irtm 20251221

final project presentation

team 1 -
B11902091 姚權維 / B12705037 陳品翔 /
R14722040 江郁文 / B11103049 林立潔 / B11705061 羅立宸

RAGify@IRTM 2025: Chatting with Your Documents via Retrieval-Augmented Generation

YEO GUAN WEI
Dept. of CSIE, NTU
Taipei, Taiwan
b11902091@csie.ntu.edu.tw

CHEN, PIN-HSIANG
Dept. of IM, NTU
Taipei, Taiwan
b12705037@ntu.edu.tw

YU-WEN CHIANG
Dept. of Accounting, NTU
Taipei, Taiwan
r14722040@ntu.edu.tw

LIN, LI-CHIEH
Dept. of History, NTU
Taipei, Taiwan
b11103049@ntu.edu.tw

LUO, LI-CHEN
Dept. of IM, NTU
Taipei, Taiwan
b11705061@ntu.edu.tw

Outline

- Motivation & Introduction
- Experimental Results & Analysis
- RAGify System Implementation
- Conclusion & Limitations

Outline

- **Motivation & Introduction**
- Experimental Results & Analysis
- RAGify System Implementation
- Conclusion & Limitations

Motivation

- As modern college students, we now have powerful tools like NotebookLM, which make reading papers and literature much easier.
- However, while the tool is powerful, **its internal mechanism is largely a black box**.

- So, we take this opportunity to study how it works under the hood.
- The key idea behind NotebookLM is **RAG**.
- Our goal is to:
  - Understand each component in the RAG pipeline;
  - [1] Analyze how different components affect retrieval performance;
  - [2] **Build RAGify, a NotebookLM-lite system !!!**

Note: [1] Experiments handled by the ***lab-group***, while [2] system implementation handled by the ***webapp-group***.

What's RAG ?

- RAG stands for **Retrieval-Augmented Generation**.
- Instead of generating answers purely from a LLM model's memory, RAG retrieves relevant documents first, then generates answers grounded in those documents. This helps (i) **reduce hallucination** and (ii) **improves reliability**.
- In short: **RAG = Retrieval + Generation**, making AI answers more accurate and trustworthy.

RAG Pipeline

1. **Indexing**

   Documents are split, processed, and indexed to enable fast and efficient search.

2. **R – Retrieval**

   Given a user query, the system retrieves the top-k most relevant passages.

3. **A – Augmentation**

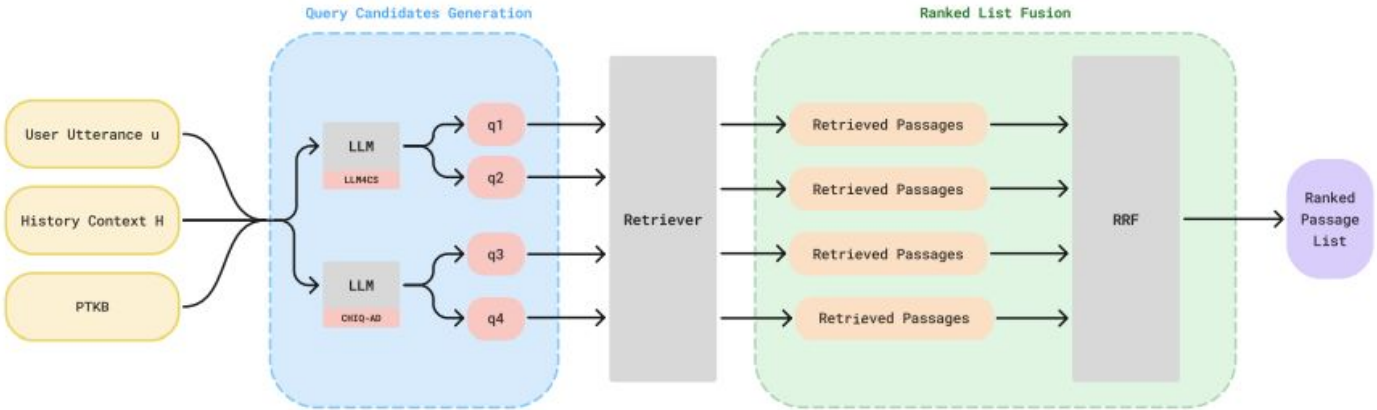
   The user query is rewritten to help the system find better passages.

4. **G – Generation**

   The model generates answers based on the retrieved passages.

System Flow

- Offline Phase: **Document** → Preprocessing → Indexing
- Online Phase: **User Query** → Query Rewriting → Passage Retrieval → Answer Generation



1) https://arxiv.org/abs/2509.15588v1

Outline

- Motivation & Introduction
- **Experimental Results & Analysis**
- RAGify System Implementation
- Conclusion & Limitations

Experimental Setup

- Datasets & Collections:
  - Dataset: **TREC CAsT 2022** – *Multi-turn, conversational, IR-friendly*
  - Document Collections:
    - **MS MARCO V2** – *QA-style, dense collection*
    - **KILT Wikipedia** – *open-domain knowledge*
- Evaluation Metrics (for retrieval quality):
  - MAP, nDCG@10, Recall@100, Precision@5
- Experiments (analyzing component effects on retrieval):
  - Indexing level: **Binary Retrieval**, **TF-IDF**, **BM25**
  - Query Rewriting level: **LLM4CS**, **CHIQ-AD**
  - LLM for rewritten queries: **meta-llama/Llama-3.1-8B-Instruct**, **Qwen/Qwen3-4B-Thinking-2507**

1) https://github.com/daltonj/treccastweb
2) https://github.com/grill-lab/trec-cast-tools

Overall Performance Comparison

- The table below shows the results across all the methods we attempted.

Table 1: Overall retrieval performance comparison on TREC CAsT 2022

Rewrite Method	Retrieval Model	MAP	nDCG@10	Recall@100	P@5
Human Rewrite	Binary Retrieval	0.0205	0.0204	0.1799	0.0066
	TF-IDF	0.0781	0.1029	0.3562	0.0275
	BM25	**0.0827**	**0.1098**	**0.3700**	**0.0363**
CHIQ-AD†,*	Binary Retrieval	0.0166	0.0196	0.1507	0.0066
	TF-IDF	0.0482	0.0612	0.2577	0.0191
	BM25	0.0535	0.0657	0.2671	0.0213
LLM4CS†,*	Binary Retrieval	0.0144	0.0197	0.1252	0.0066
	TF-IDF	0.0558	0.0681	0.2462	0.0220
	BM25	0.0606	0.0795	0.2650	0.0227
LLM4CS‡	Binary Retrieval	0.0245	0.0314	0.1255	0.0099
	TF-IDF	0.0679	0.0801	0.2875	0.0220
	BM25	0.0759	0.0872	0.2903	0.0253

† Based on Llama model; ‡ Based on Qwen model; * Results averaged over 3 runs

Key Finding 1: BM25 > TF-IDF >> Binary Retrieval

- Observation: BM25 consistently outperforms TF-IDF, which is much better than Binary Retrieval.

- Reason: Binary Retrieval ignores term frequency, treating every term occurrence equally. In contrast, BM25 applies term frequency saturation (diminishing returns for repeated terms) and document length normalization (penalizing longer documents). This makes BM25 more effective than simple TF-IDF.

Key Finding 2: LLM4CS Outperforms CHIQ-AD

- Observation: LLM4CS slightly outperforms CHIQ-AD.
- Reason: LLM4CS generates retrieval-friendly queries that work well with traditional keyword-based systems like BM25 and TF-IDF. In contrast, CHIQ-AD's queries sometimes include extra context that can dilute keyword matching.

- LLM4CS: Directly rewrites queries from conversation history using strategies like Chain-of-Thought and self-consistency.
- CHIQ-AD: Cleans and enhances conversation history (e.g., by adding pseudo responses) before generating the final queries.

1) https://arxiv.org/abs/2303.06573v2
2) https://arxiv.org/abs/2406.05013v2

What’s Query Rewriting?

Key Finding 3: “Thinking” Models Perform Better

- Observation: Qwen/Qwen3-4B-Thinking-2507 achieve the best answer quality.
- Reason: Reasoning models excel at advanced query rewriting because they can (i) interpret context and **infer the user’s intent** even when queries are ambiguous or incomplete, (ii) generate precise queries that **focus on the most relevant keywords and concepts**, and (iii) **handle multi-step logic** by combining information across conversation turns to produce coherent, retrieval-friendly queries.

Outline

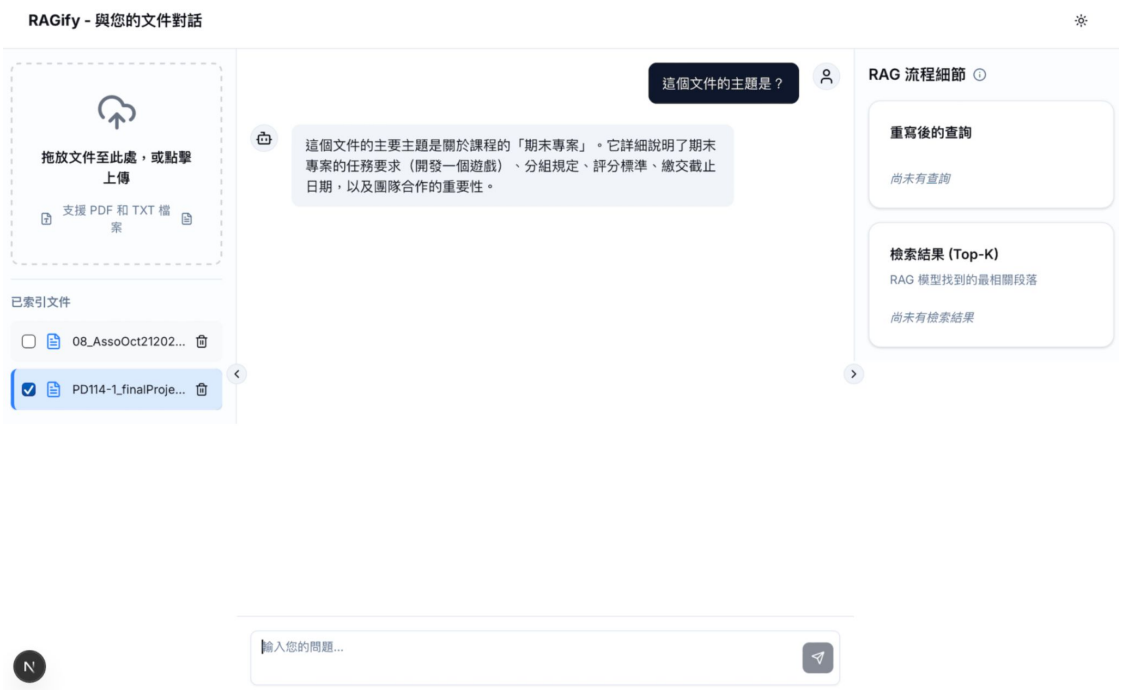
- Motivation & Introduction
- Experimental Results & Analysis
- **RAGify System Implementation**
- Conclusion & Limitations

RAGify System Implementation

- We implemented the answer generation framework from **group CFDA**, which won 1st place in the TREC iKAT 2025 Offline Generation-Only Track.

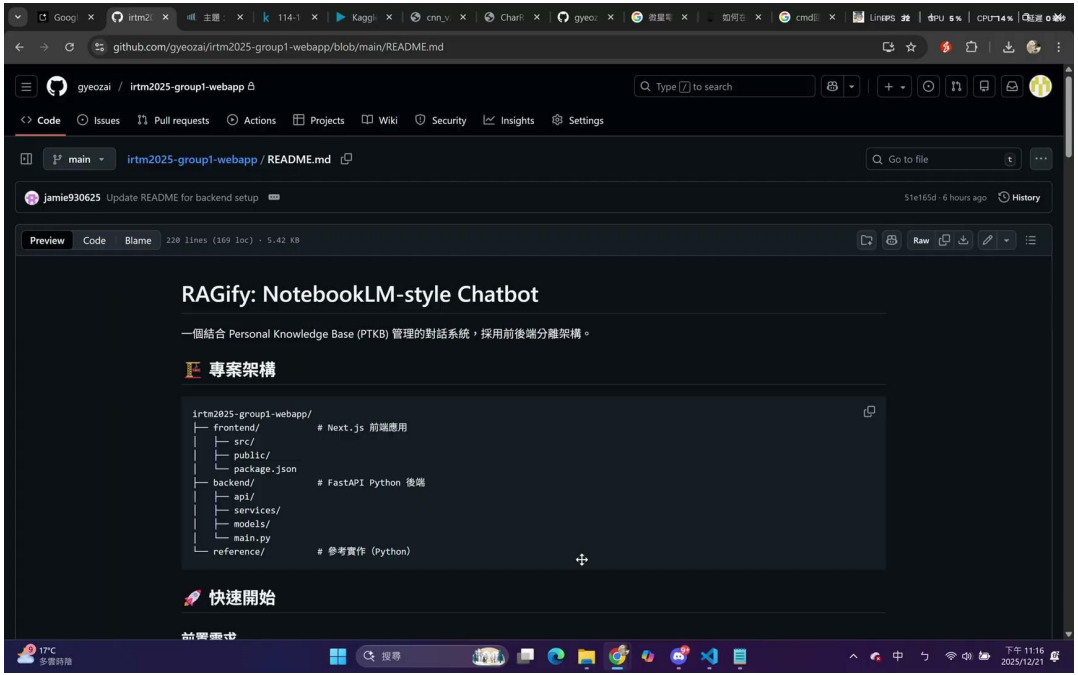
Group	Run ID	LLMeval		Nugget recall	BEM	F1
		SOLAR	GPT-4.1			
cfda	gen-only_npsg13_thru0_d4c5	**0.9333**	0.8000	**0.1195**	0.1641	**0.3136**
uva	nuggets-noptkb	0.9111	**0.8222**	0.1070	0.1721	0.3026
guidance	genonly_claritop10	0.7778	0.6889	0.1041	0.1650	0.2799
uva	nuggets-ptkb	0.8667	0.7778	0.1030	**0.1923**	0.3052
genaius	genaius-genonly-full-gpt4o	0.8889	0.7556	0.0999	0.1672	0.2827
cfda	gen-only_npsg20_thru03_d3c5	0.9111	0.7778	0.0978	0.1524	0.3065
genaius	genaius-genonly-summary-gpt4o	0.8667	0.7556	0.0811	0.1407	0.2750
usiir	usiir_run2	0.4000	0.2222	0.0510	0.1267	0.1877
usiir	usiir_run1	0.6000	0.3111	0.0508	0.1186	0.1916

1) https://arxiv.org/abs/2509.15588v1


RAGify - 與您的文件對話
拖放文件至此處，或點擊上傳
支援 PDF 和 TXT 檔案
已索引文件
08_AssoOct21202...
PD114-1_finalProje...
這個文件的主題是？
這個文件的主要主題是關於課程的「期末專案」。它詳細說明了期末專案的任務要求（開發一個遊戲）、分組規定、評分標準、繳交截止日期，以及團隊合作的重要性。
RAG 流程細節
重寫後的查詢
尚未有查詢
檢索結果 (Top-K)
RAG 模型找到的最相關段落
尚未有檢索結果
輸入您的問題...

RAGify System Demo

- Here is a demo video of our **RAGify** system.



Outline

- Motivation & Introduction
- Experimental Results & Analysis
- RAGify System Implementation
- **Conclusion & Limitations**

Conclusion

- We mainly contribute to:
  - Systematic analysis of the RAG pipeline and its components
  - Development of a NotebookLM-lite system —— **RAGify**
- We have tested the best configuration:
  - **BM25** for retrieval
  - **LLM4CS** for query rewriting
  - **Qwen/Qwen3-4B-Thinking-2507** for answer generation

- **Limitations:**
  - Limited resources prevented the use of dense indexing methods (e.g., SPLADE)
  - Did not compare query rewriting methods beyond prompt-based approaches

Thank You