本科毕业论文

TBD

学生：TBD

学号：TBD

院系：TBD

专业：TBD

班级：TBD

指导教师：TBD

2026 年 2 月 3 日

原创性声明

本人声明所提交的毕业论文为本人在导师指导下独立完成的成果，除文中已经注明引用的内容外，本论文不包含任何他人已发表或撰写的成果。

授权声明

本人同意学校保存、借阅本论文的印刷版和电子版，并允许论文被检索与引用。

中文摘要

本文面向复杂金融查询场景，围绕检索增强生成在事实可靠性、多步推理与数值计算上的不足展开研究。以 FinDER 数据集（5,703 个查询-证据-答案三元组）为载体，本文将金融问答任务建模为“检索证据—生成/计算答案”的流程，并构建 baseline → multistep → calculator 的模块化系统。方法上，统一数据准备与索引构建流程，定义检索与问答的评测口径（Recall@K、MRR@K、EM 与 numeric_em 等），并通过可复现的 run_id 与 outputs 结构记录实验产物。实验结果显示，检索器微调在 Full dev 上的 Recall@10 从 0.3246 提升至 0.3789，Complex 子集同样获得正向增益；多步检索对 Recall@10 影响较小但对 MRR@10 有轻微变化；引入计算器后可对数值问题给出可度量的正确性指标。本文最后讨论了单次随机种子与误差分析不足等局限，并给出后续改进方向。

关键词： FinDER；金融问答；RAG；多步检索；数值推理；可复现性

Abstract

This thesis addresses complex financial queries and studies retrieval-augmented generation (RAG) for factuality, multi-step reasoning, and numeric computation. Using the FinDER dataset (5,703 query – evidence – answer triples), we formulate the task as evidence retrieval followed by answer generation or calculation, and implement a modular pipeline: baseline → multi-step retrieval → calculator. We standardize data preparation and indexing, define evaluation metrics (Recall@K, MRR@K, EM, and numeric_em), and record reproducible outputs with run_id and structured artifacts. Experiments show that retriever fine-tuning improves Full dev Recall@10 from 0.3246 to 0.3789, with consistent gains on the Complex subset; multi-step retrieval keeps Recall@10 stable while slightly changing MRR@10; enabling the calculator yields measurable numeric correctness. The pipeline emphasizes traceability of evidence and outputs. Results are reported under a single-seed setting without variance estimates or detailed error breakdowns, motivating future work on stability analysis, error categorization, and robustness.

Keywords: FinDER; financial question answering; retrieval-augmented generation; multi-step retrieval; numeric reasoning; reproducibility

目录

第一章　绪论

第一章绪论

1.1　研究背景与动机

金融领域问答需要以可追溯的证据为基础来保证回答的可靠性，而检索增强生成（RAG）为此提供了“先检索后生成”的基本范式。复杂金融查询往往涉及跨文档或跨段落的信息组合，传统单步检索在此类问题上容易遗漏关键证据。为应对多跳检索与数值计算需求，本研究以多步检索与显式计算为核心动机展开系统化探索。

1.2　问题定义与任务边界

本文聚焦复杂金融查询的检索问答任务：给定金融领域查询与语料库，系统需要检索证据并给出可验证的答案，必要时执行数值计算。该定义强调多步检索与推理整合，是本文的任务边界与评测语境。FinDER 数据集由金融领域专家构建，包含 5,703 个查询-证据-答案三元组，是本文实验的主要数据基础与问题载体。

1.3　技术路线概述

本文采用 baseline → multistep retrieval → calculator 的模块化流水线，从数据准备、检索评估、单步基线到多步检索与数值计算逐步展开，形成可复现的实验链路。评测口径覆盖检索与问答两个层面：检索侧使用 Recall@K 与 MRR@K，问答侧使用 Exact Match/EM 与数值评测指标 numeric_em 及误差统计，以保证对检索与数值推理能力的可比性描述。

1.4　主要贡献

第一，提供覆盖 baseline、多步检索与数值计算的可复现流程骨架，并以配置驱动的脚本组织实验步骤，便于后续复现实验与扩展比较。第二，系统实现采用 data/indexing/retrieval/multistep

的模块化结构，便于清晰描述与分模块验证实验设置。第三，明确检索、QA 与数值 QA 的指标口径与实现，支撑论文实验部分的统一表述与对比。

1.5 论文结构安排

第二章将综述相关工作并界定本文与既有研究的关系；第三章描述方法框架与关键模块；第四章说明实验设置与评测口径；第五章报告实验结果并进行分析；第六章讨论局限性与可能改进；第七章给出总结与展望。

第二章　相关工作

第二章相关工作

2.1　金融问答与检索增强范式

金融问答场景强调回答的可追溯性与证据约束，检索增强生成（RAG）因此成为常见的系统范式。已有研究通常从“检索到证据、再生成答案”的流程出发，强调证据对齐与事实一致性，但具体实现与评测口径存在差异。[5]

金融领域的问答与事实核验任务进一步放大了领域术语、缩写歧义与时效性问题，研究常以金融文本或报告为语料构建数据集与评测基准。相关工作往往聚焦于实体消歧、证据选择与可信度约束等问题，强调“可解释证据链”的价值。[3]

从范式角度看，金融 QA 的研究脉络与通用 RAG 研究相互借鉴，但金融场景对证据可靠性与数值准确性要求更高。本文的工作定位于此类场景下的复杂查询，后续章节将以金融领域数据集与评测口径为基础展开讨论。[8]

2.2　多跳/多步检索与查询改写

多跳检索关注跨段落或跨文档的信息组合，通常通过分步检索或迭代改写查询来逐步补全证据链。相关研究提出多种检索—推理交替策略，以降低单次检索遗漏关键信息的风险。[1]

查询改写与迭代检索方法将检索过程视为序列决策问题：根据已检索到的证据生成下一轮检索线索，并在多轮迭代后汇总证据。该类工作强调“检索策略”的有效性与可控性，同时对评测口径一致性提出要求。[4]

多步检索与多跳推理常被用于复杂问题或组合型问题，其核心难点在于如何衡量多步检索带来的增益，以及如何避免候选截断对 Recall@K 的影响。本文在方法与实验设置中将明确这一口径约束，并配合候选数统计进行核验。[9]

2.3 工具增强与数值推理

工具增强（Tool-augmented）方法通过引入计算器、检索器或外部模块来处理模型难以稳定完成的计算与逻辑任务。相关研究强调“工具可控性与可验证性”，并将数值推理作为典型应用场景。[6]

数值推理研究通常关注从证据文本中抽取数值并执行算术操作，包括同比、差值与占比等基础计算。该类工作为评测“数值正确性”提供了方法论基础，但不同数据集与任务对数值精度的定义不尽一致。[2]

与纯生成式回答相比，工具增强路径倾向于将“计算”从生成模型中剥离出来，以减少算术错误与不可控推理。本文的计算器模块与数值评测口径将沿用该思路，但不预设性能结论。[7]

2.4 数据集与评测口径

数据集层面，FinDER 是面向金融领域的检索问答数据集，包含查询—证据—答案三元组，可用于评测检索与问答的协同能力。本文实验以 FinDER 为主要数据基础，并以该数据集的任务定义作为问题边界。

评测口径方面，检索侧常用 Recall@K 与 MRR@K 衡量召回与排序质量，问答侧常用 Exact Match 等指标评估答案一致性。数值 QA 任务则需要额外考虑 numeric_em 与误差统计，以反映数值计算的准确性与稳定性。

不同数据集与任务设置可能导致指标口径不一致，尤其在多步检索中需关注 top_k 与评测 k 的匹配关系。本文在实验设置中将严格对齐该口径，并给出候选数统计以保证公平性描述。

2.5 小结：本工作的定位与差异

本工作聚焦复杂金融查询的检索问答任务，采用 baseline → multistep → calculator 的模块化流水线组织实验流程，强调可复现与可追溯的证据链路，而不预设性能领先结论。

在相关工作谱系中，本文的方法定位于“多步检索与工具增强”交叉区域：以多步检索补全证据，以计算器模块处理数值推理，并使用统一评测口径进行比较。该定位与现有方法的主要差异在于流程化与可复现的工程组织方式，而非声称算法创新或显著性能领先。

第三章　方法

第三章方法

3.1　任务形式化与符号约定

本研究将原始数据统一为包含 qid、query、answer、evidences 的记录结构，通过 prepare_data 的 field_map 将原始列映射到统一字段，并将 evidences 解析为带 evidence_id 的结构化列表，保证检索与问答模块共享一致的输入格式。

在本文的接口约定中，检索模块产出“证据集合”，而答案模块产出最终答案：baseline 的 predictions.jsonl 记录 qid、pred_answer 与 used_chunks，多步检索的 retrieval_results.jsonl 记录 final_top_chunks 与 all_collected_chunks，计算器流程额外生成 predictions_calc.jsonl 等文件并保留检索结果以便回溯。

3.2　数据与索引构建

数据准备阶段由 prepare_data 负责：读取 dataset/finder_dataset.csv（FinDER CSV），按 train/dev/test 比例切分并写入 data/processed/*.jsonl，同时在 outputs/<run_id>/ 下落盘 data_stats.json 与 config.yaml，并记录随机种子以保证可复现性。

数据集总量与拆分规模已汇总为 docs/data_stats.json（含 train/dev/test 计数与 dev 复杂子集比例），可作为方法章中的数据规模证据。

语料与索引构建由 build_corpus 完成：从 data/processed 的 evidences 字段抽取证据文本，按 chunk_size=1000 与 overlap=100 进行分块，并写入 data/corpus/chunks.jsonl，chunk 的 meta 中包含 source_qid、evidence_id 与 chunk_id 等信息。

整体流水线按 README 中的脚本入口顺序组织，便于从数据处理到评测的逐步复现。

```
prepare_data(configs/prepare_data.yaml)
build_corpus(configs/build_corpus.yaml)
eval_retrieval(configs/eval_retrieval.yaml)
run_baseline(configs/run_baseline.yaml) -> outputs/<run_id>/predictions.jsonl
eval_qa(configs/eval_qa.yaml, predictions.jsonl, data/processed/dev.jsonl)
```

```
run_multistep_retrieval(configs/run_multistep.yaml) -> outputs/<run_id>/retrieval_res
eval_multistep_retrieval(configs/eval_multistep.yaml, retrieval_results.jsonl)
run_with_calculator(configs/run_with_calculator.yaml, optional multistep results)
eval_numeric(configs/eval_numeric.yaml, predictions_calc.jsonl)
```

3.3 单步检索 baseline

单步检索使用 HybridRetriever:同时维护 BM25 与 dense 向量索引,支持 bm25/dense/hybrid 三种模式，并用 alpha 融合归一化得分；当 use_faiss 启用但系统无 FAISS 时回退到 brute-force 计算。该 baseline 属于检索增强（RAG）流程，检索器与生成模块按接口分离。

baseline 运行时读取 dev 分片与 corpus，按 top_k 与 alpha 检索证据，输出 predictions.jsonl（qid、pred_answer、used_chunks）；pred_answer 采用模板式生成函数 placeholder_generate，从检索片段中截取文本构造答案。

baseline 的生成不依赖外部 LLM API，当前实现为模板式生成；最小复现分支（retrieval-only / full QA）已整理于 docs/repro_env_and_llm_dependency.md。

3.4 多步检索（multistep）

多步检索由 MultiStepRetriever 执行，配置项包含 max_steps、top_k_each_step、top_k_final、novelty_threshold 与 stop_no_new_steps，并支持 gate/refiner 等开关；这些配置来自 run_multistep.yaml 并在运行时写入 MultiStepConfig。

多步检索内部使用 StepPlanner 规划检索步骤，通过 gap 检测与 stop criteria 决定是否继续，并在必要时调用 refiner 生成下一步查询；最终通过 merge_strategy 聚合候选并在不足时回退补齐至 final_top_k。

StepPlanner 的 query_type 规则、gap 检测逻辑、停止条件（EMPTY_RESULTS / NO_GAP / MAX_STEPS / NO_NEW_EVIDENCE）与 query refiner 的改写规则已在 docs/multistep_design.md 中做可复述化整理。

多步检索输出 multistep_traces.jsonl 与 retrieval_results.jsonl，分别记录逐步检索轨迹与每个 query 的最终候选（final_top_chunks / all_collected_chunks、stop_reason、steps_used）。

3.5 证据整合与答案生成

baseline 的答案生成采用模板式占位策略：从检索到的证据片段中截取内容构造 pred_answer，并记录 used_chunks 以保持“答案—证据”可追溯关系。

在 calculator 流程中，若计算结果通过 gate 规则则基于计算结果生成 pred_answer；否则回退到检索片段与 placeholder_generate，并记录 fallback_reason 与 used_chunks，以保证失败路径同样可回溯。

3.6 数值计算器模块

run_with_calculator 支持两种输入：直接调用检索器或复用 multistep 的 retrieval_results；输出 retrieval_results.jsonl、facts.jsonl、results_R.jsonl、calc_traces.jsonl 与 predictions_calc.jsonl，为后续 numeric 评测提供输入与可解释轨迹。

计算器模块的任务类型（yoy/diff/share/multiple）、事实抽取规则与 gate 回退条件已整理于 docs/calculator_design.md，对应代码在 src/calculator 与 run_with_calculator.py 中可追溯。

数值评测由 eval_numeric 执行，按 precision 参数计算 numeric_em 与误差统计，逐条写入 numeric_per_query.jsonl，并汇总到 numeric_metrics.json 以供实验表格引用。

3.7 评测与日志/产物规范

检索评测通过 compute_retrieval_metrics 计算 Recall@K / MRR@K 等指标，eval_retrieval 写出 metrics.json 与 per_query_results.jsonl，为检索阶段提供统一评测口径。

问答评测使用 Exact Match (EM) 与 token_f1 作为核心指标，数值评测使用 numeric_em 与误差统计字段，并统一写入各自的 metrics.json / numeric_metrics.json 文件。

所有脚本使用 run_id 生成 outputs/<run_id>/ 目录并落盘 config.yaml、logs.txt 等文件；run_id 由 UTC 时间戳与短 UUID 组成，同时记录 git_commit 以支持复现。

第四章　实验与结果

第四章实验与结果

4.1　实验设置

本章实验基于 FinDER 数据集，采用 train/dev/test 划分（4562/570/571），并在 dev 上构建复杂子集（243 条）与数值子集；这些统计已汇总到 docs/data_stats.json。

检索评测使用 Recall@K 与 MRR@K，K 取 [1,5,10]；数值评测使用 numeric_em、rel_error_mean、coverage 等指标，口径由 eval_numeric 实现。

Step6 主结果使用统一的 run_experiment 入口，覆盖 baseline / multistep / calculator 组合；post_ft 相关实验使用 retriever_ft 产物 models/retriever_ft/latest。

4.2　主结果对比

在 Full dev 上，pre_ft_baseline 的 Recall@10 为 0.3246，MRR@10 为 0.2030；post_ft_baseline 的 Recall@10 为 0.3789，MRR@10 为 0.2554。

在 Complex 子集上，pre_ft_baseline 的 Recall@10 为 0.3457，MRR@10 为 0.2330；post_ft_baseline 的 Recall@10 为 0.3951，MRR@10 为 0.2961。

calculator 开启后，检索指标与 baseline 相同（retrieval_full/complex 来自同一检索流程），但 numeric 指标可从 numeric_dev 获得：numeric_em=0.3964，rel_error_mean=689.2285，coverage=0.6180。

4.3　组件贡献（baseline → multistep → calculator）

在 post_ft 设置下，multistep 使 Full/Complex 的 MRR@10 有轻微变化（例如 full_mrr10 由 0.2554 变为 0.2556），而 Recall@10 保持不变；该差异来自 multistep 的 merge 与 stop 策略对排序的影响。

相对 pre_ft_baseline 的 Δ 统计显示，post_ft 系列在 Full/Complex 的 Recall@10 与 MRR@10 上均为正增量；数值指标未计算 Δ（baseline 无 numeric 指标）。

4.4 诊断分析

Step6 使用 k_list=[1,5,10]，且 multistep 的 top_k_each_step=10、top_k_final 默认为 10，因此 Recall@10 不存在候选截断风险。

numeric 指标仅在 calculator 开启时产生，说明数值推理贡献主要来自 calculator pipeline；没有 calculator 的 run 其 numeric_dev 为空。

4.5 案例研究

成功案例（numeric）：在 run_id=20260130_014940_21aa62_m04 的 numeric_dev 中，qid=8c8c8c34 的 gold_num 与 pred_num 均为 202.0，numeric_em=1。

失败案例（numeric）：同一 run 中 qid=8b69ba09 的 pred_num 为空，numeric_em=0，表明抽取或计算失败导致数值评测未通过。

4.6 小结与局限

本章结果基于单次 run 与固定随机种子，未报告方差或显著性检验；baseline 生成不依赖外部 LLM，避免了外部 API 变动带来的不确定性，但也限制了生成端的上限。

第五章　结论

本文面向复杂金融查询场景，关注检索增强生成在事实准确性、多步推理与数值计算方面的不足，围绕 FinDER 数据集构建并验证了可复现的检索问答流水线。整体方案以单步检索 baseline 为起点，逐步引入多步检索与计算器模块，统一数据准备、索引构建、评测口径与产物记录，使实验过程和证据链条保持可追溯与可复现。同时，统一的 run_id 与 outputs 结构使得实验复现与论文引用形成闭环。

实验结果表明，检索器微调在 Full/Complex 子集上提升了 Recall@10 与 MRR@10（Full dev Recall@10 由 0.3246 提升至 0.3789，MRR@10 由 0.2030 提升至 0.2554）；在此基础上，多步检索对 Recall@10 基本保持不变但对 MRR@10 带来轻微变化。开启 calculator 后产生 numeric_em=0.3964、coverage=0.6180 等数值指标，同时检索指标与对应 baseline 保持一致，说明数值能力主要由计算器管线贡献。

主要贡献可以概括为以下四点：（1）完成 FinDER 数据准备与索引构建的统一流程，并给出可直接引用的数据统计结果，为实验设置与方法描述提供证据基础。（2）构建 baseline -> multistep -> calculator 的模块化检索问答流水线，明确各模块输入输出与产物契约，保证实验可复现。（3）将多步检索与计算器模块的内部规则整理为可复述设计文档，并与代码实现一一对应，便于后续复用与验证。（4）建立统一的评测口径与结果落盘规范，确保 Recall@K/MRR@K、numeric_em 等指标可追溯。

尽管取得了上述结果，本文仍存在一些局限：实验基于单次随机种子，尚未给出方差或置信区间；误差分析与数值错误分布尚不完善；运行成本与资源占用缺少系统统计。同时，生成端采用模板式策略，答案质量仍有提升空间。这些限制使结论主要反映当前设置下的观察结果。未来工作可在多随机种子稳定性评估、错误类型分析、成本评估与更强生成器融合等方面进一步完善。

参考文献

文献补齐说明：当前文献条目为占位符，后续将依据 phase3_citation_plan.md 补齐具体作者、标题与出处。

参考文献

[1] Placeholder. Placeholder title for multihop-retrieval-2022. 2022. PLACEHOLDER. Missing: author/title/venue. Action: fill from advisor-provided list. Source hint: cited in Chapter 2 (related work).

[2] Placeholder. Placeholder title for numeric-reasoning-2022. 2022. PLACEHOLDER. Missing: author/title/venue. Action: fill from advisor-provided list. Source hint: cited in Chapter 2 (related work).

[3] Placeholder. Placeholder title for financial-qa-2023. 2023. PLACEHOLDER. Missing: author/title/venue. Action: fill from advisor-provided list. Source hint: cited in Chapter 2 (related work).

[4] Placeholder. Placeholder title for query-reformulation-2023. 2023. PLACEHOLDER. Missing: author/title/venue. Action: fill from advisor-provided list. Source hint: cited in Chapter 2 (related work).

[5] Placeholder. Placeholder title for rag-survey-2023. 2023. PLACEHOLDER. Missing: author/title/venue. Action: fill from advisor-provided list. Source hint: cited in Chapter 2 (related work).

[6] Placeholder. Placeholder title for tool-augmented-rag-2023. 2023. PLACEHOLDER. Missing: author/title/venue. Action: fill from advisor-provided list. Source hint: cited in Chapter 2 (related work).

[7] Placeholder. Placeholder title for calculator-qa-2024. 2024. PLACEHOLDER. Missing: author/title/venue. Action: fill from advisor-provided list. Source hint: cited in Chapter 2 (related work).

[8] Placeholder. Placeholder title for financial-fact-verification-2024. 2024. PLACEHOLDER. Missing: author/title/venue. Action: fill from advisor-provided list. Source hint: cited in Chapter 2 (related work).

[9] Placeholder. Placeholder title for iterative-retrieval-2024. 2024. PLACEHOLDER. Missing: author/title/venue. Action: fill from advisor-provided list. Source hint: cited in Chapter 2 (related work).

附录

.1 复现命令清单

以下命令基于仓库脚本与配置文件，可用于复现实验流水线：

```
python scripts\prepare_data.py --config configs\prepare_data.yaml
python scripts\build_corpus.py --config configs\build_corpus.yaml
python scripts\eval_retrieval.py --config configs\eval_retrieval.yaml
python scripts\run_baseline.py --config configs\run_baseline.yaml
python scripts\eval_qa.py --config configs\eval_qa.yaml --predictions outputs/<run_id
python scripts\run_multistep_retrieval.py --config configs\run_multistep.yaml
python scripts\eval_multistep_retrieval.py --config configs\eval_multistep.yaml --res
python scripts\run_with_calculator.py --config configs\run_with_calculator.yaml
python scripts\eval_numeric.py --config configs\eval_numeric.yaml --predictions outpu
python scripts\run_experiment.py --config configs\step6_base.yaml --overrides ...
python scripts\make_tables.py --experiments configs\step6_experiments.yaml
```

.2 关键配置文件

核心配置文件包括：configs/prepare_data.yaml、configs/build_corpus.yaml、configs/eval_retrieval.yaml、configs/run_baseline.yaml、configs/eval_qa.yaml、configs/run_multistep.yaml、configs/eval_multistep.yaml、configs/run_with_calculator.yaml、configs/eval_numeric.yaml、configs/step6_base.yaml、configs/step6_experiments.yaml 等。

.3 outputs 结构说明

实验输出统一写入 outputs/<run_id>/，其中包含 config.yaml、metrics.json、logs.txt 等基础文件；检索与多步检索还会生成 retrieval_results.jsonl 与 multistep_traces.jsonl；数值评测生成 numeric_metrics.json 与 numeric_per_query.jsonl。

.4 环境与依赖

本文复现条件以 requirements.txt 所列依赖为准；项目当前未记录固定的 Python 版本与硬件信息，本文不将其作为复现前置条件。baseline 生成不依赖外部 LLM API，相关说明见 docs/repro_env_and_llm_dependency.md。

致谢

在本研究与论文撰写过程中，我要感谢导师在研究方向选择与问题定义上的指导与建议；感谢课题组同学在实验复现与问题讨论中的帮助；同时感谢相关开源社区与数据集维护者提供的工具与数据支持，使本研究能够在可复现的基础上完成。