALM和ORQA是最近引入的两种结合了掩码语言模型和可微检索器的模型，已显示出有希望的结果，但仅探索了开放域的抽取式问答。在这里，我们将混合参数和非参数记忆引入到NLP的“主力”，即序列到序列（seq2seq）模型中。



**图1 我们使用的方法概述。我们结合了一个预训练的检索器（查询编码器 + 文档索引）和一个预训练的序列到序列模型（生成器），并进行端到端的微调。对于查询x，我们使用最大内积搜索（MIPS）来找到前K个文档zi。对于最终预测y，我们将z视为一个潜在变量，并对给定不同文档的序列到序列预测进行边缘化处理。**

我们通过一种通用的微调方法为预训练的参数记忆生成模型赋予非参数记忆，我们称之为检索增强生成（RAG）。我们构建RAG模型，其中参数记忆是预训练的seq2seq变压器，非参数记忆是Wikipedia的密集向量索引，通过预训练的神经检索器访问。我们在一个端到端训练的概率模型中结合这些组件（图1）。检索器（密集段落检索器，以下简称DPR）提供基于输入的潜在文档，seq2seq模型（BART）然后基于这些潜在文档和输入生成输出。我们通过top-K近似对潜在文档进行边缘化，可以是基于每个输出（假设同一文档负责所有标记）或基于每个标记（不同文档负责不同标记）。像T5或BART一样，RAG可以在任何seq2seq任务上进行微调，从而共同学习生成器和检索器。

之前有大量工作提出了丰富系统的非参数记忆架构，这些架构是为特定任务从头开始训练的，例如记忆网络，堆栈增强网络和记忆层。相比之下，我们探索了一种设置，其中参数和非参数记忆组件都经过预训练并预加载了丰富的知识。关键是，通过使用预训练的访问机制，访问知识的能力无需额外训练即可实现。

我们的结果突出了将参数和非参数记忆与生成结合起来的好处，特别是对于知识密集型任务——人类在没有外部知识来源的情况下无法合理执行的任务。我们的RAG模型在开放自然问题、WebQuestions和CuratedTrec上取得了最先进的结果，并且在TriviaQA上大大超越了使用专门预训练目标的最新方法。尽管这些是抽取任务，我们发现不受约束的生成优于以前的抽取方法。对于知识密集型生成，我们在MS-MARCO和Jeopardy问题生成上进行了实验，发现我们的模型生成的响应比BART基线更具事实性、具体性和多样性。对于FEVER事实验证，我们的结果在使用强检索监督的最先进的管道模型中相差4.3％。最后，我们展示了可以替换非参数记忆以随着世界的变化更新模型的知识。

# 2 方法

我们探索RAG模型，该模型使用输入序列 来检索文本文档 ，并在生成目标序列 时将其用作附加上下文。如图1所示，我们的模型利用了两个组件：（i）一个检索器 ，其参数为 ，根据查询 返回（前K截断的）文本段落分布；（ii）一个生成器 ，其参数为 ，根据前 个标记 的上下文、原始输入 和检索到的段落 生成当前标记。

为了端到端地训练检索器和生成器，我们将检索到的文档视为潜在变量。我们提出了两种模型，以不同的方式对潜在文档进行边缘化，以生成文本分布。在*RAG-Sequence*方法中，模型使用相同的文档来预测每个目标标记。而第二种*RAG-Token*方法，可以基于不同的文档预测每个目标标记。接下来，我们正式介绍这两种模型，然后描述 和 组件，以及训练和解码过程。

2.1 模型

2.1.1 RAG-Sequence 模型

RAG-Sequence 模型使用相同的检索文档生成完整的序列。技术上，它将检索到的文档视为单个潜在变量，通过top-K 近似进行边缘化以获得seq2seq概率 。具体来说，使用检索器检索前 K 个文档，生成器为每个文档生成输出序列概率，然后进行边缘化。

2.1.2 RAG-Token 模型

在 RAG-Token 模型中，我们可以为每个目标标记绘制不同的潜在文档并相应地进行边缘化。这允许生成器在生成答案时从多个文档中选择内容。具体来说，使用检索器检索前 K 个文档，然后生成器为每个文档生成下一个输出标记的分布，然后进行边缘化，并对后续输出标记重复此过程。正式地，我们定义：

最后，我们注意到 RAG 可以通过将目标类视为长度为一的目标序列来用于序列分类任务，在这种情况下，RAG-Sequence 和 RAG-Token 是等价的。

2.2 检索器：DPR

检索组件 基于 DPR。DPR 遵循双编码器架构：

其中 是由 BERTBASE 文档编码器生成的文档的密集表示，是由查询编码器生成的查询表示，也基于 BERTBASE。计算 ，即具有最高先验概率 的 个文档 的列表，是一个最大内积搜索 (MIPS) 问题，可以在子线性时间内近似解决。我们使用预训练的 DPR 双编码器来初始化我们的检索器并构建文档索引。该检索器被训练用于检索包含 TriviaQA问题和 Natural Questions答案的文档。我们将文档索引称为非参数内存。

2.3 生成器：BART

生成器组件 可以使用任何编码器-解码器进行建模。我们使用 BART-large，一个具有 400M 参数的预训练 seq2seq transformer。在从 BART 生成时，我们简单地将输入 x 与检索到的内容 z 连接起来。BART 使用去噪目标和多种不同的噪声函数进行预训练。它在多种生成任务上获得了最先进的结果，并且优于相似大小的T5 模型。我们将 BART 生成器参数 称为参数内存。

2.4 训练

我们在没有任何关于应检索哪个文档的直接监督的情况下联合训练检索器和生成器组件。给定一个输入/输出对 的微调训练语料库，我们使用Adam的随机梯  
度下降最小化每个目标的负边缘对数似然 。在训练期间更新文档编码  
器 BERTd 是昂贵的，因为它需要像 REALM 在预训练期间那样定期更新文档索引。我们发现这一步对于强性能不是必需的，并保持文档编码器（和索引）固定，仅微调  
查询编码器 BERTq 和 BART 生成器。

2.5 解码

在测试时，RAG-Sequence 和 RAG-Token 需要不同的方法来近似。2.5.1 RAG-Token

RAG-Token 模型可以被视为一个标准的自回归 seq2seq 生成器，其转  
移概率为： 要解码，我们可以  
将 插入标准的束搜索解码器中。

2.5.2 RAG-Sequence

对于 RAG-Sequence，似然 不会分解为传统的每个标记的似然，因此我们不能用单个束搜索来解决它。相反，我们为每个文档 z 运行束搜索，使用 对每个假设进行评分。这产生了一组假设 Y ，其中一些可能没有出现在所有文档的束中。为了估计假设 y 的概率，我们为每个文档 z 运行一个额外的前向传递，其中 y 没有出现在束中，将生成器概率与 相乘，然后在束中对边缘进行概率求和。我们将此解码过程称为“彻底解码”。对于较长的输出序列， 可能会变大，需要许多前向传递。为了更高效的解码，我们可以进一步近似 ，其中y 在从,的束搜索中未生成。这避免了一旦候选集 Y 生成后需要运行额外的前向传递。我们将此解码过程称为“快速解码”。

# 3实验

我们在广泛的知识密集型任务中对 RAG 进行实验。对于所有实验，我们使用单个维基百科转储作为我们的非参数知识源。与Lee 和 Karpukhin 相同，我们使用 2018 年 12 月的转储。每篇维基百科文章被分割成不相交的 100 字块，总共 21M 文档。我们使用文档编码器为每个文档计算嵌入，并使用 FAISS构建一个单一的 MIPS索引，采用分层可导航小世界近似以实现快速检索。在训练期间，我们为每个查询检索前 k 个文档。我们在训练中考虑 k ∈{5, 10}，并使用开发数据在测试时设置 k。现在我们讨论每个任务的实验细节。

3.1 开放域问答

开放域问答（QA）是一个重要的现实应用，也是知识密集型任务的常见测试平台 。我们将问题和答案视为输入输出文本对 (x, y)，通过直接最小化答案的负对数似然来训练 RAG。我们将 RAG 与流行的抽取式 QA 范式进行比较，其中答案是从检索到的文档中提取的跨度，主要依赖于非参数知识。我们还与“闭卷 QA”方法进行比较，这些方法与 RAG 一样生成答案，但不利用检索，而是纯粹依赖于参数知识。我们考虑四个流行的开放域 QA 数据集：Natural Questions (NQ)，TriviaQA(TQA)，WebQuestions (WQ) 和 CuratedTrec (CT)。由于 CT 和 WQ 较小，我们遵循 DPR，通过用我们的 NQ RAG 模型初始化 CT和 WQ 模型。我们使用与先前工作相同的训练/开发/测试分割并报告精确匹配（EM）分数。对于 TQA，为了与 T5 进行比较，我们还在 TQA Wiki 测试集上进行评估。

3.2 抽象问答

RAG 模型可以超越简单的抽取式QA，通过自由形式的抽象文本生成来回答问题。为了测试 RAG 在知识密集型环境中的自然语言生成（NLG），我们使用 MSMARCO NLG任务 v2.1。该任务包括问题、从搜索引擎检索到的十个黄金段落以及从检索到的段落中注释的完整句子答案。我们不使用提供的段落，只使用问题和答案，将MSMARCO视为开放域抽象 QA 任务。MSMARCO 有些问题无法在没有黄金段落的情况下以匹配参考答案的方式回答，例如“加利福尼亚火山的天气如何？”因此不使用黄金段落时性能会较低。我们还注意到，一些 MSMARCO 问题无法仅使用维基百科回答。在这里，RAG 可以依赖参数知识生成合理的响应。

3.3 Jeopardy问题生成

为了评估 RAG 在非 QA 环境中的生成能力，我们研究开放域问题生成。我们没有使用标准开放域 QA 任务中的问题，这些问题通常由简短、简单的问题组成，而是提出了生成 Jeopardy 问题的更具挑战性的任务。Jeopardy 是一种不寻常的格式，试图从关于实体的事实中猜测该实体。例如，“世界杯”是问题“1986 年墨西哥成为第一个两次举办这一国际体育赛事的国家”的答案。由于 Jeopardy 问题是精确的事实陈述，生成基于其答案实体的 Jeopardy 问题构成了一个具有挑战性的知识密集型生成任务。

我们使用 SearchQA 的分割，包括 100K 训练、14K 开发和 27K 测试示例。由于这是一个新任务，我们训练了一个 BART 模型进行比较。然后，我们使用 SQuAD调整的 Q-BLEU-1 指标进行评估。Q-BLEU 是 BLEU 的一个变体，对匹配实体赋予更高的权重，与人类判断的问题生成相关性比标准指标更高。我们还进行了两次人工评估，一次评估生成的事实性，一次评估特异性。我们将事实性定义为陈述是否可以通过可信的外部来源证实，特异性定义为输入和输出之间的高度相互依赖性。我们遵循最佳实践，使用成对比较评估。评估者会看到一个答案和两个生成的问题，一个来自 BART，一个来自 RAG。然后他们被要求选择四个选项之一——问题 A 更好，问题 B 更好，两个都好，或者都不好。

3.4 事实验证

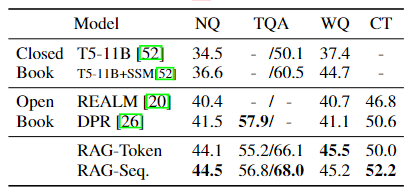
FEVER要求对自然语言声明进行分类，判断其是否被维基百科支持或反驳，或者是否没有足够的信息来决定。该任务需要从维基百科中检索与声明相关的证据，然后对这些证据进行推理，以判断声明是正确、错误还是仅凭维基百科无法验证。FEVER 是一个检索问题，结合了一个具有挑战性的蕴涵推理任务。它还为探索 RAG 模型处理分类而非生成的能力提供了一个合适的测试平台。我们将 FEVER 类标签（支持、反驳或信息不足）映射到单个输出标记，并直接用声明-类对进行训练。至关重要的是，与大多数其他 FEVER 方法不同，我们不使用检索到的证据的监督。在许多现实应用中，检索监督信号不可用，不需要这种监督的模型将适用于更广泛的任务。我们探索了两种变体：标准的三分类任务（支持/反驳/信息不足）和Thorne与Vlachos研究的二分类（支持/反驳）任务。在这两种情况下，我们都报告标签准确性。

# 4 结果

4.1 开放域问答

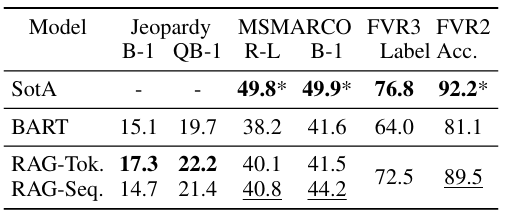
表1显示了 RAG 与最先进模型的结果。在所有四个开放域 QA 任务中，RAG 设定了新的最先进水平（仅在 TQA 的 T5 可比拆分上）。RAG 结合了“闭卷”（仅参数）方法的生成灵活性和“开卷”基于检索的方法的性能。与 REALM 和 T5+SSM 不同，RAG在没有昂贵的、专门的“显著跨度掩码”预训练的情况下享有强大的结果。值得注意的是，RAG 的检索器是使用 DPR 的检索器初始化的，后者在自然问题和 TriviaQA上使用检索监督。RAG 与 DPR QA 系统相比，后者使用基于 BERT 的“交叉编码器”对文档进行重新排序，并使用抽取式阅读器。RAG 表明，既不需要重新排序器也不需要抽取式阅读器即可实现最先进的性能。即使可以提取答案，生成答案也有几个优势。包含关于答案的线索但不包含答案逐字的文档仍然可以促成正确答案的生成，这在标准抽取方法中是不可能的，从而导致对文档的更有效边缘化。此外，即使在任何检索到的文档中没有正确答案，RAG 也可以生成正确答案，在 NQ 中这种情况下实现11.8％的准确率，而抽取模型将得分为0％。

**表1 开放域QA测试分数**



备注：对于TQA，左列使用开放域 QA 的标准测试集，右列使用 TQA-Wiki 测试集。有关更多详细信息，请参见附录D。

**表2 生成和分类测试分数**



备注：\*号表示使用上下文/证据。最佳模型没有下划线。

4.2 抽象问答

如表2所示，RAG-Sequence 在开放MS-MARCO NLG 得分上优于BART 2.6 Bleu 和2.6 Rouge-L。RAG 接近最先进的模型性能，这令人印象深刻，因为（i）这些模型访问具有生成参考答案所需的特定信息的金色段落，（ii）许多问题在没有金色段落的情况下是无法回答的，并且（iii）并非所有问题都可以仅从 Wikipedia 中回答。表3显示了我们模型生成的一些答案。定性地，我们发现 RAG 模型比 BART 更少产生幻觉，并且更频繁地生成事实正确的文本。稍后，我们还将展示 RAG 生成比 BART 生成更具多样性（见 §4.5）。

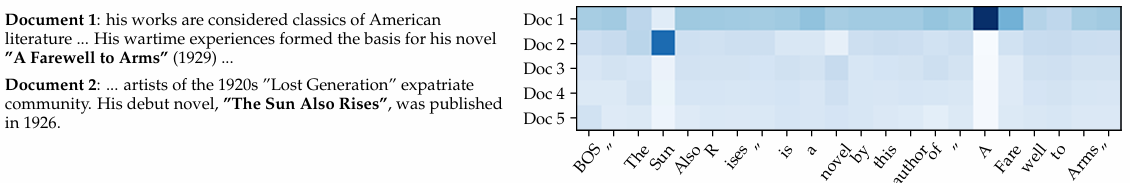
4.3 Jeopardy 问题生成

表2显示 RAG-Token 在 Jeopardy 问题生成上优于 RAG-Sequence，两种模型在 Q-BLEU-1 上均优于 BART。表4显示了 452 对 BART 和 RAG-Token 生成的人工评估结果。评估者指出，只有 7.1％的情况下 BART 比 RAG 更具事实性，而 RAG 在 42.7％的情况下更具事实性，并且在进一步的 17％的情况下，RAG 和 BART 都具有事实性，清楚地展示了 RAG 在任务上的有效性超过了最先进的生成模型。评估者还发现 RAG生成在很大程度上更具体。表3显示了每个模型的典型生成。

Jeopardy问题通常包含两个独立的信息片段，而RAG-Token可能表现最佳，因为它能够生成结合多个文档内容的回答。图4展示了一个例子。当生成“Sun”时，后验概率在文档2中较高，该文档提到了“The Sun Also Rises”。同样，当生成“A Farewell to Arms”时，文档1的后验概率占主导地位。有趣的是，在生成每本书的第一个词后，文档的后验概率趋于平缓。这一观察表明，生成器可以在不依赖特定文档的情况下完成标题。换句话说，模型的参数化知识足以完成标题。我们通过向仅使用BART的基线模型提供部分解码“The Sun”来验证这一假设。BART完成了生成“The Sun Also Rises is a novel by this author of The Sun Also Rises”，表明标题“The Sun Also Rises”存储在BART的参数中。类似地，BART会将部分解码“The Sun Also Rises is a novel by this author of A”补充为“The Sun Also Rises is a novel by this author of A Farewell to Arms”。这个例子展示了参数化和非参数化记忆如何协同工作——非参数化组件有助于引导生成，提取存储在参数化记忆中的特定知识。

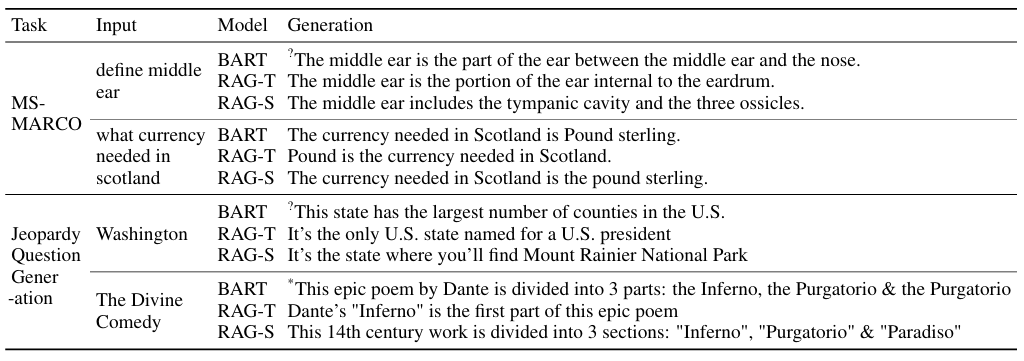
4.4 实际验证

表3展示了我们在FEVER数据集上的结果。对于三分类任务，RAG的得分与最先进的模型相差在4.3%以内，而这些最先进的模型是复杂的流水线系统，具有特定领域的架构和大量的工程优化，并且使用了中间检索监督进行训练，而RAG并不需要这些。



**图4-1 RAG-Token 文档后验概率  对于输入“Hemingway”在 Jeopardy 生成任务中每个生成词元的分布，检索了 5 篇文档。当生成“A Farewell to Arms”时，文档 1 的后验概率较高；当生成“The Sun Also Rises”时，文档 2 的后验概率较高。**

**表3 生成任务中的示例。RAG模型生成的回答更加具体且事实准确。“？”表示事实错误的回答，“\*”表示部分正确的回答。**



对于二分类任务，我们与Thorne和Vlachos的工作进行了比较，他们训练了RoBERTa模型，在给定黄金证据句子的情况下将声明分类为真或假。RAG的准确率与该模型相差在2.7%以内，尽管RAG仅提供了声明并自行检索证据。我们还分析了RAG检索到的文档是否与FEVER中标注为黄金证据的文档一致。我们计算了RAG检索到的前k篇文档与黄金证据标注之间的文章标题重叠率。我们发现，在71%的情况下，检索到的第一篇文档来自黄金文章，而在90%的情况下，黄金文章出现在检索到的前10篇文章中。

4.5 额外结果

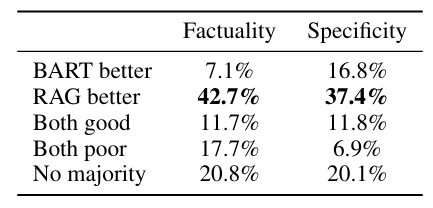
**生成多样性** 第4.3节显示，RAG模型在Jeopardy问题生成任务中比BART更具事实性和具体性。根据最近关于促进多样性解码的研究，我们还通过计算不同模型生成的不同n-gram与总n-gram的比例来研究生成多样性。表5显示，RAG-Sequence的生成比RAG-Token更具多样性，而两者都比BART显著更具多样性，且无需任何促进多样性的解码方法。

**检索消融实验** RAG的一个关键特性是学习为任务检索相关信息。为了评估检索机制的有效性，我们进行了消融实验，在训练期间冻结检索器。如表6所示，学习检索提高了所有任务的结果。

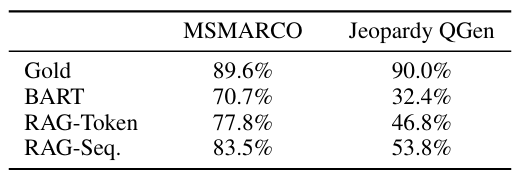
我们将RAG的密集检索器与基于词重叠的BM25检索器进行了比较。在这里，我们用固定的BM25系统替换RAG的检索器，并在计算 时使用BM25检索分数作为logits。表6显示了结果。对于FEVER任务，BM25表现最佳，可能是因为FEVER声明高度以实体为中心，因此非常适合基于词重叠的检索。可微检索在所有其他任务上提高了结果，尤其是在开放域问答任务中，这一点至关重要。

**索引热交换** 像RAG这样的非参数记忆模型的一个优势是，知识可以在测试时轻松更新。像T5或BART这样的纯参数模型需要进一步训练才能随着世界的变化更新其行为。为了证明这一点，我们使用2016年12月的DrQA维基百科转储构建了一个索引，并将使用此索引的RAG输出与主要结果中使用的较新索引（2018年12月）进行比较。我们准备了一份在这两个日期之间发生变化的世界领导人名单，并使用模板“Who is {position}?”（例如“Who is the President of Peru?”）来查询我们的NQ RAG模型。RAG使用2016年索引对2016年世界领导人的回答正确率为70%，使用2018年索引对2018年世界领导人的回答正确率为68%。索引不匹配时的准确率较低（使用2018年索引和2016年领导人时为12%，使用2016年索引和2018年领导人时为4%）。这表明我们可以通过简单地替换RAG的非参数记忆来更新其世界知识。

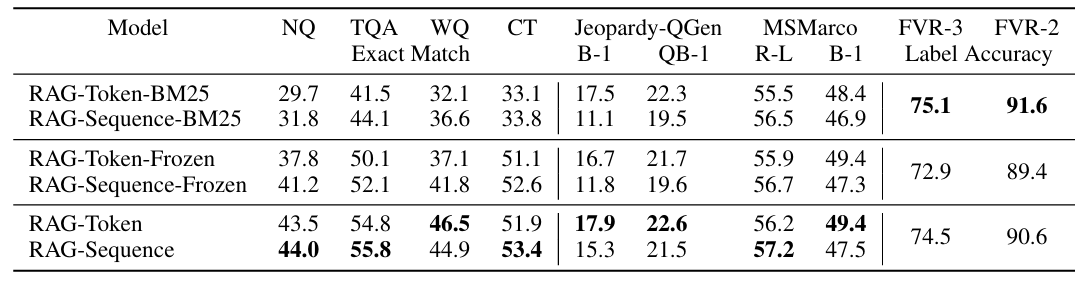
**表4 Jeopardy问题生成任务的人工评估结果**



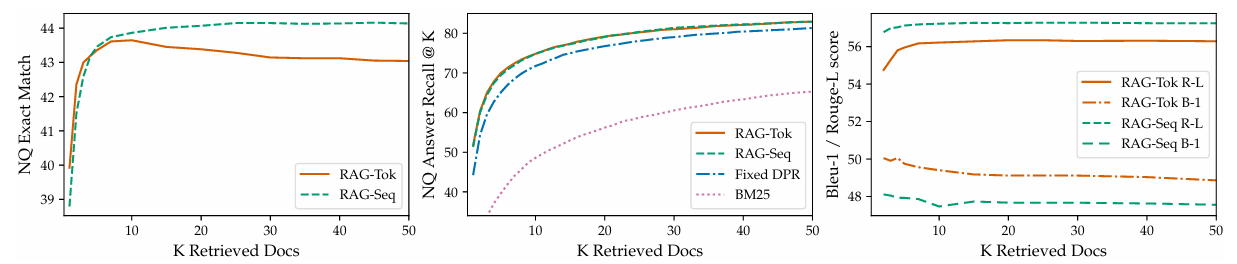
**表5 生成任务中不同三元组与总三元组的比例**



**表6 在开发集上的消融实验。由于FEVER是一个分类任务，两种RAG模型是等效的。**



检索更多文档的效果 模型在训练时使用5或10个检索到的潜在文档，我们未观察到它们之间的性能有显著差异。我们可以在测试时灵活调整检索文档的数量，这可能会影响性能和运行时间。左图显示，在测试时检索更多文档会单调提升RAG-Sequence在开放域问答任务中的表现，但RAG-Token的性能在检索10个文档时达到峰值。右图显示，检索更多文档会提高RAG-Token的Rouge-L分数，但会以Bleu-1分数为代价，而这一效果在RAG-Sequence中不太明显。



**图3 左图：随着检索文档数量的增加，NQ任务的表现。中图：NQ任务中的检索召回表现。右图：随着检索文档数量的增加，MS-MARCO任务的Bleu-1和Rouge-L分数。**

# 5 相关工作

**单任务检索** 先前的研究表明，当单独考虑时，检索可以提高各种NLP任务的性能。这些任务包括开放域问答、事实核查、事实补全、长形式问答、维基百科文章生成、对话、翻译以及语言建模。我们的工作统一了以往将检索融入单个任务的成功经验，表明基于检索的单一架构能够在多个任务中实现强大的性能。

**通用NLP架构** 先前关于NLP通用架构的研究在不使用检索的情况下取得了巨大成功。研究表明，经过微调的单一预训练语言模型可以在GLUE基准测试中的各种分类任务上实现强大的性能。GPT-2进一步证明，单一的从左到右预训练语言模型可以在判别性和生成性任务中均表现出色。为了进一步提升性能，BART和T5提出了单一的预训练编码器-解码器模型，利用双向注意力机制在判别性和生成性任务中实现更强的表现。我们的工作旨在通过学习检索模块来增强预训练的生成性语言模型，从而扩展单一统一架构可能适用的任务范围。

学习检索 在信息检索领域，学习检索文档的研究已经取得了显著进展，尤其是最近使用与我们类似的预训练神经语言模型的研究。一些工作优化了检索模块以辅助特定的下游任务，例如问答，使用搜索、强化学习或潜在变量方法（如我们的工作）。这些成功案例利用不同的基于检索的架构和优化技术在单一任务上实现了强大的性能，而我们的研究表明，单一的基于检索的架构可以通过微调在多种任务上实现强大的性能。

**基于记忆的架构** 我们的文档索引可以被视为神经网络关注的大型外部记忆，类似于记忆网络。同时期的一项研究学习检索输入中每个实体的训练嵌入，而不是像我们的工作那样检索原始文本。其他研究通过关注事实嵌入来提高对话模型生成事实文本的能力。我们记忆的一个关键特征是它由原始文本而非分布式表示组成，这使得记忆既具有人类可读性，为模型提供了一种可解释性，又具有人类可写性，使我们能够通过编辑文档索引动态更新模型的记忆。这种方法也被用于知识密集型对话中，其中生成器直接以检索到的文本为条件，尽管是通过TF-IDF而非端到端学习的检索获得的。

**检索-编辑方法** 我们的方法与检索-编辑风格的方法有一些相似之处，后者为给定输入检索类似的训练输入-输出对，然后进行编辑以提供最终输出。这些方法在多个领域（包括机器翻译和语义解析）中已被证明是成功的。我们的方法有几个不同之处，包括较少强调对检索到的项目进行轻微编辑，而是更注重从多个检索到的内容中聚合信息，以及学习潜在检索并检索证据文档而非相关的训练对。尽管如此，RAG技术在这些场景中可能表现良好，并可能成为未来有前景的研究方向。

# 6 讨论

在本工作中，我们提出了结合参数化和非参数化记忆的混合生成模型。我们展示了RAG模型在开放域问答任务中取得了最先进的结果。我们发现，与纯参数化的BART相比，人们更倾向于RAG的生成结果，认为RAG更具事实性和具体性。我们对学习到的检索组件进行了深入研究，验证了其有效性，并展示了如何通过热交换检索索引来更新模型而无需重新训练。在未来的工作中，研究这两个组件是否可以从头开始联合预训练可能会取得丰硕成果，无论是使用类似于BART的去噪目标还是其他目标。我们的工作开辟了关于参数化和非参数化记忆如何交互以及如何最有效地结合它们的新研究方向，展示了在多种NLP任务中应用的潜力。

# 7 更广泛的影响

与之前的工作相比，本研究带来了几项积极的社会效益：它更牢固地基于真实事实知识（在本例中为维基百科），减少了生成中的“幻觉”，使生成内容更具事实性，并提供了更多的控制性和可解释性。RAG可以应用于多种直接惠及社会的场景，例如通过为其配备医学索引并询问其相关领域的开放域问题，或帮助人们更高效地完成工作。

然而，这些优势也伴随着潜在的缺点：维基百科或任何潜在的外部知识源可能永远不会完全真实且毫无偏见。由于RAG可以用作语言模型，类似于GPT-2的担忧在这里也适用，尽管程度可能较轻，包括可能被用于生成滥用、伪造或误导性的新闻或社交媒体内容；冒充他人；或自动化生成垃圾邮件/钓鱼内容。先进的语言模型也可能在未来几十年内导致各种工作的自动化。为了减轻这些风险，可以部署AI系统来对抗误导性内容和自动化垃圾邮件/钓鱼行为。

# 8 致谢

作者感谢评审人员对本文提出的深思熟虑和建设性反馈，以及HuggingFace在开源RAG模型代码方面的帮助。作者还感谢Kyunghyun Cho和Sewon Min的富有成效的讨论和建议。EP感谢NSF研究生研究奖学金的支持。PL得到了FAIR博士项目的支持。

# 附录A 实现细节

对于开放域问答任务，我们报告了使用15个检索文档的RAG-Token模型的测试结果。对于RAG-Sequence模型，我们报告了使用50个检索文档的测试结果，并采用了Thorough Decoding方法，因为答案通常较短。我们使用贪心解码进行问答任务，因为未发现束搜索能提升结果。对于开放域MSMarco和Jeopardy问题生成任务，我们报告了使用10个检索文档的RAG-Token和RAG-Sequence模型的测试结果，并训练了一个BART-large模型作为基线。我们使用束大小为4，并对RAG-Sequence模型采用Fast Decoding方法，因为Thorough Decoding未提升性能。

# 附录B 人类评估

图4展示了人工评估的用户界面。为了避免屏幕位置带来的偏见，每个示例中模型对应的句子A和句子B是随机选择的。我们鼓励标注者使用互联网研究相关主题，并在完整说明标签中提供了详细的指导和示例。我们包含了一些黄金句子以评估标注者的准确性。两名标注者在这些示例上表现不佳，因此他们的标注结果被从最终结果中移除。

# 

**图4 用于事实性人工评估的标注界面。点击“View tool guide”时会弹出详细说明和示例。**

# 附录C 训练设置细节

我们使用Fairseq训练所有RAG模型和BART基线。我们采用混合精度浮点运算进行训练，并在8块32GB NVIDIA V100 GPU上分布式训练，尽管训练和推理可以在单块GPU上运行。我们发现使用FAISS进行最大内积搜索在CPU上已经足够快，因此我们将文档索引向量存储在CPU上，整个维基百科的索引需要100GB的CPU内存。在提交后，我们将代码移植到了HuggingFace Transformers，该版本实现了与之前版本相当的性能，但实现更简洁且更易于使用。此版本也已开源。我们还使用FAISS的压缩工具对文档索引进行了压缩，将CPU内存需求减少到36GB。运行RAG实验的脚本可以在https://github.com/huggingface/transformers/blob/master/examples/rag/README.md找到，RAG模型的交互式演示可以在https://huggingface.co/rag/查看。

# 附录D 更多开放域问答的细节

对于开放域问答任务，通常一个问题会有多个答案标注。这些答案标注在训练期间被抽取式模型利用，因为在准备训练数据时通常会使用所有答案标注来在文档中查找匹配项。对于RAG，我们也利用Natural Questions和WebQuestions的多个标注示例，通过分别训练每个（问答）对来提高准确性。对于TriviaQA，一个问题通常有许多有效答案，其中一些不适合作为训练目标，例如表情符号或拼写变体。对于TriviaQA，如果答案候选未出现在查询的前1000个文档中，我们会将其过滤掉。

**CuratedTrec预处理** CuratedTrec的答案以正则表达式的形式给出，这被认为是其不适合答案生成模型的原因之一。为了解决这个问题，我们采用了一个预处理步骤：首先为每个查询检索前1000个文档，并使用最频繁匹配正则表达式模式的答案作为监督目标。如果未找到匹配项，我们采用一种简单的启发式方法：为每个正则表达式生成所有可能的排列，将正则表达式嵌套树结构中的非确定性符号替换为空格。

**TriviaQA评估设置** 开放域问答社区通常将公开的开发数据集用作测试数据集，因为问答数据集的测试数据通常受限并专用于阅读理解目的。我们使用DPR中使用的数据集划分报告结果，这与开放域问答的常见做法一致。对于TriviaQA，此测试数据集是公开的TriviaQA Web开发集。Roberts等人使用了TriviaQA官方的维基百科测试集。Févry等人遵循了这一惯例以便与Roberts等人进行比较（参见Févry等人的附录）。我们在两个测试集上报告结果，以便与两种方法进行公平比较。我们发现，使用官方的维基百科测试集时，我们的性能远高于使用更常见的开放域测试集，我们将其归因于官方维基百科测试集的问题更容易从维基百科中回答。

# 附录E 更多FEVER的细节

对于FEVER分类任务，我们遵循BART的做法，首先重新生成声明，然后使用最终隐藏状态的表示进行分类，最后在文档之间进行边缘化以获得类别概率。FEVER任务传统上有两个子任务。第一个是将声明分类为“支持”、“反驳”或“信息不足”，这是我们在主论文中探讨的任务。FEVER的另一个子任务涉及从维基百科中提取句子作为支持分类预测的证据。由于FEVER使用了与我们不同的维基百科转储，直接解决此任务并不容易。我们希望在未来工作中解决这个问题。

# 附录F **空文档概率**

我们尝试为RAG添加“空文档”机制，类似于REALM，以建模无法为给定输入检索到有用信息的情况。在这里，如果检索到k个文档，我们会额外“检索”一个空文档，并为空文档预测一个logit，然后在k+1个预测之间进行边缘化。我们通过以下方式探索了对此空文档logit的建模：（i）学习空文档的文档嵌入，（ii）静态学习偏置项，或（iii）使用神经网络预测logit。我们发现这些方法并未提高性能，因此为了简化，我们省略了它们。对于Open MS-MARCO，由于并非总能检索到有用的文档，我们观察到模型学会为不太可能从检索中受益的问题始终检索一组特定的文档，这表明RAG可能不需要空文档机制。

# 附录G **参数**

我们的RAG模型包含DPR的BERT-base查询和文档编码器的可训练参数，每个编码器有1.1亿个参数（尽管我们不训练文档编码器），以及BART-large的4.06亿个可训练参数，总计6.26亿个可训练参数。性能最佳的“封闭书”（仅参数化）开放域问答模型是T5-11B，具有110亿个可训练参数。与我们的模型参数数量最接近的T5模型是T5-large（7.7亿个参数），其在Natural Questions上的得分为28.9 EM，远低于RAG-Sequence的44.5 EM，这表明混合参数化/非参数化模型在开放域问答任务中需要更少的可训练参数即可实现强大的性能。非参数化记忆索引不包含可训练参数，但包含2100万个728维向量，总计153亿个值。这些可以轻松以8位浮点精度存储，以管理内存和磁盘占用。

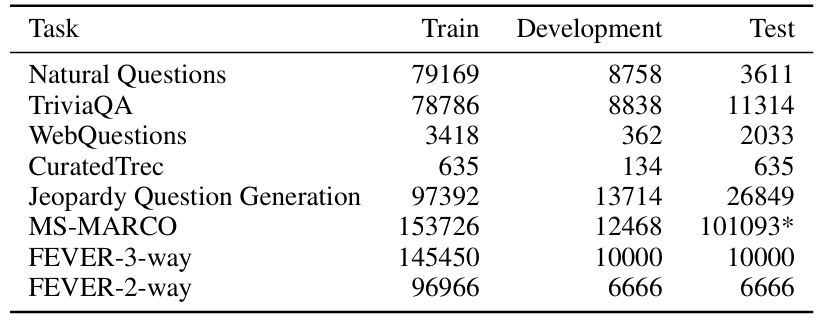
# 附录H **检索崩溃**

在初步实验中，我们观察到，对于某些任务（如故事生成），检索组件会“崩溃”，并学会无论输入如何都检索相同的文档。在这些情况下，一旦检索崩溃，生成器会学会忽略文档，RAG模型的表现将与BART相当。崩溃的原因可能是某些任务对事实知识的需求不够明确，或者目标序列较长，这可能导致检索器的梯度信息较少。Perez等人也发现，在优化检索组件以提高下游任务性能时，会出现虚假的检索结果。

# 附录I **每个数据集的实例数量**

表7中显示了每个数据集的训练、开发和测试数据点的数量。

**表7 使用的数据集中实例的数量。\*此数据的隐藏子集用于评估。**



本文译自：GAO Y, XIONG Y, GAO X, et al. Retrieval-Augmented Generation for Large Language Models: A Survey[Z]. 2024