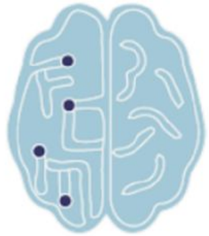
LIGHTRAG

Simple And Fast Retrieval-Augmented Generation



FOM

Focus On Data Mining

INDEX

01 Introduction

LLM - Large Language Model

- Nature Language Generative AI
  - Two types
    - Closed Source
      - EX)
        - OpenAI - GPT Series (Recently, GPT o1 - preview)
        - Anthropic - Claude Series (Recently, Claude 3.5 Sonnet)
        - Etc,...
      - 대기업 활용 LLM, 클라우드 기반 LLM -> **관련 어플리케이션 및 API**를 통해서만 활용 가능
      - 최고 수준의 성능
      - 모델 구조(코드) 및 데이터 **미공개**
    - Open Source
      - EX)
        - Llama Series (Recently, Llama-3.1-50B)
        - GLM-4
        - QWEN
        - Etc,...
      - 연구용 LLM, 사용자 기반 성장 (커스터마이즈) -> 개인, 연구기관, 스타트업이 주로 활용 **& 소스 코드**를 가져와서 활용 가능
      - 중간 수준의 성능 -> **Fine tuning을 통해 성능 향상 가능**
      - 모델 구조(코드) 및 데이터 **공개**

01 Introduction

RAG - Retrieval Augmented Generation

- RAG - Retrieval Augmented Generation
  - Generating 단계에서 LLM의 성능을 높이기 위한 방법론
  - **외부 데이터를 활용**하여 LLM의 input을 보충 및 도메인 지식 제공을 통해 성능 향상시킴
  - 기존 RAG의 한계점
    - Query를 Vector DB에 저장 후 **단순한 Cosine Similarity**의 top-k texts만 활용
    - 단편적인 text chunks만 활용
  - 기존의 간단한 Vector DB와 Cosine Sim만을 활용한 RAG보다 빠르고 더 정확하고 연관성을 잘 파악할 수 있는 방법론을 해당 논문에서 제시함
    - **Graph와 dual-level retrieval**가 주를 이루는 방법론을 제시

02 Related Works

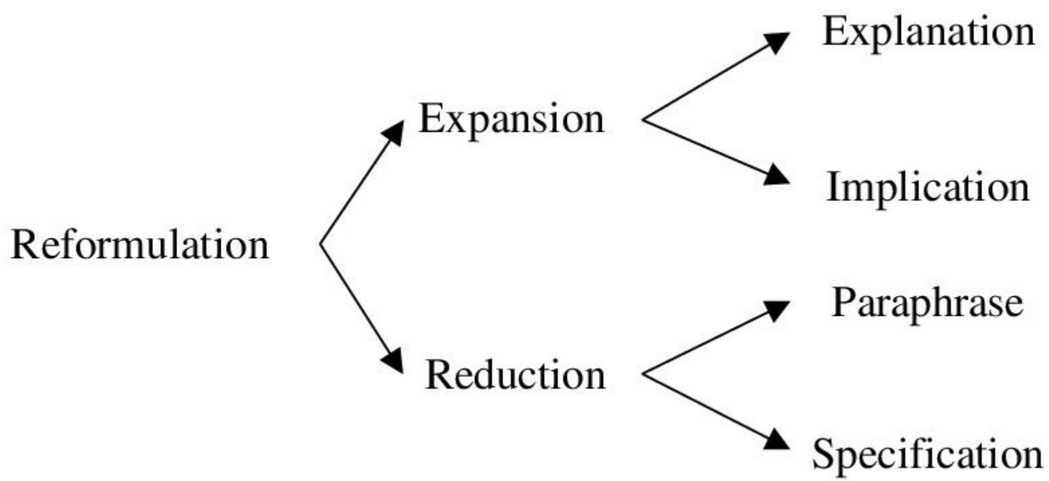
LLMs for Graphs

- Graph for Data Structure
  - **복잡한 관계를 가지는 데이터 구조**에는 그래프 구조가 강력한 힘을 가짐
  - LLM과 Graph를 결합하여 LLM의 성능을 향상 시키는 방법들을 활발히 연구 중
    - GNNs for Prefix (Graph → LLM)
      - GNN를 초기 처리 layer로 적용
      - Graph data를 구조 인식 token으로 변환하여 LLM의 추론에 활용
    - LLMs for Prefix (LLM → Graph)
      - Graph data를 text data로 강화하여 LLM이 node embedding이나 label 생성
      - 이를 통해 GNN 훈련 과정을 개선
    - **LLMs - Graphs Integration** (LLM + Graph)
      - LLM과 Graph data의 상호작용을 통해 성능 향상

02 Related Works

Reformulating and Sub-query Generation

- User query reformulating & Sub query
  - 사용자의 질문의 의도를 보다 더 잘 파악하기 위해 **재구성 및 서브 쿼리 생성**
    - Query reformulation은 특히 IR과 QA 부분에서 좋은 성능 향상을 보였음
      - 주로, Multi-step decomposition, Dynamic Knowledge Structure 등에 활용 됨
    - GenCRF, RQ-RAG, Hybrid RAG 등 다양한 논문에 언급 및 활용됨



03 Proposed Methods

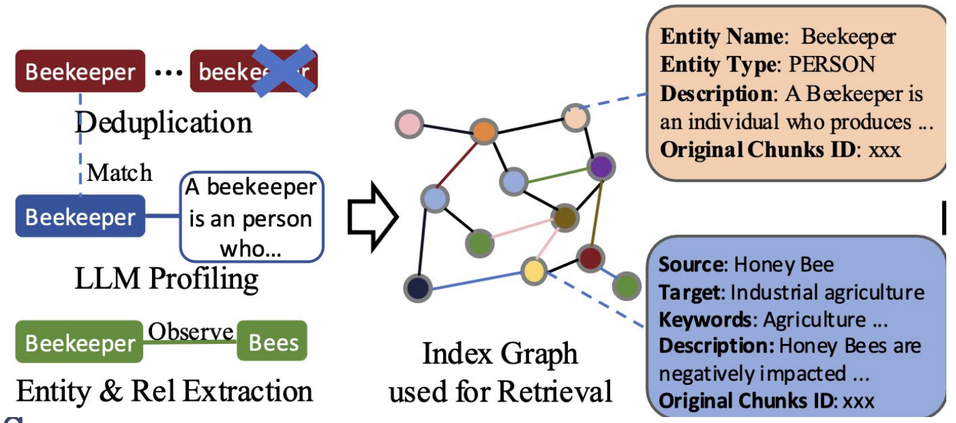
Data preprocessing mechanism

- Data preprocessing

$$\hat{\mathcal{D}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}}) = \text{Dedupe} \circ \text{Prof}(\mathcal{V}, \mathcal{E}), \quad \mathcal{V}, \mathcal{E} = \cup_{\mathcal{D}_i \in \mathcal{D}} \text{Recog}(\mathcal{D}_i)$$

D_i : Raw documents
Recog(~); R(~) : Extracting entity & relationship
Prof(~); P(~) : Profiling entity & relationship into Key-value structure
Dedupe(~); D(~) : Deduplicating Key-value structure

  - **Documents preprocessing**
  - **Chunking**
    - 1200 단위로 documents chunking
  - **Entity & Relationship extracting** → R(~)
    - Chunking된 texts에서 Entity와 Relationship을 추출 ← LLM을 통해 추출 및 생성
  - **Group structure** → P(~)
    - Node : Entity → 주체 (주인공 느낌)
    - Edge : Relationship → entities끼리의 관계
  - **Deduplicating** → D(~)
    - Node & Edge에서 중복되는 내용을 제거
  - **Key - Value Indexing**
    - Entity와 Relationship을 Key-Value Structure로 변환하여 Indexing
      - 이를 통해 Retrieval 속도를 향상 시킴



03 Proposed Methods

Query processing mechanism

- Query processing
  - User의 prompt(question) preprocessing
  - LLM을 통해 **Local & Global Key-words 추출 및 구조화**
    - Local Key-words : Prompt에서의 Entity
    - Global Key-words : Prompt에서의 Relationship
  - 위의 방법으로 추출된 Key-Value는 Graph Structure인 Vector DB와 결합되어 Retrieval & Generating Phase에서 활용됨

03 Proposed Methods

Information Retrieval

- Dual - level Retrieval
  - **Low** - level retrieval
    - **Specific** entity & relationship retrieval
    - Graph내에서 연관된 **특정한 entity**에 대해서 자세히 탐색
    - Graph에서 직접 연결된 부분들만 참고하는 **one-hop retrieval**
  - **High** - level retrieval
    - Focus on **topic** and **context**
    - Graph내에서 연관된 **넓은 범위의 entities**과의 관계를 탐색
    - Graph에서 직접 연결된 부분이 아닌 다른 node & edge까지 참고하는 **multi-hop retrieval** 진행
      - **Sub graph**까지 깊게 검색 → **Broader & Inter-dependent** information retrieval 가능

03 Proposed Methods

Graph & Vector Structure Advantages

- Integrating Graph and Vectors for Efficient Retrieval
  - Query Keyword Extraction
  - Keyword matching
    - 위의 두개의 Keyword 부분은 preprocessing에서 언급 한 바와 같이 Local & Global Key word를 추출함
  - **Incorporating High-Order Relatedness**
    - Graph structure이기 때문에, 집중하고 싶은 node(entity)와 인접 노드 탐색으로 고차 관계까지 확장할 수 있음

$$\{v_i | v_i \in \mathcal{V} \wedge (v_i \in \mathcal{N}_v \vee v_i \in \mathcal{N}_e)\}$$

dge까지 documents의 key와 intersection 후 활용하는 수식

(node & edge의 이웃 (neighborhood)의 union) 기존 documents의 Key와의 intersection

  - 이를 통해 검색에 불필요한 데이터 탐색을 제거
  - Vector와 graph의 결합으로 정확하고 빠른 관계 파악 가능

03 Proposed Methods

Response Generation

- LLM Response Generation
  - Low & High level Retrieval을 통해 검색된 entity, relationship, text chunk를 통합하여 활용
  - 위의 data를 모두 **통합 & 요약하여 LLM에게 제공**
  - LLM의 output을 user's needs에 맞춰 output formatting
    - user's needs에 맞춰서 세부 내용 요약 및 추가 정보 제공 등 가능

03 Proposed Methods

Fast & quick adaptation for new dataset

- LightRAG
  - LightRAG의 graph structure & data preprocessing (**Profiling** & **Deduplicate**)등의 과정을 새로운 documents에 적용시 다른 방법론보다 더 빠르고 효율적으로 **dynamic update** 가능
  - 새로운 documents에 대해서 똑같은 preprocessing 적용 시 기존 documents에서 얻은 graph를 재활용
    - 새로운 documents에 적용시킨 preprocessing 데이터를 **기존의 graph와 비교**하여 **중복 제거 처리** 등 진행
    - 모든 graph를 update하지 않아도 일부의 data만 graph에 추가하여 dynamic하게 새로운 데이터 추가할 수 있음

03 Proposed Methods

Complexity analysis of the LightRAG

- Computing Complexity Analysis
  - Two main parts
    - **The graph-based Index phase**
      - Using LLM to **extract entities & relationships** from each chunks
      - Need to use **LLM** for $\frac{\text{total tokens}}{\text{chunk size}}$ times
        - Ex. total tokens = 100,000, chunk size = 200 → 500 times calling
    - **The process involves the graph-based retrieval phase**
      - First, Need to utilize LLM to generate relevant key words → Like original RAG relied on vector-based search
      - However, LightRAG only need to **relying on retrieving entities and relationships** → Use less computing power & low complexity

03 Experiments

Performance comparison

- Performance Measure
  - Comprehensiveness - 응답 포괄성
  - Diversity - 응답 다양성
  - Empowerment - 사용자 이해도
  - Overall - 전반적 성과

- Experiment Dataset - UltraDomain Benchmark
  - Agriculture - 농업 관련 주제
  - CS - 컴퓨터 과학 및 머신러닝
  - Legal - 법률 및 규제 자료
  - Mix - 다양한 문학 및 철학 텍스트

- Comparison models
  - NaiveRAG, RQ-RAG, HyDE, GraphRAG

03 Experiments

Performance comparison

- LightRAG Performance
  - Comprehensiveness : **67%** 더 높은 확률로 포괄적인 응답 제공
  - Diversity : **최대 20%** 높은 확률로 다양한 응답 제공
  - Empowerment : 사용자에게 더 나은 정보 활용 능력 제공
  - Overall : 모든 데이터셋에서 기존 모델 대비 우위에 있는 능력 제공

Table 1: Win rates (%) of baselines v.s. LightRAG across four datasets and four evaluation dimensions.

	Agriculture		**CS**		**Legal**		**Mix**	
	NaiveRAG	**LightRAG**	NaiveRAG	**LightRAG**	NaiveRAG	**LightRAG**	NaiveRAG	**LightRAG**
Comprehensiveness	32.4%	67.6%	38.4%	61.6%	16.4%	83.6%	38.8%	61.2%
Diversity	23.6%	76.4%	38.0%	62.0%	13.6%	86.4%	32.4%	67.6%
Empowerment	32.4%	67.6%	38.8%	61.2%	16.4%	83.6%	42.8%	57.2%
Overall	32.4%	67.6%	38.8%	61.2%	15.2%	84.8%	40.0%	60.0%
	RQ-RAG	**LightRAG**	RQ-RAG	**LightRAG**	RQ-RAG	**LightRAG**	RQ-RAG	**LightRAG**
Comprehensiveness	31.6%	68.4%	38.8%	61.2%	15.2%	84.8%	39.2%	60.8%
Diversity	29.2%	70.8%	39.2%	60.8%	11.6%	88.4%	30.8%	69.2%
Empowerment	31.6%	68.4%	36.4%	63.6%	15.2%	84.8%	42.4%	57.6%
Overall	32.4%	67.6%	38.0%	62.0%	14.4%	85.6%	40.0%	60.0%
	HyDE	**LightRAG**	HyDE	**LightRAG**	HyDE	**LightRAG**	HyDE	**LightRAG**
Comprehensiveness	26.0%	74.0%	41.6%	58.4%	26.8%	73.2%	40.4%	59.6%
Diversity	24.0%	76.0%	38.8%	61.2%	20.0%	80.0%	32.4%	67.6%
Empowerment	25.2%	74.8%	40.8%	59.2%	26.0%	74.0%	46.0%	54.0%
Overall	24.8%	75.2%	41.6%	58.4%	26.4%	73.6%	42.4%	57.6%
	GraphRAG	**LightRAG**	GraphRAG	**LightRAG**	GraphRAG	**LightRAG**	GraphRAG	**LightRAG**
Comprehensiveness	45.6%	54.4%	48.4%	51.6%	48.4%	51.6%	50.4%	49.6%
Diversity	22.8%	77.2%	40.8%	59.2%	26.4%	73.6%	36.0%	64.0%
Empowerment	41.2%	58.8%	45.2%	54.8%	43.6%	56.4%	50.8%	49.2%
Overall	45.2%	54.8%	48.0%	52.0%	47.2%	52.8%	50.4%	49.6%

04 Conclusion

Graph & Vector Structure And entity & Relationship preprocessing

- Graph & Vector Structure
  - **Graph** Structure Database → Fast & quick retrieval
  - Incorporating High-Order Relatedness → More specific & boarder retrieval
  - **Dual-Level Retrieval** (Low / High) → More specific & boarder retrieval
- Entity & Relationship preprocessing
  - Fast & quick retrieval
  - Easy to understand **complex data interactions** (**Inter-dependency data**)
  - Easy to detect **High-Order Relatedness** relationship
  - Performance improvement by **Local & Global Keyword** matching

Q & A

감사합니다