

# Lab2：基于 LLM 和 RAG 的多跳问题回答系统

## 1. 背景介绍

大型语言模型（LLMs）在理解和生成人类语言方面展现出了卓越的能力。虽然大模型在训练时已经见过海量数据，但在面对事实性问答以及私有数据相关问答时，结合检索增强生成（Retrieval-Augmented Generation, RAG）能够通过提供相关的外部知识，显著提高模型回答问题的准确性。

在 RAG 相关的任务中，多跳问题回答（Multi-hop Question Answering）是指需要模型从多个信息源或文档的不同部分提取信息，并将这些信息综合起来才能得到最终答案的任务。以下是一个简单的例子：

上下文文档：

# 文档 1

上海交通大学创建于1896年，当时名为南洋公学。这是中国最早的高等学府之一，伴随中国近代化过程，经历了晚清、民国和新中国三个历史时期。1921年更名为交通大学上海学校，1928年更名为国立交通大学上海学校。1959年，经国务院批准，交通大学两部分独立建校，交通大学上海部分成立上海交通大学。

...

# 文档 2

上海交通大学软件学院成立于2001年5月。在起步阶段，上海交通大学软件学院以"学校、政府、社会三力合一"、"技术、工程、管理三位一体"、"教学、研发、产业三者统一"为指导原则，进行了一系列改革。软件学院是国家首批37所示范性软件学院之一，在2011年时已经发展了十周年。

...

# 文档 3

曾毓群，1968年3月出生于福建宁德，中华人民共和国企业家。他是宁德时代新能源科技股份有限公司的创办人兼董事长。曾毓群于1989年毕业于上海交通大学船舶工程系，2001年获得华南理工大学电子与信息工程系硕士学位，2006年获得中科院物理研究所凝聚态物理专业博士学位。

多跳问题示例：

Q1. 上海交通大学软件学院的创建时间比上海交通大学的创建时间晚多少年？

Q2. 曾毓群先生本科毕业的学校最初成立时的名字是什么？

回答这些问题需要模型理解并综合不同文档中的信息，这就是多跳推理的典型例子。例如，要回答 Q1，直接使用 Q1 进行 RAG 搜索可能比较难得到准确的答案，一种常用的方式是利用 LLM 将 Q1 分解成 '上海交通大学的创建时间是什么时候？' 和 '上海交通大学软件学院的创建时间是什么时候？' 两个子问题 (Query Decomposition)，然后分别进行 RAG 的搜索，最后将两个子问题的答案结合起来得到 Q1 的答案。类似地，Q2 可以分解为 '曾毓群先生本科毕业的学校是哪所？' 和 '该学校最初成立时的名字是什么？' 两个子问题，依次求解出两个子问题，最后将答案合并即可得到 Q2 的答案。

在本次实验中，我们研究一种简单的方案，即直接使用原问题进行 RAG 检索，然后根据检索到的内容回答问题。因为多跳问题本身的语义和相关文档的语义还是存在一定相似性的，所以在不使用 Query Decomposition 的情况下，也能搜索到一定的相关信息，支撑一个简单版本的 RAG 问答系统。

## 2. 任务目标

本实验的主要目标是构建一个能够有效回答多跳问题的系统, 该系统将利用 LLM 和 RAG 技术. 你需要:

1. 实现一个能够处理 HotpotQA 数据集中的多跳问答的 RAG 系统
2. 使用评估指标 (EM 和 F1) 评估系统性能
3. 进行 case study 分析系统的优缺点
4. 新策略探讨与尝试: 根据实验分析提出自己的改进建议, 并做代码尝试 (可以自己提出改进策略, 或通过查阅论文将最新技术应用均可)

性能提升不是强制考核指标, 有尝试新策略即可

### 3. 数据集

HotpotQA 是一个众包的英文多跳问答数据集, 专为测试模型的多跳推理能力而设计. 为便于实验, 我们只需用其中的 200 条作为实验数据集 (即 LongBench 这个 Benchmark 中挑选的 200 条), 数据集包含以下四个主要字段:

- `id` (str): 问题的标识
- `question` (str): 待回答的问题
- `context` (str): 上下文文档
- `answer` (str): 问题的答案

我们已将准备好的数据整理在了 `hotpotqa_longbench.json` 文件中.

Yang, Zhilin, et al. "HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering." Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018.

Bai, Yushi, et al. "LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding." Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2024.

### 4. 评价指标

**EM (Exact Match):** 只有当系统预测的答案与标准答案完全一致时, 才记为1分, 否则为0分.

**F1 Score:** 计算预测答案和标准答案之间的词级别重叠程度, 综合考虑精确率 (Precision) 和召回率 (Recall):

- 精确率 ( $P$ ): 预测答案中正确词的比例
- 召回率 ( $R$ ): 标准答案中被正确预测的词的比例
- $F1 = 2 \times P \times R / (P + R)$

简单示例:

假设问题是"上海交通大学软件学院的创建时间是什么时候?"

- 标准答案: 2001 年 5 月 (分词后为: 2001、年、5、月, 共4个词)
- 系统预测1: 2001 年 5 月  $\rightarrow$  EM=1, F1=1.0 (完全匹配)
  - 精确率  $P = \frac{4}{4} = 1.0$
  - 召回率  $R = \frac{4}{4} = 1.0$
  - $F1 = \frac{2 \times P \times R}{P + R} = \frac{2 \times 1.0 \times 1.0}{1.0 + 1.0} = 1.0$

- 系统预测2: 2001 年 → EM=0, F1=0.67 (部分匹配)
  - 精确率  $P = \frac{2}{2} = 1.0$
  - 召回率  $R = \frac{2}{4} = 0.5$
  - $F1 = \frac{2 \times P \times R}{P + R} = \frac{2 \times 1.0 \times 0.5}{1.0 + 0.5} = \frac{1.0}{1.5} \approx 0.67$

实验中, 我们将在整个测试集上计算平均EM和F1分数, 以评估系统性能.

## 5. 具体任务

### 5.1 系统搭建

实现一个基于 LLM 和 RAG 技术的多跳问题回答系统, 能够处理 HotpotQA 数据集中的问题. 请注意这一步骤中不需要进行 Query Decomposition, 直接使用原问题进行 RAG 检索即可.

其中的 RAG 部分需要先对上下文文档进行分块 (chunking) 处理, 然后进行检索. 检索部分可以自由选择 [BM25](#) 等简单方法 (对计算资源要求会更低), 或使用 BERT 等预训练模型进行 embedding 向量检索 (如[课堂所讲解](#)) 的方法等.

请注意, 分块时候设定的块大小等参数会影响最终的检索效果, 请合理设置和选择.

### 5.2 评估

使用 EM (精确匹配) 和 F1 分数评估系统性能. 并将每个问题的预测答案保存到 `outputs.json` 文件中, 需要包含 `id` 和 `pred_answer` 两个 key, 文件示例:

```
[{
  "id": "b84411acb9b2e39a68f92ed9e164eb053d3dd9ba4160f0af",
  "pred_answer": "Jānis Strazdiņš"
},
...
]
```

### 5.3 新策略

分析系统表现以及出错的 bad case, 并根据实验分析提出自己的改进建议, 并做代码尝试 (可以自己提出改进策略, 或通过查阅论文将最新技术应用均可)

性能提升不是强制考核指标, 有尝试新策略即可.

如果进行了改进, 提交文件中请保留改进后的输出文件 `outputs_improved.json`.

### 5.4 报告

提交一份简洁的报告, 清晰描述系统实现、评估结果和设计的新策略 (如有). 报告建议不超过 3 页, 主要讲清楚自己的系统实现和结果分析/改进 (如有) 和启发. 如果使用或参考了外部资源的代码, 请在报告中注明.

## 6. 示例代码

我们提供了一个 `example.py` 文件, 其中包含了一个让大模型直接回答问题 (未使用 RAG) 的基础框架和一些评估功能:

- **数据加载:** `load_dataset` 函数可以加载 `hotpotqa_longbench.json` 数据集.
- **模型交互:** `query_chat_model` 函数演示了如何调用课程提供的 `Llama3.1-8B-Instruct` 模型接口 (基于我们搭建的 `vLLM` 服务器).
- **评估函数:** 提供了用于清洗答案的 `normalize_answer` 函数与计算分数的 `compute_em` 和 `compute_f1` 函数.
- **主流程:** `run_evaluation` 函数串联了数据加载、模型调用 (注意: 当前示例未实现 **RAG**, 上下文为空)、评估和结果保存的完整流程.

在当前 `example.py` 中, 我们使用 `Llama3.1-8B-Instruct` 模型不经过 RAG 直接回答问题, 可以得到如下效果 (反复运行多次可能会得到略有波动的结果):

```
Final Average EM: 0.2000, Final Average F1: 0.2467
```

可以看到虽然模型依靠自己的记忆能力, 能够回答一部分问题, 但整体效果较差, 因此需要使用 RAG 技术来提高效果. 本次作业中, 要求 RAG 系统最终达到  $EM > 0.3$ ,  $F1 > 0.4$  即可视为正确实现.

## 7. 模型访问

在调用 LLM 时, 有两种选择:

1. **课程提供的 vLLM 服务器:** 使用我们使用 `vllm` 提供的 `Llama3.1-8B-Instruct` 模型, 参考调用方式可在 `example.py` 中找到, 但需要注意的是该服务器的并发能力有限, 请合理安排使用时间, 避免在截止日期临近时集中使用导致请求响应缓慢
2. **自选 API:** 使用 DeepSeek, OpenAI, Anthropic 等其他任何你喜欢的 LLM 服务, 当然也可以自己用 `vllm` 和 `ollama` 等工具搭建自己的本地服务, 还可以尝试量化版本的模型. [模型列表](#) [模型量化介绍](#)

## 8. 提交要求

提交名为 `学号_姓名.zip` 的压缩文件, 文件组织示例:

请注意不要把原始数据集/模型放进压缩包, 以免导致文件过大

```
学号_姓名.zip
├── code/
│   ├── main.py
│   ├── README.md (简要版本即可)
│   ├── requirements.txt
│   └── results/
│       ├── outputs.json
│       └── outputs_improved.json (如有)
└── report.pdf
```

## 9. 评分标准

分数范围	描述
90-100%	<b>功能：</b> 系统功能完善； <b>分析：</b> 对普通RAG进行了深入的错误分析； <b>改进：</b> 设计了新方案并有清晰结果与分析； <b>代码与报告：</b> 提供了全面的分析和完善的代码, 报告详细清晰.
80-89%	<b>功能：</b> 系统工作正常, 有效实现了RAG技术 (EM>0.3, F1>0.4); <b>分析：</b> 评估结果可靠且有一定的分析深度； <b>改进：</b> 有一定的新方案尝试； <b>代码与报告：</b> 代码结构清晰, 报告说明充分.
70-79%	<b>功能：</b> 系统基本功能正常, 正确实现了标准评估指标, 但问题准确率较低； <b>分析：</b> 错误例子分析不足； <b>改进：</b> 新方案设计较为简陋； <b>代码与报告：</b> 代码和报告说明基本完整.
60-69%	<b>功能：</b> 系统可以运行但存在明显不足； <b>分析：</b> 完成了基本评估, 但分析简陋； <b>改进：</b> 几乎没有尝试新方案； <b>代码与报告：</b> 代码可运行, 报告较为简略.
低于 60%	<b>功能：</b> 系统关键组件缺失或无法正常工作； <b>分析：</b> 评估不完整； <b>改进：</b> 没有新方案； <b>代码与报告：</b> 报告不足以理解系统实现, 代码存在严重问题.

## 10. 截止日期

作业截止日期：2025-05-11 23:59:59

## 11. 提示和建议

- 可以尝试通过 Deep Research 类型的工具 (例如免费的 [Grok 镜像站](#), [Gemini](#) 和付费的 [ChatGPT](#) 等) 快速调研和了解 RAG 和相关的 Multi-hop QA 技术
- 若计划使用课程提供的 vLLM 服务, 请尽早开始, 以免 ddl 临近时遇到资源不足问题

## 12. 支持

如果遇到任何问题或有疑问, 请联系助教石雨凌 (从 QQ 群直接联系或者邮件 [yuling.shi@sjtu.edu.cn](mailto:yuling.shi@sjtu.edu.cn)).