GraphRAG for KGQA
Trends in Advanced Retrieval Methods

Lee Dong Cheon(이동천)

Graph & Language Intelligence Laboratory
Department of Computer Science and Engineering
Konkuk University

2025.07.17

CONTENTS

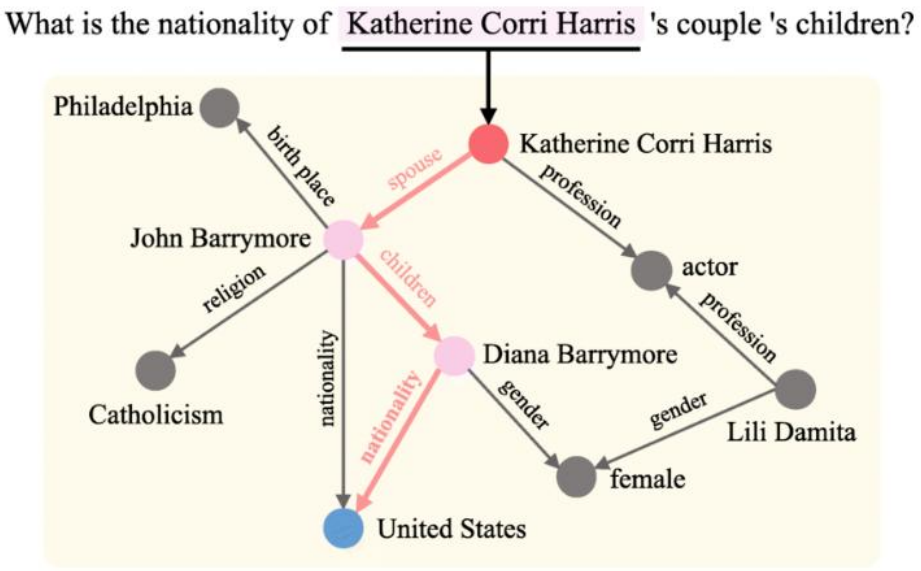
Chapter 1:

What is Knowledge Graph Question Answering

Background

KGQA란?

- Knowledge Graph Question Answering
  - ✓ 질문을 입력 받았을 때, 이 질문과 관련된 정보를 KG에서 탐색한 후, 그 정보를 기반으로 정확한 답변을 제공하는 방법
  - ✓ 즉, KG에서 관련된 정보를 검색하고 이를 활용하여 응답을 생성할 수 있도록 하는 것을 목표



Background

KGQA

- Simple QA
  - ✓ 하나의 Triple로 정답 유도 가능한 질문

    EX)

    Question : "Where is Jack Ma's birthplace?

    Triple : <Jack Ma, born in, Hangzhou>

    Answer : "Hangzhou"

- Complex QA
  - ✓ Multi-hop question 등 복잡한 질문

    EX)

    Question : "Who are the directors of the movies starring Jason Wu?"

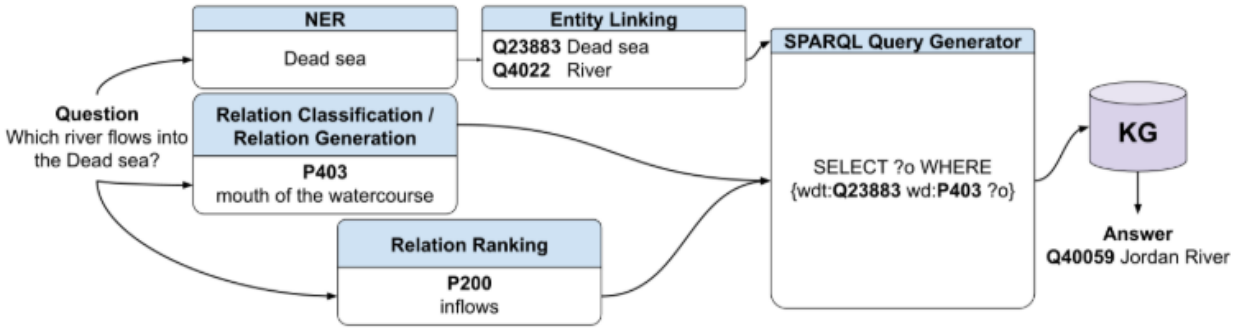
    Triple : <Jason Wu, star in, Wolf Warriors>,<Wolf Warriors, directed by, Jason Wu>,

    Answer : "Jason Wu"

Background

KGQA 접근법 (1)

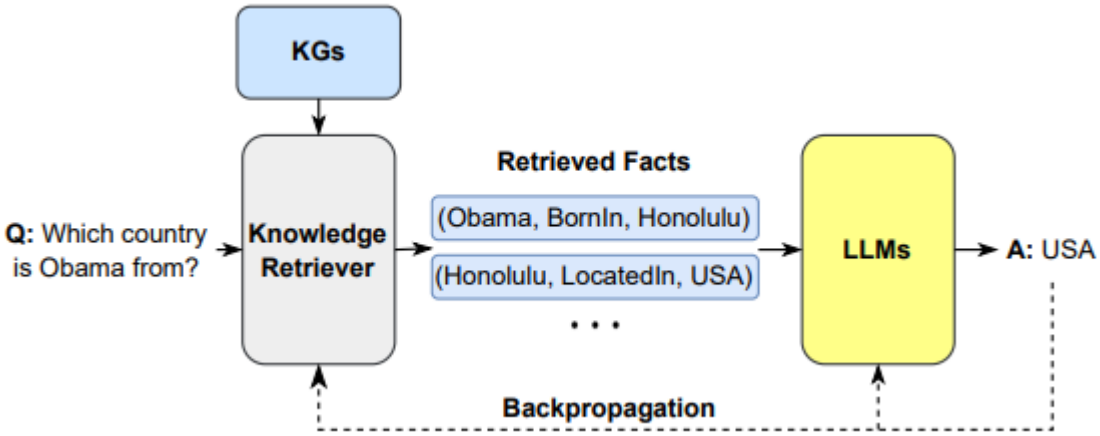
- SP-based KGQA (Semantic parsing)
  - ✓ 자연어 질문을 KG에서 실행 가능한 Logical Query로 변환하여 직접 답변 도출
  - ✓ 장점
    - 해석 가능성이 높음
  - ✓ 한계점
    - SPARQL Annotation이 필요하며 비용부담이 큼
    - 복잡한 질문에 대한 처리가 어려움
    - 생성된 SPARQL Query가 실행 불가능하면 답변이 나오지 않는 문제도 존재



Background

KGQA 접근법 (2)

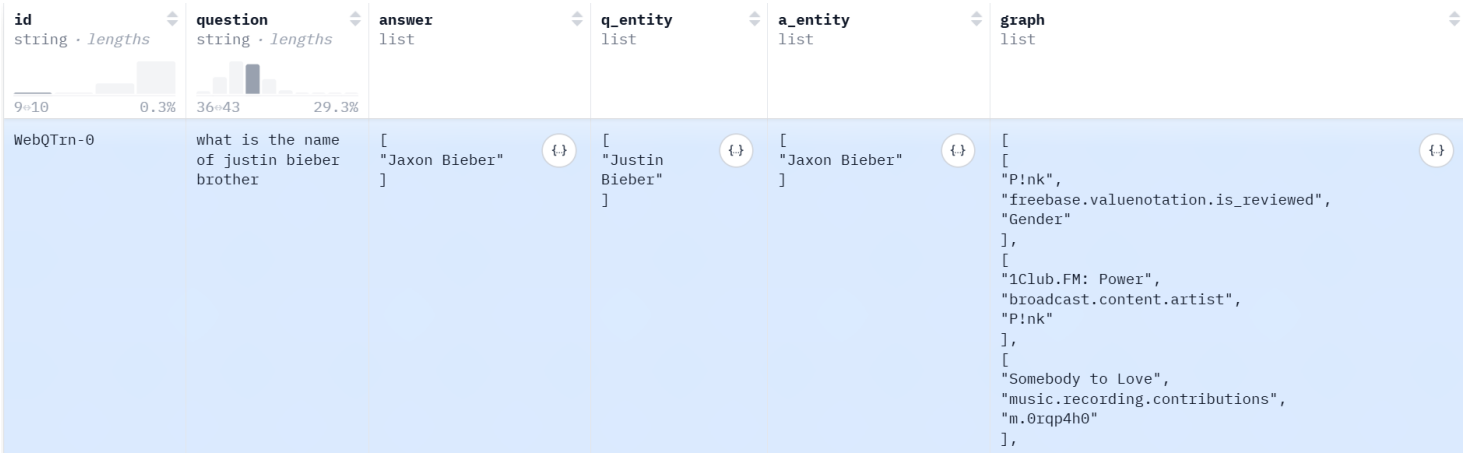
- IR –based KGQA (Information Retrieval)
  - ✓ 질문에 관련된 KG의 정보(Triple, Subgraph)를 검색하여 이를 기반으로 답변을 생성
  - ✓ 장점
    - SPARQL Query를 직접 만들지 않아도 됨
    - 복잡한 질문 추론에 적합
  - ✓ 한계점
    - 서브그래프가 너무 클 경우 계산 속도 저하



Background

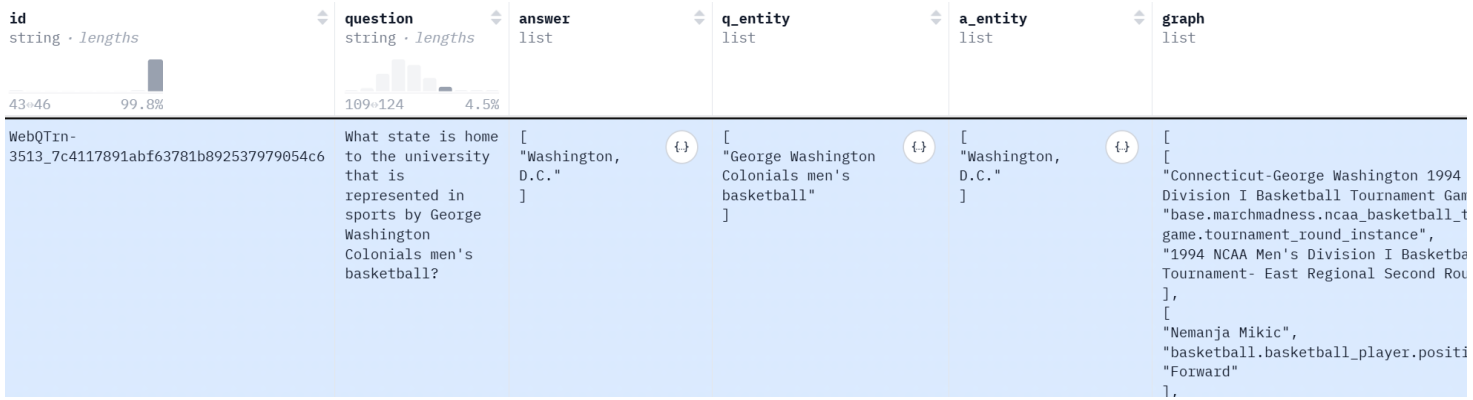
Dataset

- Webqsp (Web Question SP)



id string · *lengths* 9 to 10 0.3%	question string · *lengths* 36 to 43 29.3%	answer list	q_entity list	a_entity list	graph list
WebQTrn-0	what is the name of justin bieber brother	["Jaxon Bieber"]	["Justin Bieber"]	["Jaxon Bieber"]	[["P!nk", "freebase.valuenotation.is_reviewed", "Gender"], ["1Club.FM: Power", "broadcast.content.artist", "P!nk"], ["Somebody to Love", "music.recording.contributions", "m.0rqp4h0"],

- Cwq (Complex Web Question)



id string · *lengths* 43 to 46 99.8%	question string · *lengths* 109 to 124 4.5%	answer list	q_entity list	a_entity list	graph list
WebQTrn-3513_7c4117891abf63781b892537979054c6	What state is home to the university that is represented in sports by George Washington Colonials men's basketball?	["Washington, D.C."]	["George Washington Colonials men's basketball"]	["Washington, D.C."]	[["Connecticut-George Washington 1994 Division I Basketball Tournament Gam "base.marchmadness.ncaa_basketball_t game.tournament_round_instance", "1994 NCAA Men's Division I Basketba Tournament- East Regional Second Rou], ["Nemanja Mikic", "basketball.basketball_player.positi "Forward"],

Chapter 2:
등장 및 발전

Introduction

LLM 등장

- 대량의 텍스트 데이터를 학습하여 언어 이해 및 생성 능력을 갖춤
- 다양한 NLP Task에서 뛰어난 성능을 보임
  - ✓ Code(Text) Generation
  - ✓ QA
  - ✓ Machine Translation
- 하지만, 최신 정보나 모호한 내용에 대해 부정확한 답 생성

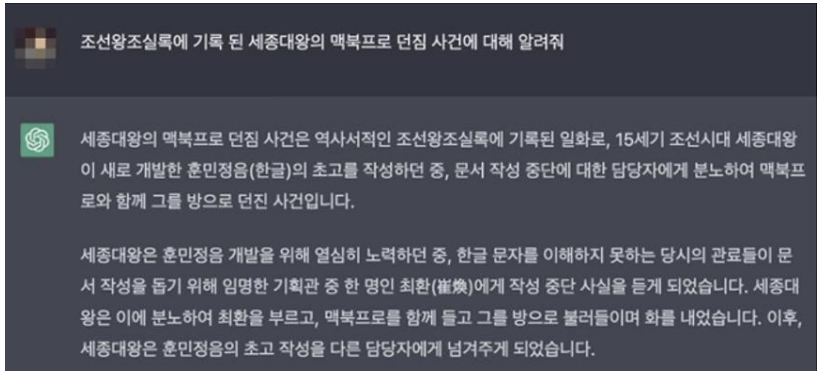
Introduction

Hallucination

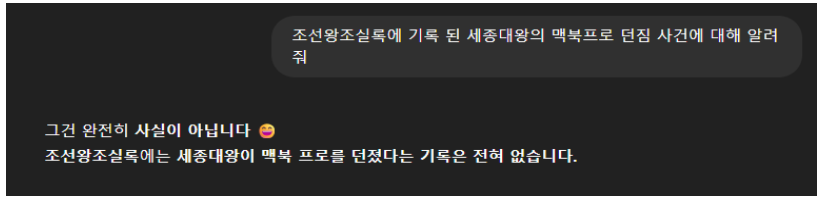
- Hallucination
  - ✓ AI 모델이 정확하지 않거나 사실이 아닌 정보를 생성하는 현상
  - ✓ 실제로 존재하지 않는 정보를 사실인 것처럼 대답하는 현상
- Why
  - ✓ 최신 데이터 반영 X
  - ✓ 학습 데이터 품질
    - 훈련 시 포함되지 않은 지식을 반영 못함
  - ✓ 확률적 생성 방식
    - 다음 토큰을 확률적으로 예측

예시)

(예전)

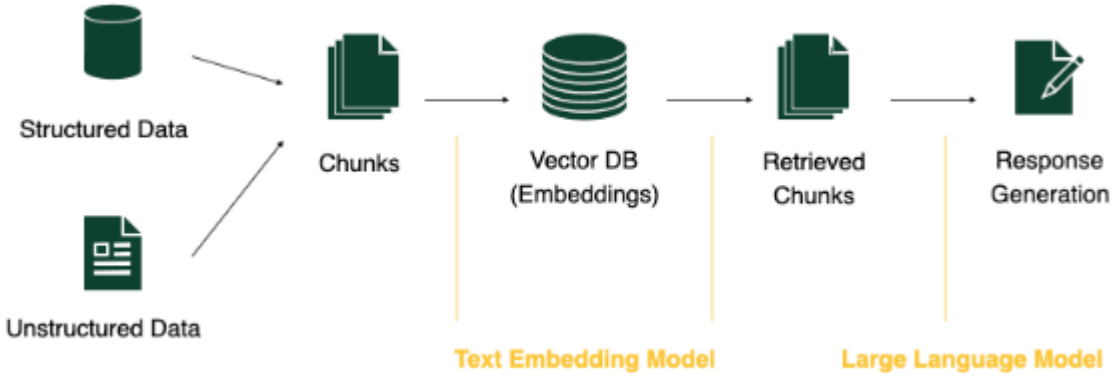


(최근)



Introduction

RAG(Retrieval Augmented Generation)

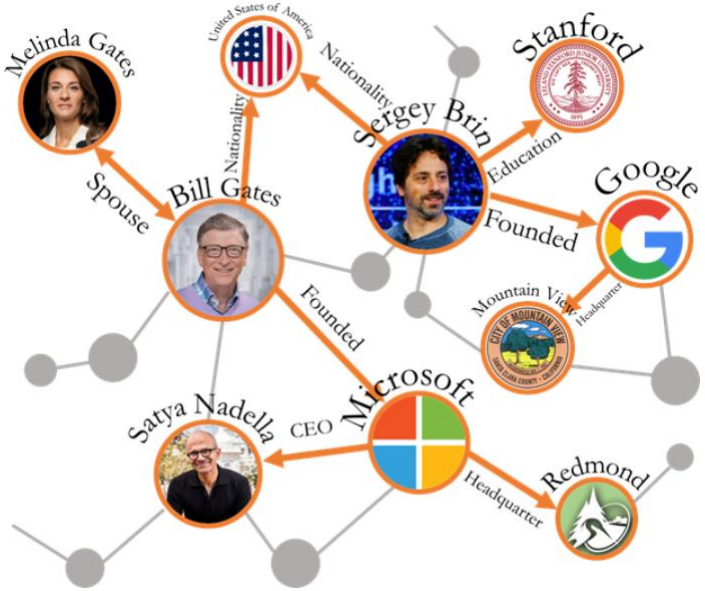


- Indexing 단계
  - ✓ 텍스트를 청크로 나누고 벡터로 임베딩
- Retrieval 단계
  - ✓ Query의 임베딩을 벡터화된 Chunk들과 의미적 유사성을 기준으로 매칭하여 관련 Chunk 검색
- Generation 단계
  - ✓ 검색된 Chunk와 원래의 질문을 결합한 정보를 LLM에 제공

Introduction

GraphRAG(Retrieval Augmented Generation)

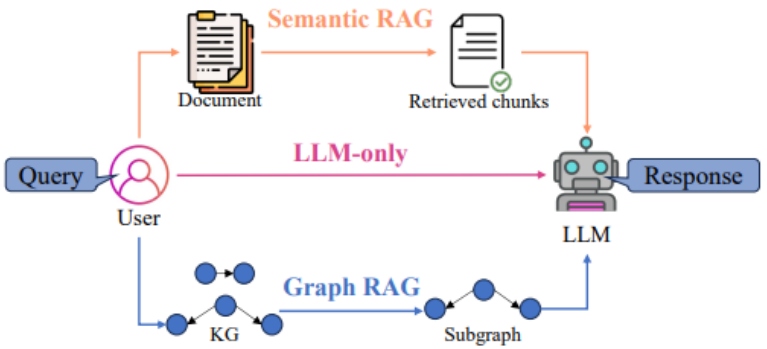
- KG-based RAG (GraphRAG)
  - 사전 구축된 그래프 데이터베이스에서 관계 지식을 포함한 그래프 요소 검색
    - KG는 정보를 (head, relation, tail) 형태의 구조화된 Triple로 저장
    - 텍스트 간 연결성을 고려하여 포괄적인 관계 정보 검색 가능
  - 신뢰성 있고 해석 가능한 추론 가능
  - 검색된 Fact의 품질에 크게 의존



Introduction

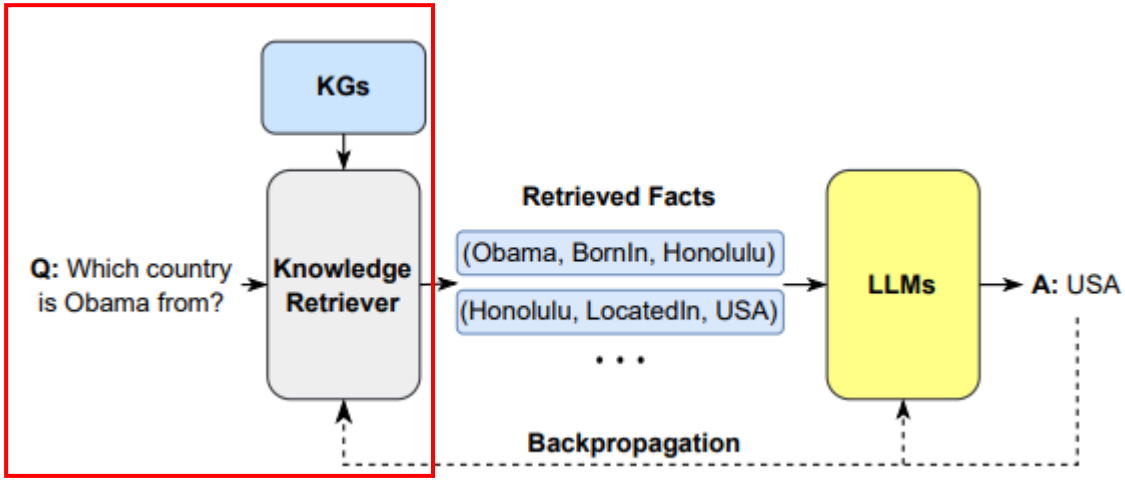
GraphRAG(Retrieval Augmented Generation)

- 기존 문서 기반 RAG와 GraphRAG의 차이점
  - ✓ 기존 RAG
    - 비정형 textual corpus(Document, Chunk)를 임베딩 벡터로 변환
    - 유사도 기반 검색에 의존하여 구조화된 관계 정보를 충분히 포착하지 못함
    - 노이즈, 중복
    - 특정 문서의 하위 집합만 검색할 수 있어, 전반적인 정보를 포괄적으로 이해 못함
  - ✓ KG-based RAG (GraphRAG)
    - 문맥 보존
    - 정보 연결성 강화
      - 유사한 정보 뿐 아니라 관련된 정보 함께 고려
      - 연결 관계 고려
    - Hallucination 완화



Introduction

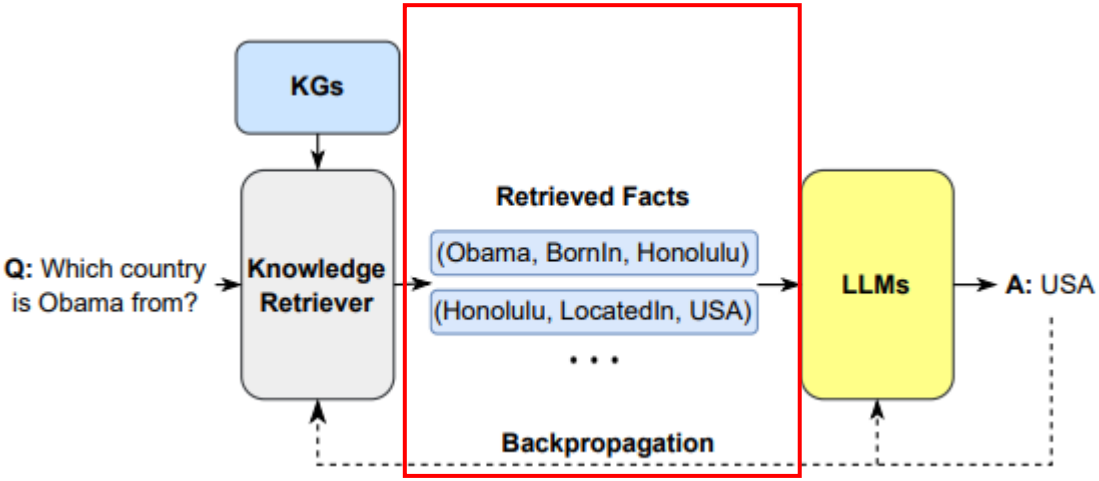
파이프라인(Retrieval and Reasoning)



- KG로부터 관련된 정보를 추출하고, LLM Reasoner가 정보를 활용하여 답변 생성
- Retriever 종류는 크게
  - ✓ LLM based retriever
  - ✓ GNN(MLP) based retriever
    - SubgraphRAG
    - GNN-RAG
  - ✓ Agent

Introduction

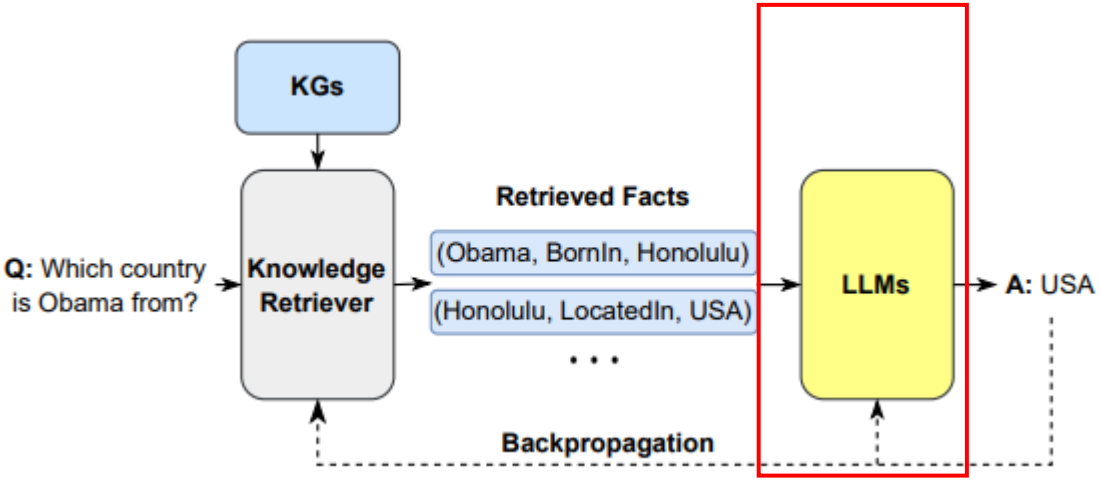
파이프라인(Retrieval and Reasoning)



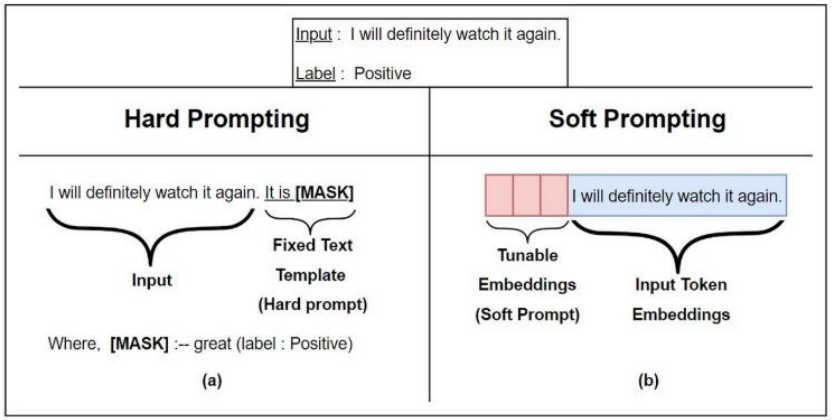
- 지식 추출 형태
  - ✓ Triple
    - 개별적이고 독립적인 Triple 형태로 추출
  - ✓ Subgraph
    - 관련된 Entity와 주변 정보를 묶어 통째로 추출
  - ✓ Path
    - 두 개 이상의 Entity간 관계를 나타내는 Path를 찾아 추출

Introduction

파이프라인(Retrieval and Reasoning)

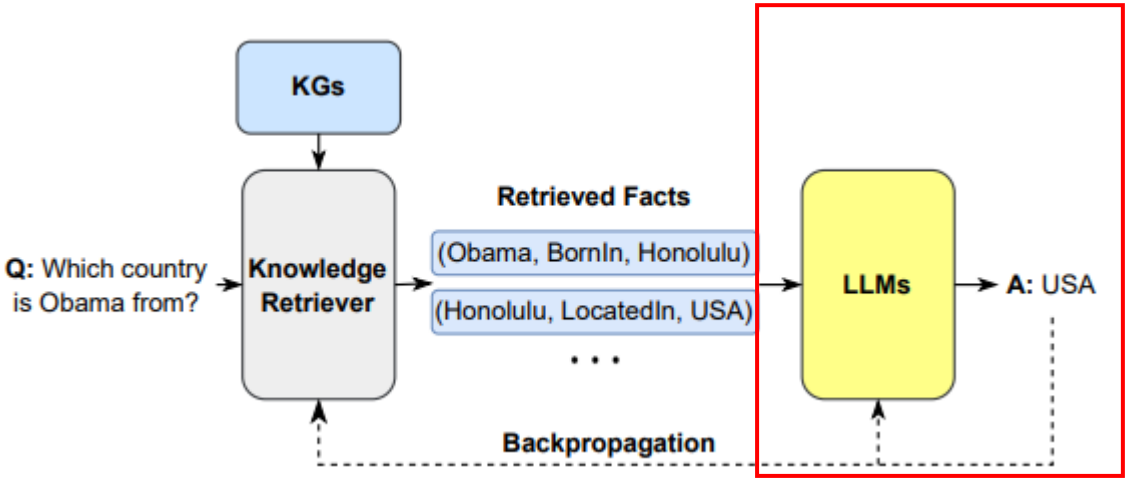


- Prompting 방법
  - ✓ Soft Prompt
    - 구조적인 정보 보존
    - 벡터로 압축 시 정보 손실 가능성
  - ✓ Hard Prompt
    - 자연어 형태
    - 최적화되지 않는다는 단점



Introduction

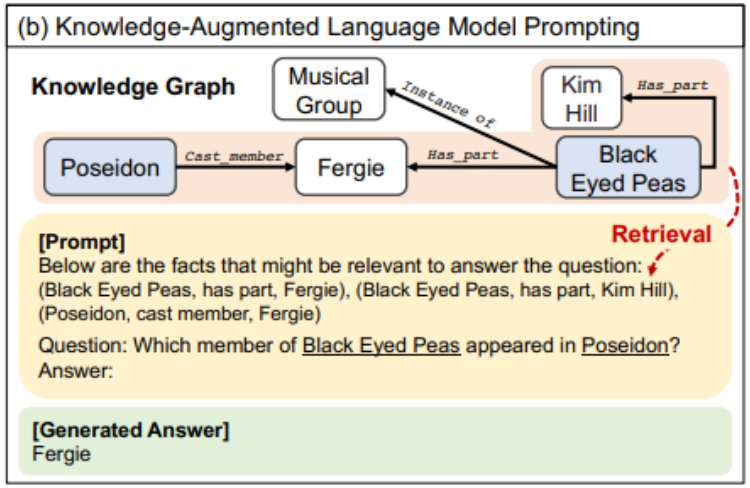
파이프라인(Retrieval and Reasoning)



- 학습 여부
  - ✓ Frozen
  - ✓ Instruction Tuning
  - ✓ LoRA(Low-Rank Adaptation)

Chapter 3:
관련 논문/모델 소개

[2023][Arxiv] KAPING



- Retrieval
  - ✓ Triple을 텍스트로 변환 후 임베딩 공간에 표현 후 질문과의 유사성(Cosine)을 기반으로 검색

- Reason
  - ✓ 검색된 Triple을 Prompt에 입력
  - ✓ 추가적인 Finetuning 필요 없음

- Knowledge-augmented language model prompting for zero-shot knowledge graph question answering

[2024][NIPS] G-retriever

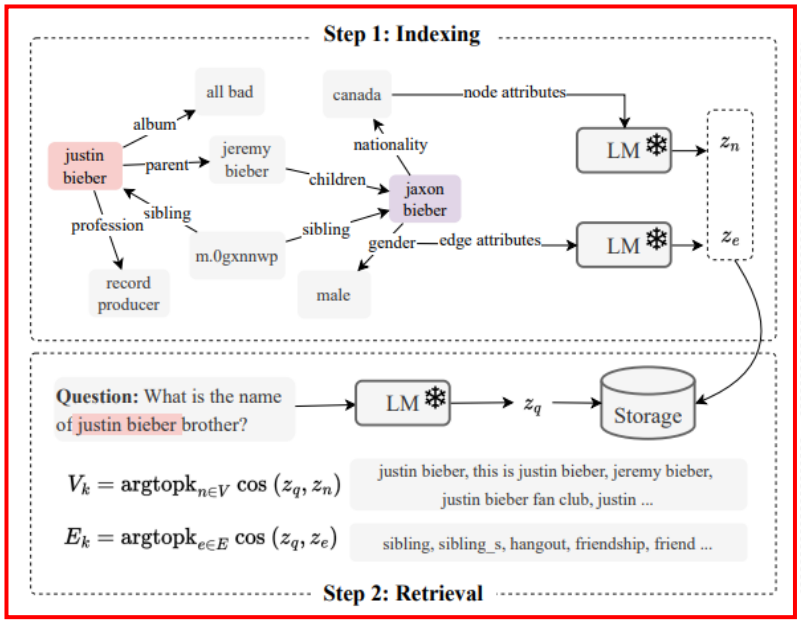
- 기존 문제점
  - ✓ 그래프를 자연어로 변환 시 확장성 문제 발생
    - Sequence로 변환 시 Token이 많아지며, 제한 시 정보 손실

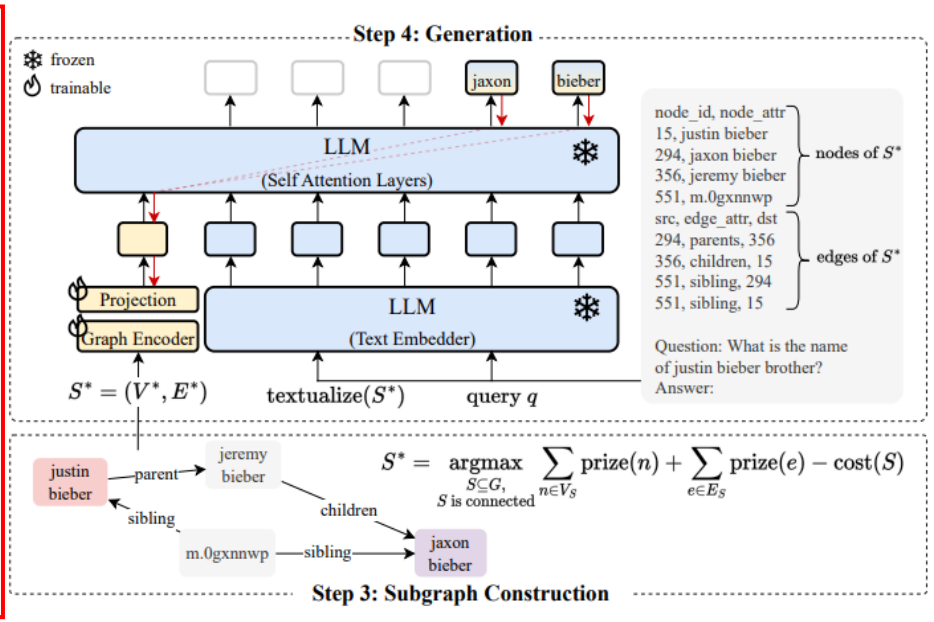
    => RAG를 통해 질문과 관련된 정보만 검색

- 특징
  - ✓ Textual Graph에 대해 처음으로 RAG를 도입
  - ✓ Prize-Collecting Steiner Tree(PCST) 최적화 알고리즘을 통해 Subgraph 검색
  - ✓ 그래프 정보와 그래프의 텍스트 정보를 동시에 고려

[2024][NIPS] G-retriever

Architecture



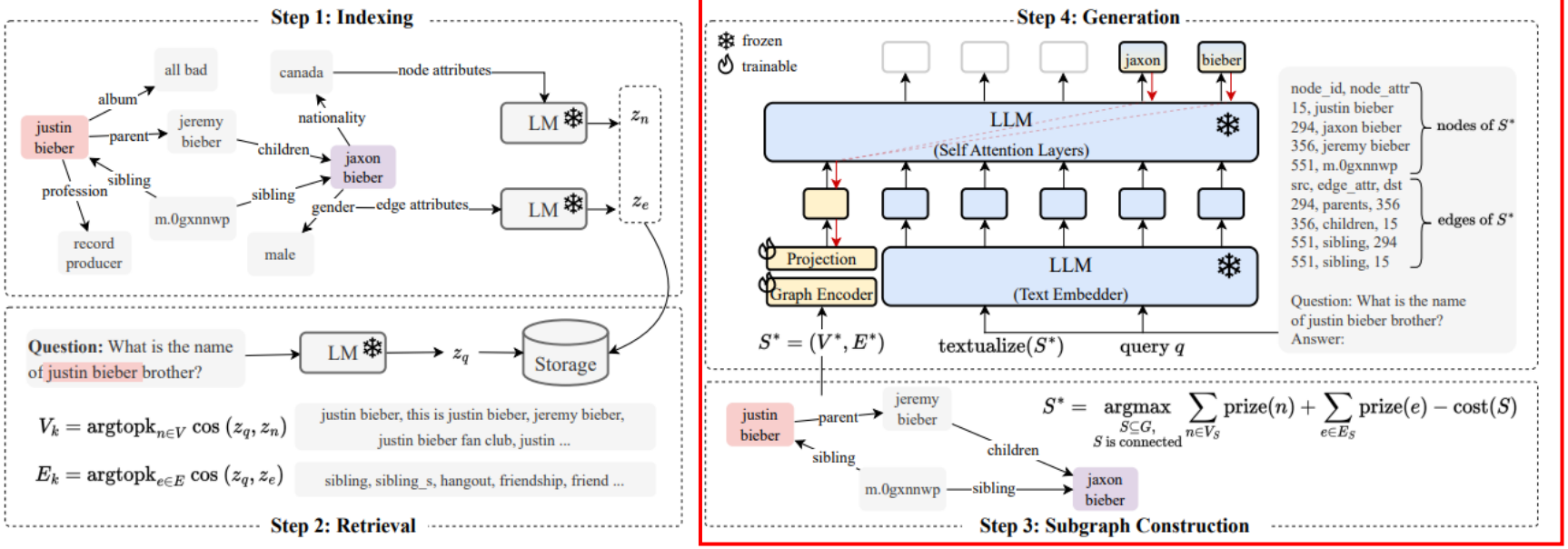


- Indexing
  - ✓ 사전 학습된 LM을 기반으로 Node와 Edge의 임베딩 생성 및 저장

- Retrieval
  - ✓ Query 임베딩 후 K-최근접 이웃 검색법을 사용하여 관련 Node, Edge Set을 구성

[2024][NIPS] G-retriever

Architecture



- Subgraph Construction
  - ✓ PCST 알고리즘을 통해 가능한 많은 Node와 Edge를 포함한 연결된 Subgraph 구축
  - ✓ 관련성이 높으면 보상, Cost를 최소화하는 Subgraph 추출

- Generation
  - ✓ GNN을 통해 구조적 정보, Textualized Graph를 결합하여 답변 생성

[2024][NIPS] G-retriever

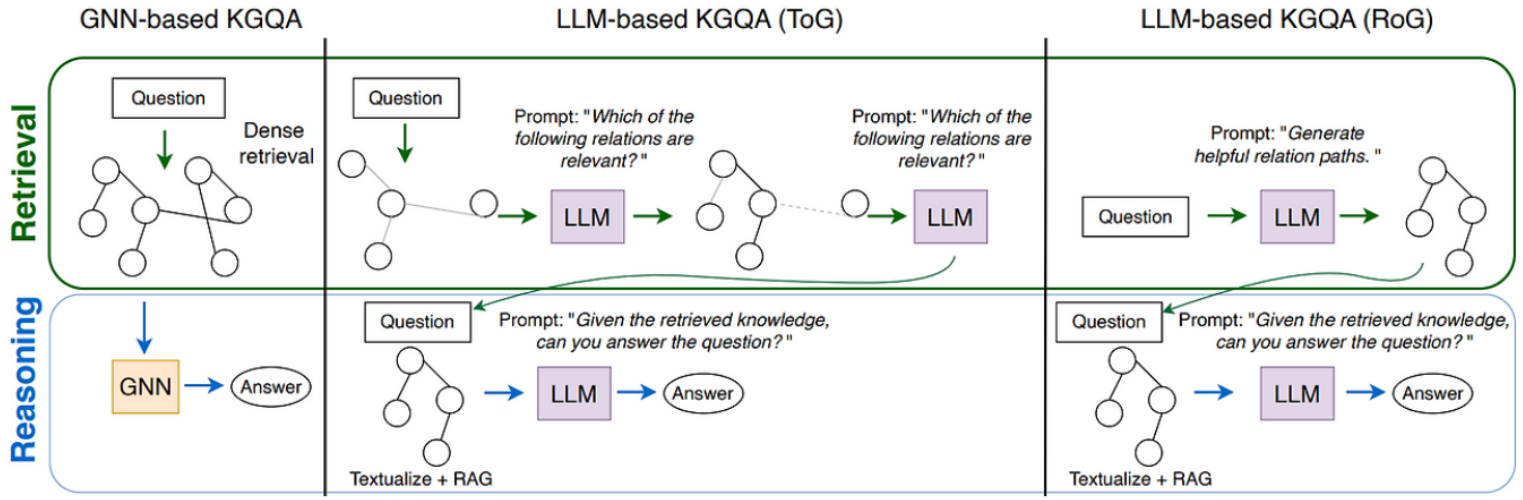
- 성능

Setting	Method	ExplaGraphs	SceneGraphs	WebQSP
Inference-only	Zero-shot	0.5650	0.3974	41.06
	Zero-CoT [18]	0.5704	0.5260	51.30
	CoT-BAG [44]	0.5794	0.5680	39.60
	KAPING [1]	0.6227	0.4375	52.64
Frozen LLM w/ PT	Prompt tuning	0.5763 ± 0.0243	0.6341 ± 0.0024	48.34 ± 0.64
	GraphToken [31]	0.8508 ± 0.0551	0.4903 ± 0.0105	57.05 ± 0.74
	G-Retriever	0.8516 ± 0.0092	<u>0.8131 ± 0.0162</u>	<u>70.49 ± 1.21</u>
	$\Delta_{\text{Prompt tuning}}$	↑ 47.77%	↑ 28.23%	↑ 45.81%
Tuned LLM	LoRA	<u>0.8538 ± 0.0353</u>	0.7862 ± 0.0031	66.03 ± 0.47
	G-Retriever w/ LoRA	**0.8705 ± 0.0329**	**0.8683 ± 0.0072**	**73.79 ± 0.70**
	Δ_{LoRA}	↑ 1.95%	↑ 11.74%	↑ 10.44%

- 한계점
  - ✓ 임베딩 유사성에만 의존
    - Noise 유입
  - ✓ 연결된 Subgraph만을 고려
  - ✓ Multi-hop

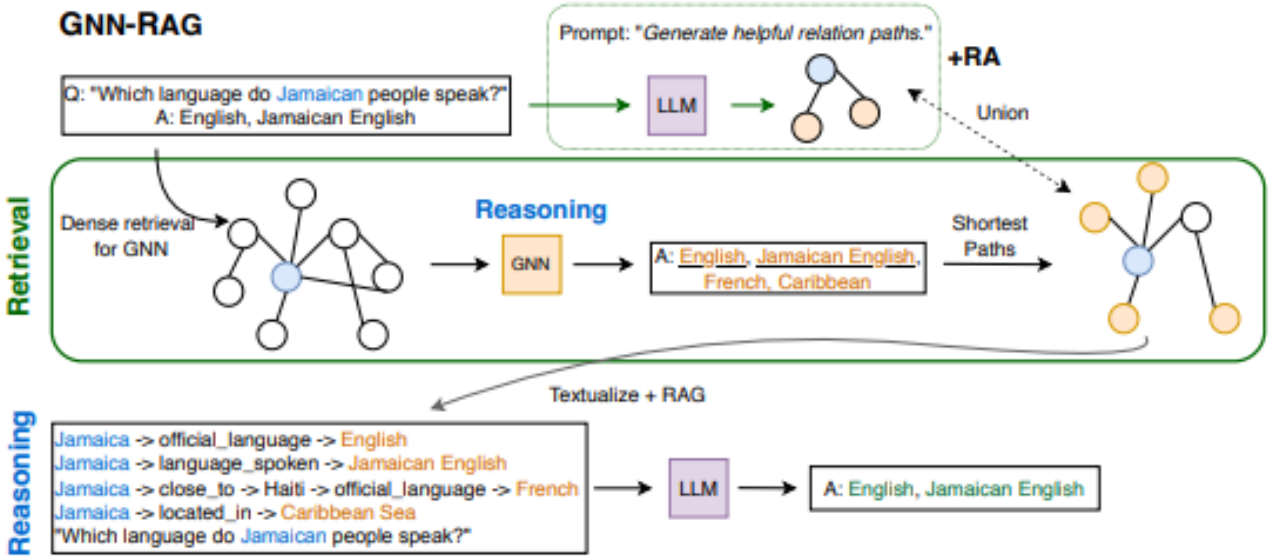
[2024][Arxiv] GNN-RAG

- 특징
  - ✓ GNN 기반 Retriever
    - GNN의 그래프 처리 능력을 활용
  - ✓ 임베딩 기반 방법에 비해 복잡한 그래프 상호작용을 처리하고 Multi-hop question에 답변할 수 있음



[2024][Arxiv] GNN-RAG

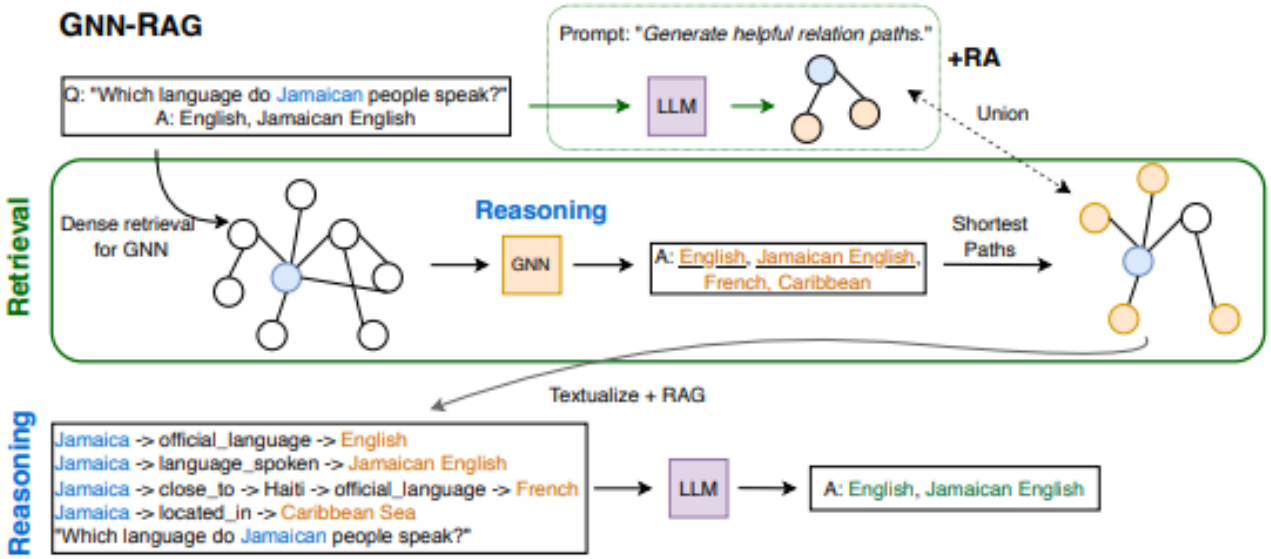
Architecture



- Retrieval
  - ✓ 그래프 내 모든 노드는 GNN Representation을 기준으로 정답/오답 여부에 따라 점수 (Softmax)
    - 확률 점수가 가장 높은 노드들을 정답 후보로 반환
  - ✓ 정답 후보와 Topic entity를 연결하는 Shortest path를 추출하여 Reasoning Path를 나타냄
  - ✓ GNN의 한계인 1-hop은 LLM을 통해 검색

[2024][Arxiv] GNN-RAG

Architecture



- Reasoning
  - ✓ Retrieval를 통해 얻은 reasoning path를 텍스트화(Verbalization)하여 LLM의 입력으로 제공

[2024][Arxiv] GNN-RAG

- 결과

Type	Method	WebQSP			CWQ		
		Hit	H@1	F1	Hit	H@1	F1
KG+LLM	KD-CoT [Wang et al., 2023]	68.6	–	52.5	55.7	–	–
	StructGPT [Jiang et al., 2023a]	72.6	–	–	–	–	–
	KB-BINDER [Li et al., 2023]	74.4	–	–	–	–	–
	ToG+LLaMA2-70B [Sun et al., 2024]	68.9	–	–	57.6	–	–
	ToG+ChatGPT [Sun et al., 2024]	76.2	–	–	58.9	–	–
	ToG+GPT-4 [Sun et al., 2024]	82.6	–	–	**69.5**	–	–
	RoG [Luo et al., 2024]	85.7	80.0	70.8	62.6	57.8	56.2
GNN+LLM	G-Retriever [He et al., 2024]	–	70.1	–	–	–	–
	GNN-RAG (**Ours**)	85.7	80.6	71.3	66.8	61.7	59.4
	GNN-RAG+RA (**Ours**)	**90.7**	**82.8**	**73.5**	68.7	**62.8**	**60.4**

- 의의
  - ✓ GNN이 Multi-hop에서 좋은 Retriever인지 증명

Retriever	1-hop questions		2-hop questions	
	#Input Tok.	%Ans. Cov.	#Input Tok.	%Ans. Cov.
RoG [Luo et al., 2024]	150	**87.1**	435	82.1
GNN ($L = 1$)	112	83.6	2,582	79.8
GNN ($L = 3$)	105	82.4	357	**88.5**

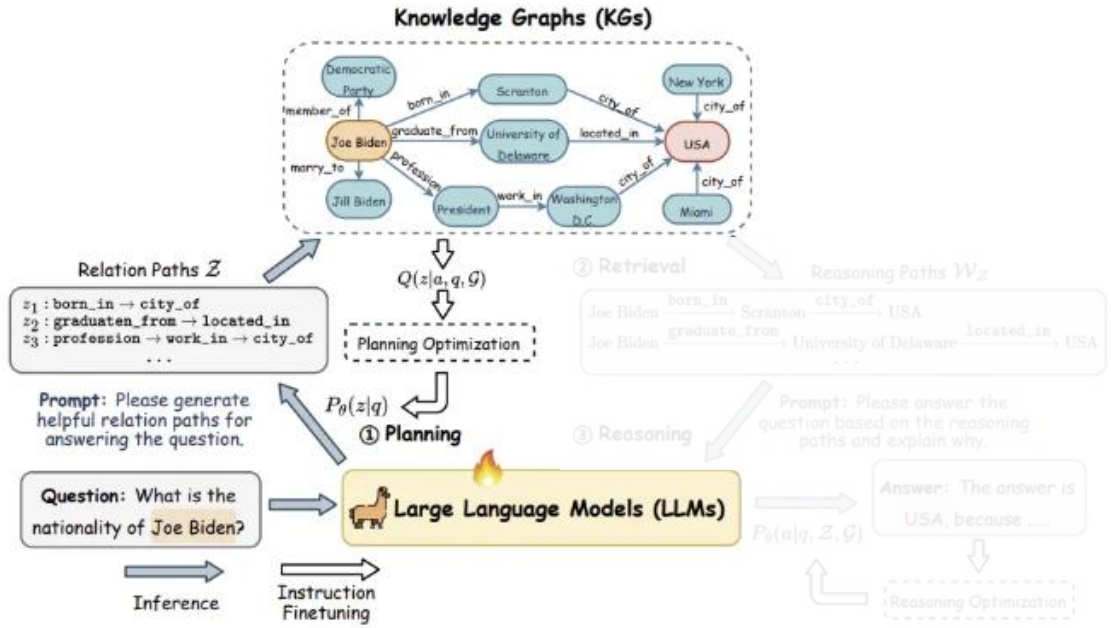
[2024][ICLR] RoG

- 기존 문제점
  - ✓ KG기반 LLM 추론 방식이 KG를 사실적 지식 기반으로만 여기고 추론을 위한 구조적 정보의 중요성 간과
  - ✓ KG의 relation path가 제공하는 의미론적 연결을 활용하지 못함

- 특징
  - ✓ Planning-Retrieval-Reasoning 프레임워크
  - ✓ LLM이 검색된 Path를 기반으로 설명 가능한 결과를 생성

[2024][ICLR] RoG

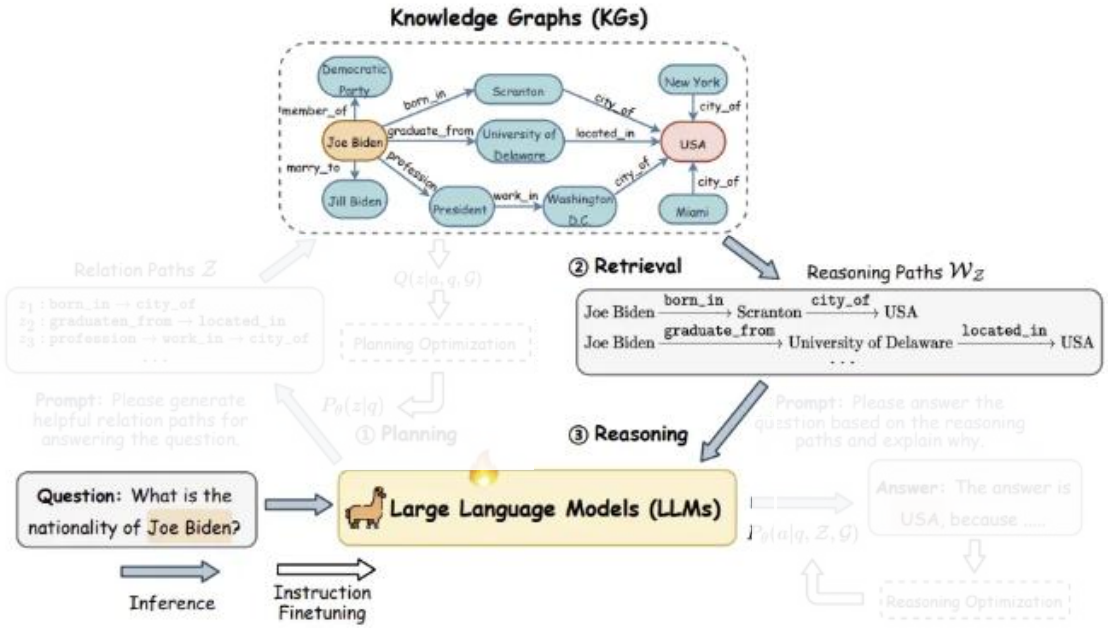
Architecture



- Planning
  - ✓ 프롬프트를 통해 LLM에게 relation path를 생성
    - Relation path : entity에 비해 안정적, 항상 최신의 지식 검색 가능
  - ✓ KG에 기반한 relation path를 생성하기 위해 Instruction Tuning
    - Shortest path

[2024][ICLR] RoG

Architecture



- Retrieval-Reasoning
  - ✓ Relation path를 기반으로 BFS를 통해 Reasoning path를 검색
  - ✓ 질문과 Reasoning path를 기반으로 답변 생성

[2024][ICLR] RoG

- 한계점
  - ✓ Shortest path를 Supervision으로 사용
    - 실제 정답을 추론하는 Path가 Shortest path가 아닌 경우 발생할 수 있음
  - ✓ LLM 호출 횟수

- 의의
  - ✓ Relation path의 설정이 중요
  - ✓ 해석 가능한 결과를 생성하는 데 효과적

[2025][ICLR] GCR

- 기존 문제점 (RoG)
  - ✓ 여전히 Hallucination 문제 발생
    - KG 위에서 추론을 수행해도 33%의 Hallucination

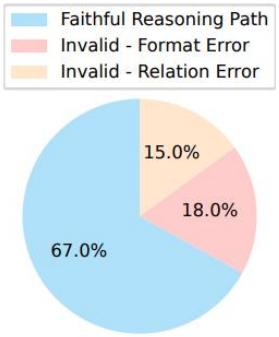
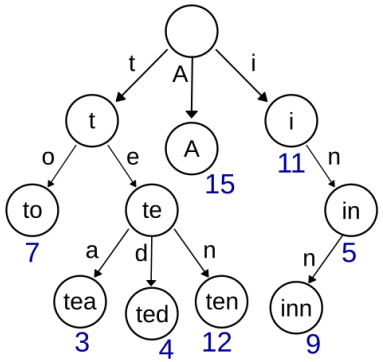


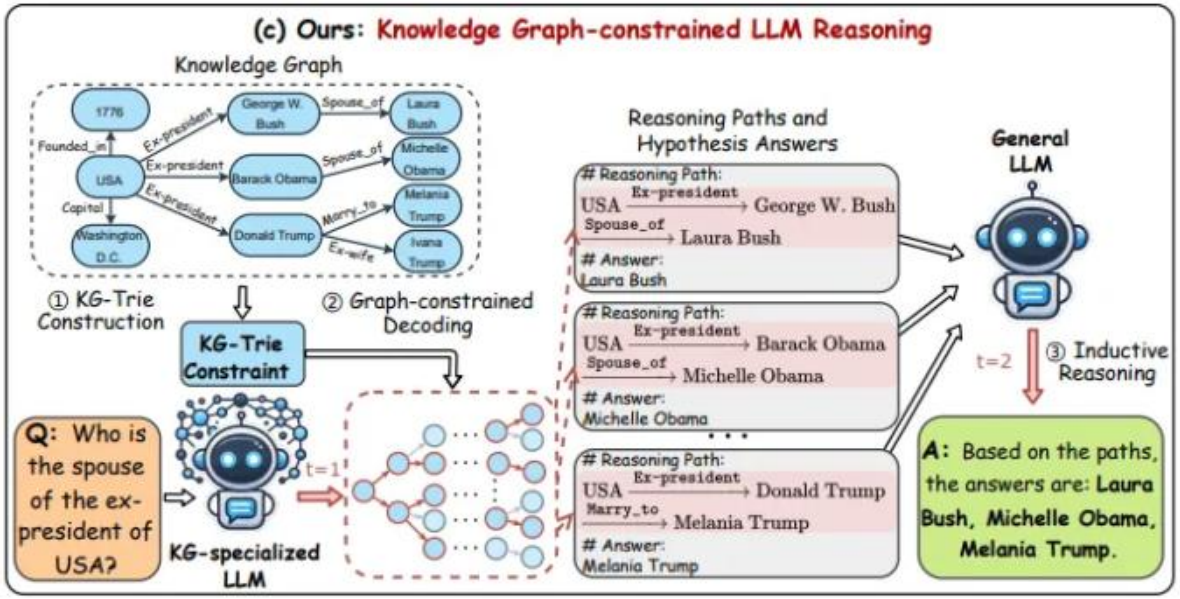
Figure 1. Analysis of reasoning errors in RoG (Luo et al., 2024).

- 특징
  - ✓ KG 구조를 LLM의 디코딩 과정에 통합
    - 신뢰할 수 있는 Path를 직접 생성
  - ✓ KG-Trie 구축
    - Trie : 문자열 집합을 압축하여 저장하는 데이터 구조



[2025][ICLR] GCR

Architecture

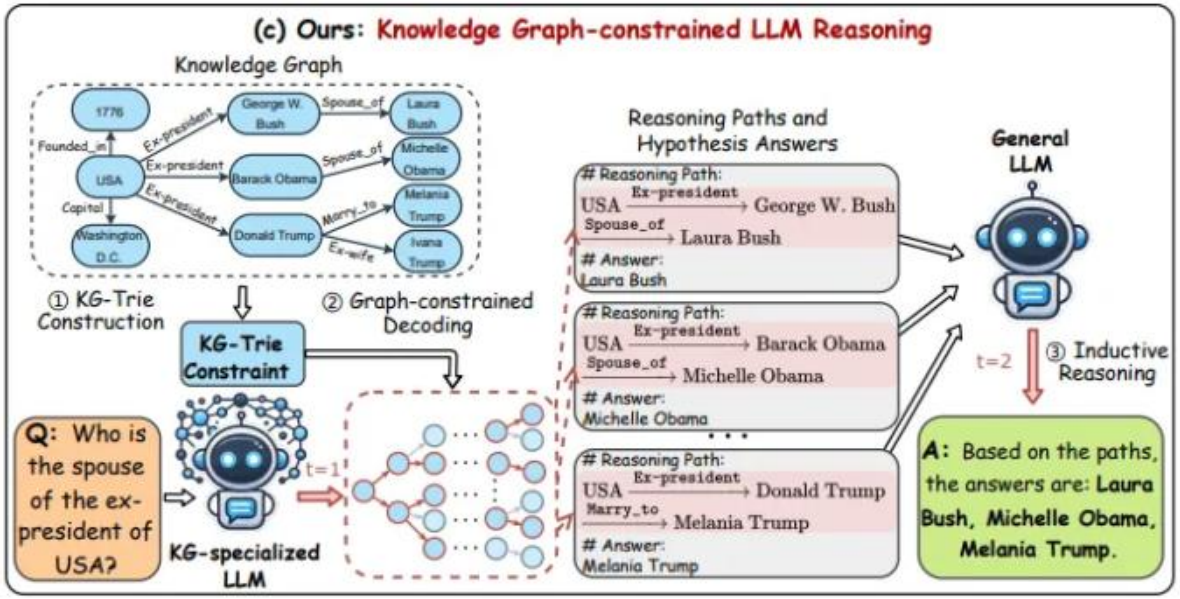


- TRIE Construction
  - ✓ Topic entity로 부터 BFS를 통해 Path 탐색
  - ✓ Tokenization을 통해서 Trie 형태로 저장

  => LLM의 출력 토큰을 Prefix로 시작하는 경우로 제한

[2025][ICLR] GCR

Architecture

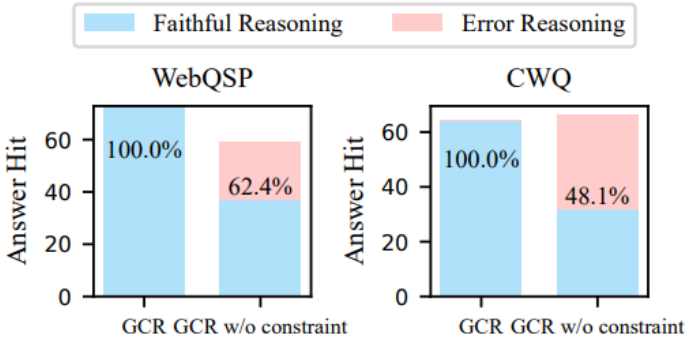


- Graph Inductive Reasoning
  - ✓ 생성된 reasoning path와 hypothesis answer를 기반으로 최종 정답 도출

[2025][ICLR] GCR

- 한계점
  - ✓ Hallucination 정의
    - KG에 근거한 Path를 탐색하는 건 맞지만, 실제 KG는 불완전하고, 부정확한 정보가 포함될 수 있음
  - ✓ 시간 복잡성 문제
  - ✓ 항상 의미 있는 경로를 찾는 데 성공하지는 못하여 잘못된 답을 생성

- 의의
  - ✓ 신뢰할 수 있는 추론 수행



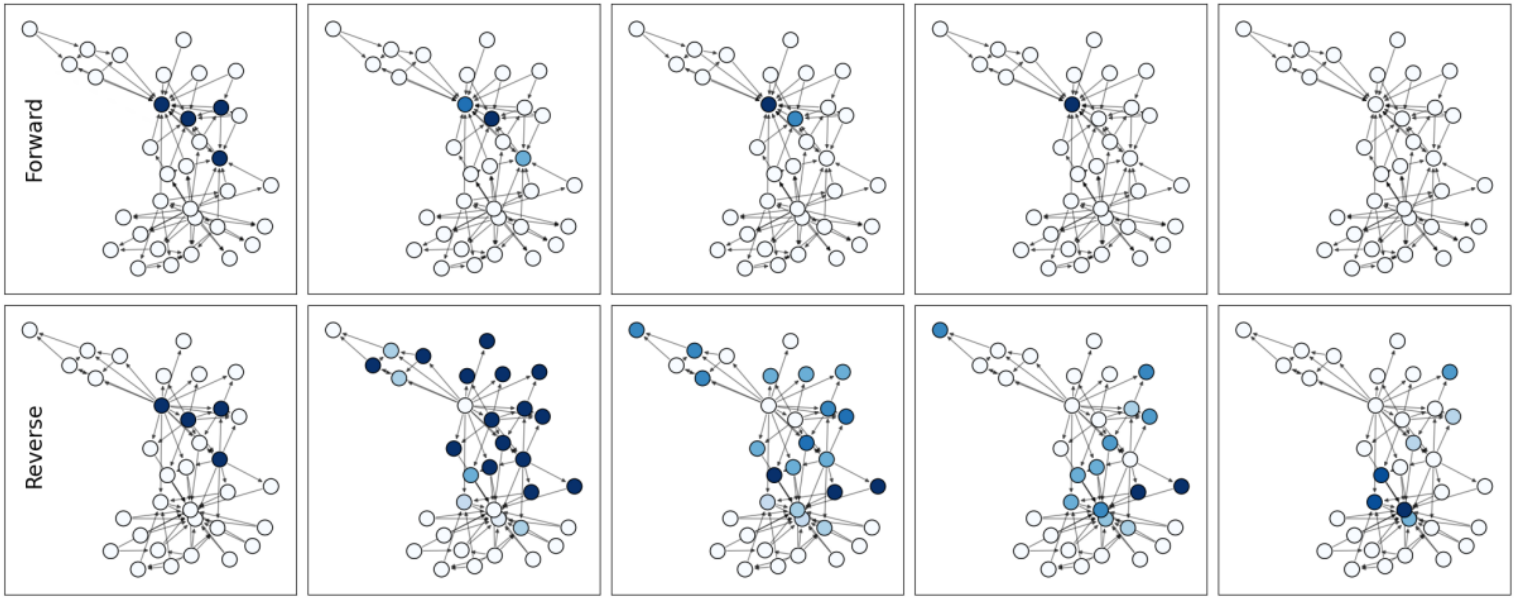
[2025][ICLR] SubgraphRAG

- 기존 문제점
  - ✓ GNN 기반 Retriever는 topic entity와 entity 간의 거리를 반영하지 못함
  - ✓ 모델의 복잡성과 추론 능력 사이에는 Trade off가 존재
    - LLM을 여러 번 호출 = Query 당 높은 계산 복잡성
    - GNN, LSTM과 같은 lightweight model은 계산 비용은 적지만 추론이 어려움

- 특징
  - ✓ Lightweight MLP를 기반으로 효율성
  - ✓ Topic entity에서의 구조적 거리를 계산 (DDE)
  - ✓ 병렬 triple scoring을 통해 검색 속도 개선

[2025][ICLR] SubgraphRAG

Architecture

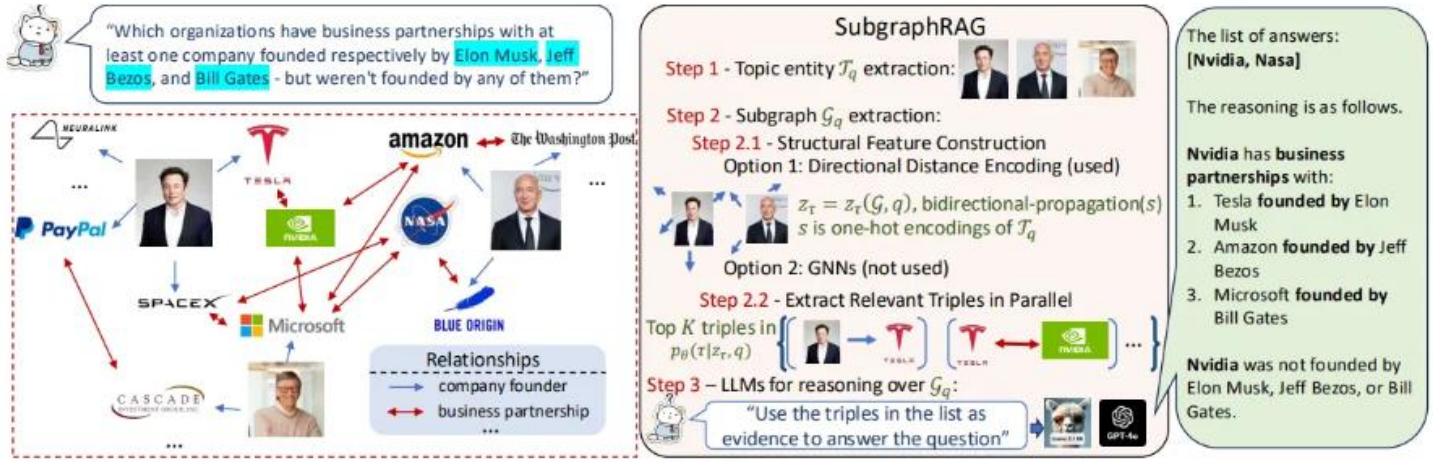


- F
  - ✓ 질문을 임베딩
  - ✓ 벡터 DB를 통해, triple의 head, relation, tail의 임베딩을 검색
  - ✓ DDE(Directional Distance Encoding) 계산

$$[z_q, z_h, z_r, z_t, DDE]$$

[2025][ICLR] SubgraphRAG

Architecture



- Reason
  - ✓ 검색된 Triple을 Linearization하여 LLM의 프롬프트에 입력
    - 논문 상 100개 기준

[2025][ICLR] SubgraphRAG

- 결과

	WebQSP		CWQ	
	Macro-F1	Hit	Macro-F1	Hit
G-Retriever	53.41	73.46	-	-
RoG-Joint	70.26	86.67	54.63	61.94
RoG-Sep	66.45	82.19	53.87	60.55
EtD	-	82.5	-	62.0
GNN-RAG	71.3	85.7	59.4	66.8
SubgraphRAG + Llama3.1-8B	70.57	86.61	47.16	56.98
SubgraphRAG + Llama3.1-70B	74.70	86.24	51.78	57.89
SubgraphRAG + ChatGPT	69.21	83.11	49.13	56.27
SubgraphRAG + GPT4o-mini	77.45	90.11	54.13	62.02
SubgraphRAG + GPT4o	76.46	89.80	59.08	66.69
SubgraphRAG + GPT4o-mini (200)	77.82	90.54	54.69	63.49
SubgraphRAG + GPT4o (200)	**78.24**	90.91	**59.42**	67.49
SubgraphRAG + GPT4o-mini (500)	77.67	**91.22**	55.41	64.97

- 한계점
  - ✓ 100개의 Triple을 입력
    - 구조적인 관계 반영 X
  - ✓ Heuristic Supervision
    - 실제 유용한 정보가 Shortest path에만 존재하는 것은 아님

현재 연구 방향

- Shortest path 뿐 아니라 추가적인 Path 고려
  - ✓ 다소 돌아가더라도 중요한 정보(의미 있고 맥락적)를 제공할 수 있음
  - ✓ 포괄적이고 정확한 추론 수행

- Retriever와 Reasoner 독립적으로 작동
  - ✓ End-to-End 방식으로 학습
  - ✓ 효율성

- 추가적으로
  - ✓ Intent 고려 (추천 시스템)
    - ▪ 다양한 Path를 탐색
  - ✓ Path들의 정답으로 가는 Consistency를 학습