DeepShark 학부연구생 김승겸

Self-RAG

KARMA 확장 연구를 위한 모듈

2026.7.03

목 차

연구 배경 및 문제 정의

기존 연구

로컬 LLM 기반 KARMA 확장

한계

- 문맥 누락
- 관계 술어 파편화
- 추출된 엔터티의 중복

후속 연구

문장 단위로 RAG를 이용하여
세밀한 트리플 추출

문장 단위 RAG 기반 트리플 추출 파이프라인

PDF 입력 후 텍스트 추출

↓

문장 단위 분할

↓

각 문장을 검색 질의로 사용

↓

RAG를 통해 관련 문서 또는 기존 지식 검색

↓

원문 문장 + 주변 문맥 + 검색 결과를 LLM에 입력

↓

트리플 추출

↓

엔터티 / 관계 정규화

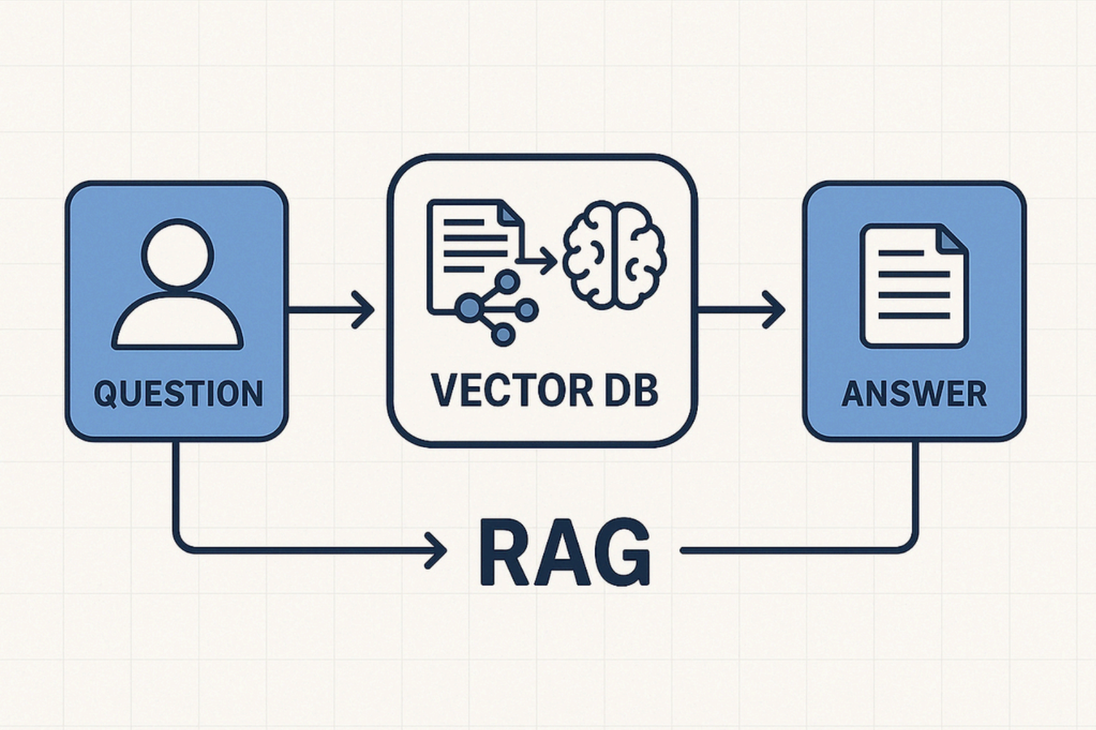
↓

중복 제거 / 신뢰도 평가

↓

지식 그래프 통합

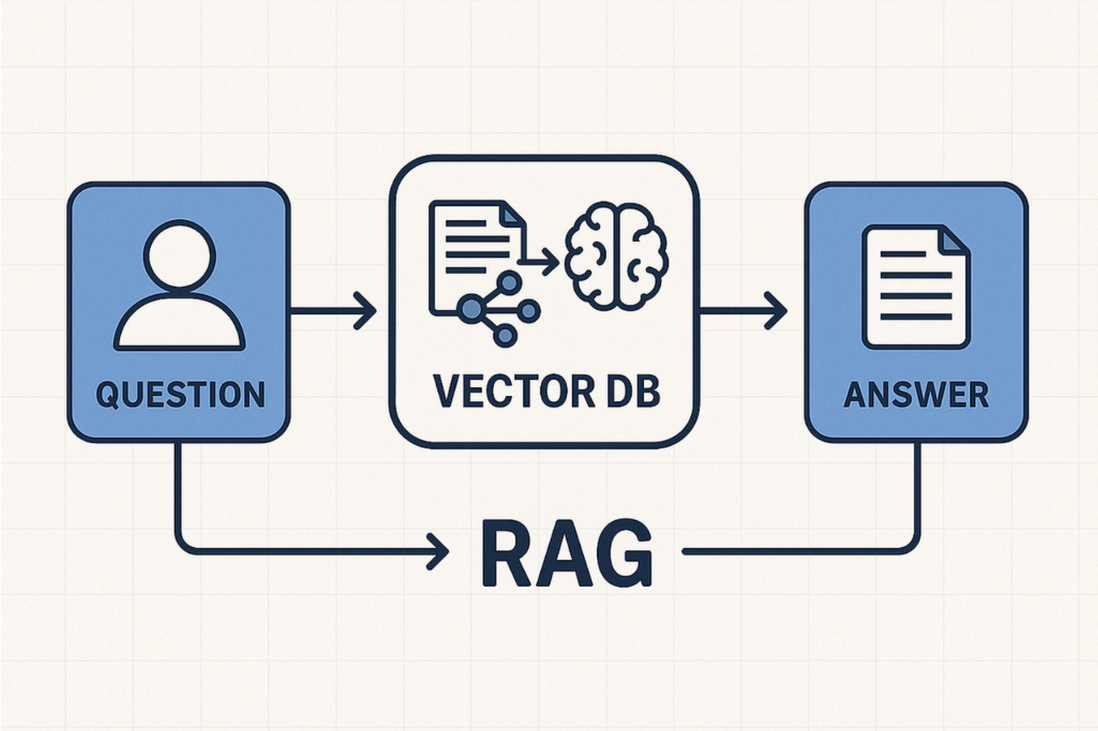
일반 RAG 적용의 한계



일반 RAG 구조

PDF 문장 입력
→ top-k 문서 검색
→ 검색 결과 + 입력
→ LLM 생성
→ 트리플 생성

일반 RAG 적용의 한계

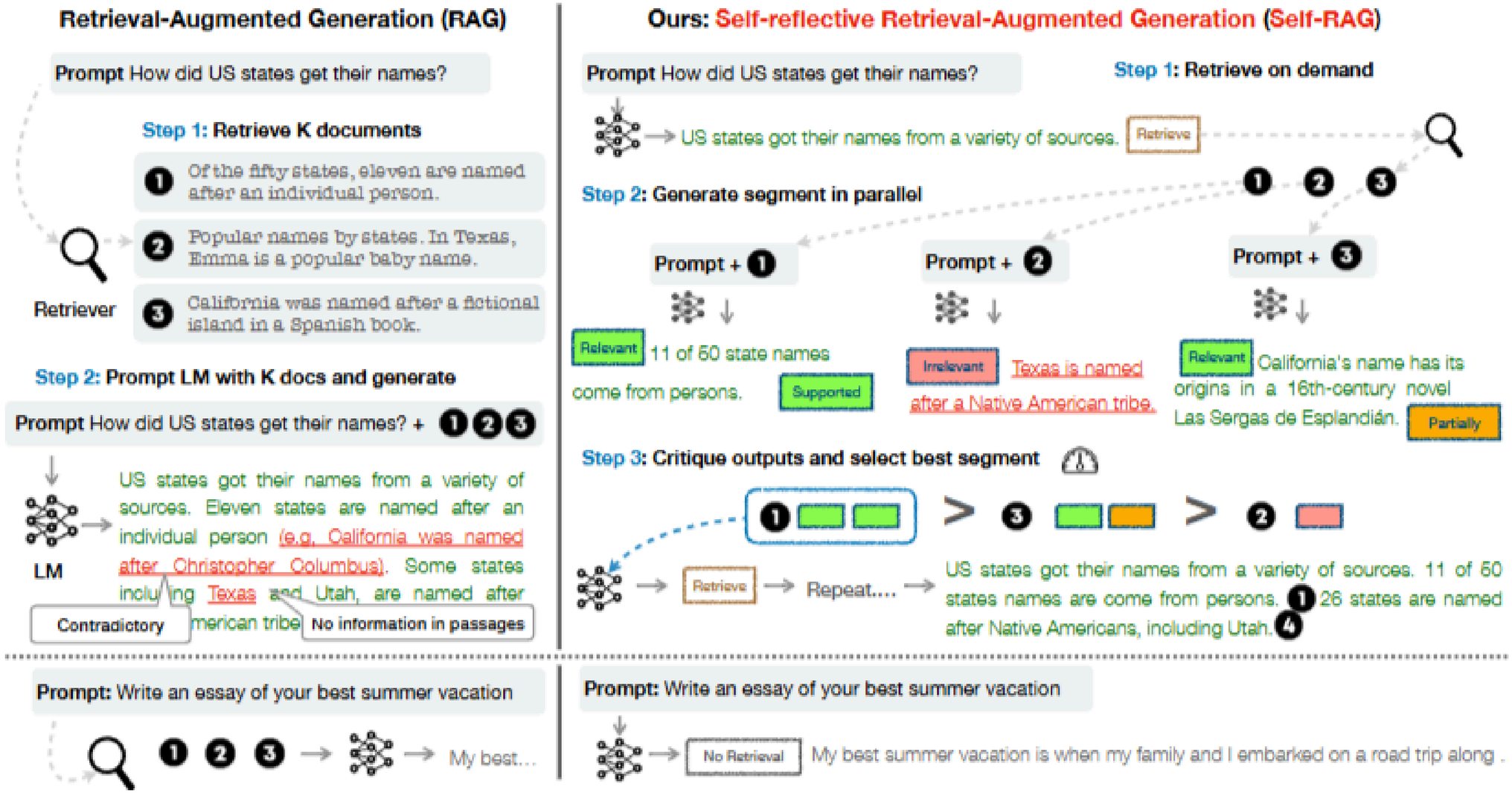


- 모든 문장이 검색을 필요로 하지는 않는다
- top-k 검색 결과가 항상 현재 문장의 트리플 추출에 유용한 것은 아니다
- 원문에 존재하지 않고, 검색 결과에만 있는 정보가 트리플로 생성될 수 있다
- LLM이 어떤 근거를 기준으로 삼았는지 불명확하다

Self-RAG 구조

입력
→ 검색 필요성 판단
→ 필요한 경우 검색
→ 검색 결과 관련성 평가
→ 근거 기반 생성

Self-RAG 개요


Retrieval-Augmented Generation (RAG)
Prompt How did US states get their names?
Step 1: Retrieve K documents
1 Of the fifty states, eleven are named after an individual person.
2 Popular names by states. In Texas, Emma is a popular baby name.
3 California was named after a fictional island in a Spanish book.
Retriever
Step 2: Prompt LM with K docs and generate
Prompt How did US states get their names? + 1 2 3
LM
US states got their names from a variety of sources. Eleven states are named after an individual person (e.g. California was named after Christopher Columbus). Some states inclu ing Texas and Utah, are named after merican tribe
Contradictory
No information in passages
Ours: Self-reflective Retrieval-Augmented Generation (Self-RAG)
Prompt How did US states get their names?
Step 1: Retrieve on demand
US states got their names from a variety of sources.
Retrieve
Step 2: Generate segment in parallel
Prompt + 1
Prompt + 2
Prompt + 3
Relevant 11 of 60 state names come from persons.
Supported
Irrelevant Texas is named after a Native American tribe.
Relevant California's name has its origins in a 16th-century novel Las Sergas de Esplandián.
Partially
Step 3: Critique outputs and select best segment
Retrieve
Repeat....
US states got their names from a variety of sources. 11 of 60 states names are come from persons. 1 26 states are named after Native Americans, including Utah. 4
Prompt: Write an essay of your best summer vacation
My best...
Prompt: Write an essay of your best summer vacation
No Retrieval
My best summer vacation is when my family and I embarked on a road trip along .

Self-RAG의 전체 학습 구조

Critic Model

입력과 출력이 주어졌을 때,
해당 출력에 대한 reflection token을 예측하도록 학습

→ 무엇이 좋은 답변인지 판단하는 기준을 학습

Generator Model

일반 텍스트 출력뿐만 아니라
reflection token도 함께 생성하도록 학습

→ Critic이 부여한 reflection token이 포함된 데이터를 학습하여,
답변과 reflection token을 함께 생성하는 법을 학습

Reflection Token

"Self-RAG가 (텍스트 + reflection token) 생성"

"검색의 필요성, 검색 문서의 관련성, 생성 결과의 근거성 평가, 출력의 유용성 평가하는 특수 토큰"

Retrieve

- "검색이 필요한가?"
- yes / no / continue

ISREL

- "검색 문서가 실제로 관련 있는가?"
- relevant / irrelevant

ISSUP

- "출력이 해당 문서로 충분히 뒷받침 되는가?"
- fully / partially / no support

ISUSE

- "전체적으로 출력이 유용한가?"
- 1~5 점의 유용성 점수

핵심 수식: Critic Model 학습

$$\max_{C} \mathbb{E}_{((x,y),r)\sim\mathcal{D}_{critic}} \log p_C(r|x,y)$$

C : critic model(입력과 출력이 주어졌을 때, reflection token을 예측하도록 학습)

x : 입력

y : 출력

r : reflection token

D_{critic} : critic 학습 데이터

$p_C(r|x,y)$: 입력 x와 출력 y가 주어졌을 때 reflection token r을 예측할 확률

핵심 수식: Generator Model 학습

$$\max_{\mathcal{M}} \mathbb{E}_{(x,y,r)\sim\mathcal{D}_{gen}} \log p_{\mathcal{M}}(y, r|x).$$

$\mathcal{M}$: generator model(일반 텍스트와 reflection token을 동시에 예측하도록 학습)

x : 입력

y : 출력(일반 텍스트)

r : reflection token

D_{gen} : reflection token이 포함된 학습 데이터

$p_{\mathcal{M}}(y, r|x)$: 입력 x가 주어졌을 때 출력 y와 reflection token r을 예측할 확률

Self-RAG 학습 과정

Type	Input	Output	Definitions
Retrieve	x / x, y	{yes, no, continue}	Decides when to retrieve with $\mathcal{R}$
ISREL	x, d	{**relevant**, irrelevant}	d provides useful information to solve x.
ISSUP	x, d, y	{**fully supported**, partially supported, no support}	All of the verification-worthy statement in y is supported by d.
ISUSE	x, y	{**5**, 4, 3, 2, 1}	y is a useful response to x.

Algorithm 1 SELF-RAG Inference

Require: Generator LM $\mathcal{M}$, Retriever $\mathcal{R}$, Large-scale passage collections $\{d_1, \ldots, d_N\}$
Input: input prompt x and preceding generation $y_{<t}$, **Output:** next output segment y_t
$\mathcal{M}$ predicts Retrieve given $(x, y_{<t})$
if Retrieve == Yes **then**
  Retrieve relevant text passages $\mathbf{D}$ using $\mathcal{R}$ given (x, y_{t-1}) ▷ Retrieve
  $\mathcal{M}$ predicts ISREL given x, d and y_t given $x, d, y_{<t}$ for each $d \in \mathbf{D}$ ▷ Generate
  $\mathcal{M}$ predicts ISSUP and ISUSE given x, y_t, d for each $d \in \mathbf{D}$ ▷ Critique
  Rank y_t based on ISREL, ISSUP, ISUSE ▷ Detailed in Section 3.3
else if Retrieve == No **then**
  $\mathcal{M}_{gen}$ predicts y_t given x ▷ Generate
  $\mathcal{M}_{gen}$ predicts ISUSE given x, y_t ▷ Critique

1. GPT-4를 이용해 reflection token의 정답지 데이터셋(critic 학습 데이터) 구축

2. 이 데이터를 이용해 Llama 2-7B 기반 Critic Model을 학습

3. 학습된 Critic Model은 새로운 입력-출력 쌍에 대해
   reflection token(Retrieve, ISREL, ISSUP, ISUSE)을 예측

4. Critic Model이 예측한 reflection token을 기존 학습 데이터에 삽입하여
   Generator 학습용 데이터셋 생성

5. Generator Model은 일반 답변 y뿐만 아니라 reflection token r도 함께 생성하도록 학습

6. 최종적으로 Generator는 inference 단계에서
   검색 필요성 판단, 검색 결과 관련성 평가, 근거성 판단, 유용성 판단을
   스스로 수행

핵심 수식: Self-RAG의 추론 과정

$$\frac{p(\text{Retrieve} = \text{YES})}{p(\text{Retrieve} = \text{YES}) + p(\text{Retrieve} = \text{NO})} > \delta.$$

- 검색 필요성 진단
- 검색 확률이 임계값 δ보다 크면 검색 수행
- 그렇지 않으면 검색 없이 생성 진행

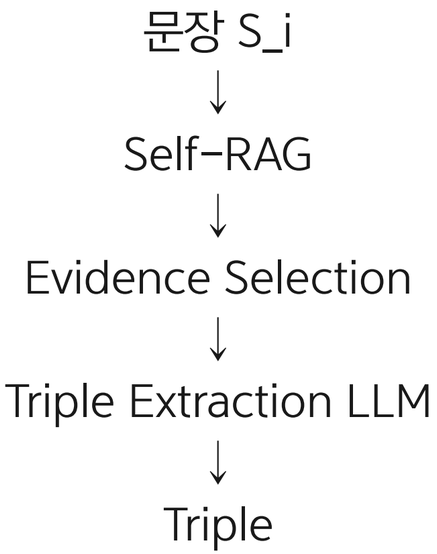
$$f(y_t, d, \text{Critique}) = p(y_t|x, d, y_{<t}) + \mathcal{S}(\text{Critique}), \text{where}$$

- 출력 후보 점수화
- y_t : 현재 생성할 출력 segment
- $p(y_t|x, d, y_<t)$: 생성 확률
- $S(Critique)$: reflection token 기반 평가 점수

$$\mathcal{S}(\text{Critique}) = \sum_{G \in \mathcal{G}} w^G s_t^G \text{ for } \mathcal{G} = \{\text{ISREL}, \text{ISSUP}, \text{ISUSE}\}$$

- Reflection Token Scoring
- s_t^G: 각 reflection token의 점수
- w^G : 각 reflection token에 부여하는 가중치
- ISREL: 검색 결과가 입력과 관련 있는가
- ISSUP: 출력이 근거에 의해 지지되는가
- ISUSE: 출력이 유용한가

후속 연구에서의 Self-RAG 적용



Self-RAG의 역할

- 검색 필요성 판단
- evidence 검색
- 검색 결과 관련성 평가
- 관련 evidence 선별

Q&A
감사합니다

2026.7.04