

# Upstage AI Lab

IR 경진 대회

: [3조] Scientific Knowledge Answering(RAG)

24.04.17

[www.fastcampus.co.kr](http://www.fastcampus.co.kr)

Copyright © FAST CAMPUS Corp. All Rights Reserved. 무단전재 및 재배포 금지

01

# 팀원 소개



## 목차

01. 팀원 소개 및 작업 방식 안내

02. Total Process

03. 결과 및 인사이트 공유

04. 프로젝트 회고

IR 계급 전쟁

# 흑백 개발자



인간을 잘 이해하는 AI개발자  
저는 정인웅 입니다.



Interested in

서비스 개발(앱, 웹)  
생성형 ai, EDA

Introduction

- 심리학(임상 및 상담) 석사,
- 소프트웨어 공학 학사

건강하고 성장하는 삶을 지향합니다.

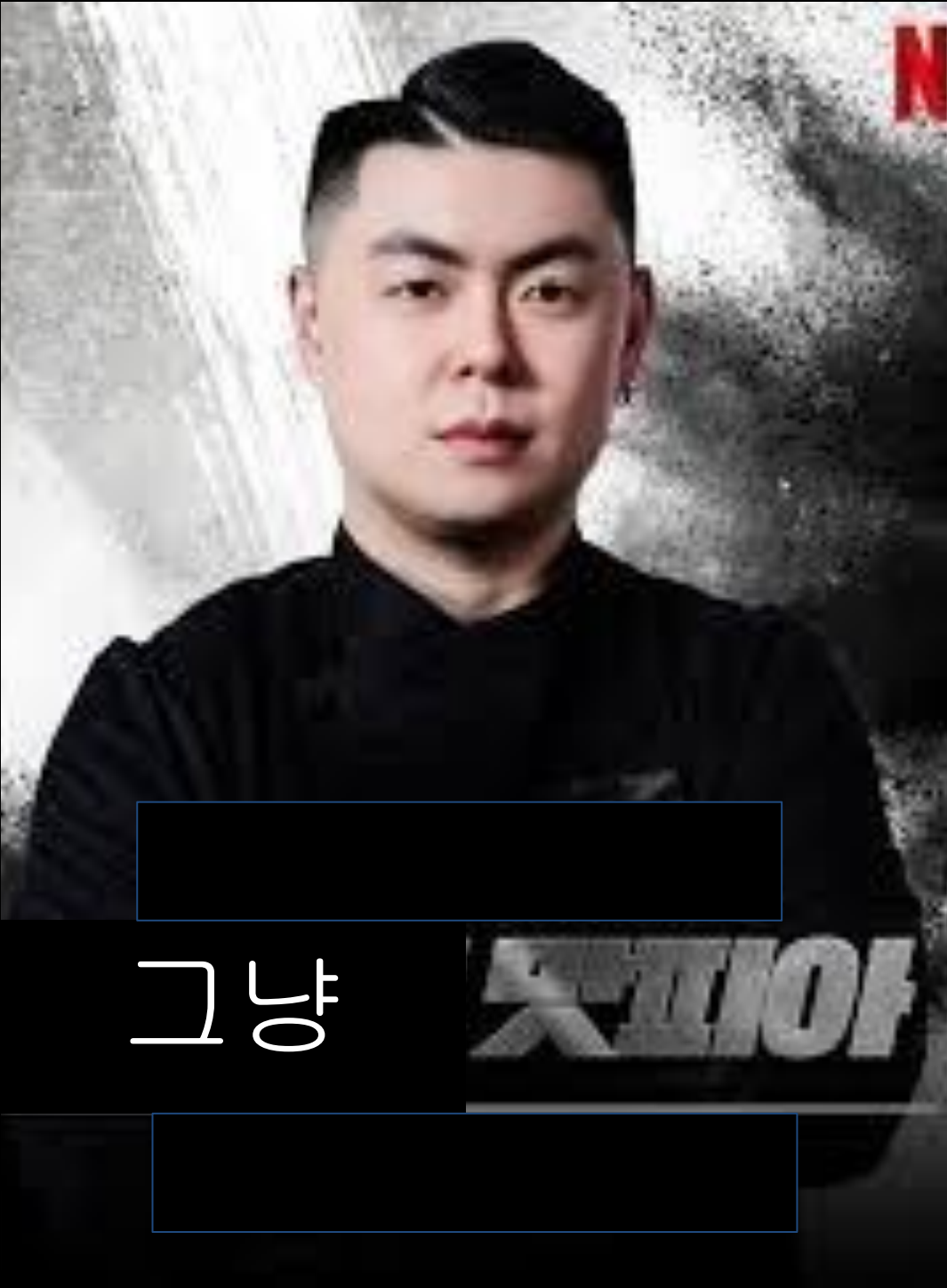
Role

중요 피처 선택, 데이터 모델 파라미터 최적화,  
앙상블  
팀장 역할, 운동의 중요성 설파

In Upstage AI Lab

서비스를 개발할 수 있는 기본기 익히길  
어떤 진로를 결정하든 기본기가 탄탄해지길

인간을 잘 이해하는 AI개발자  
저는 그냥 마피아입니다.



Interested in

서비스 개발(앱, 웹)  
생성형 ai, EDA

Introduction

- 심리학(임상 및 상담) 석사,
- 소프트웨어 공학 학사

건강하고 성장하는 삶을 지향합니다.

Role

중요 피처 선택, 데이터 모델 파라미터 최적화,  
앙상블  
팀장 역할, 운동의 중요성 설파

In Upstage AI Lab

서비스를 개발할 수 있는 기본기 익히길  
어떤 진로를 결정하든 기본기가 탄탄해지길



80%의 힘으로 10배로 노력하자!  
저는 이범희 입니다.



*Interested in*

- NLP를 활용한 Knowledge Tracking

*Introduction*

- 전공: 노어노문학 & 국어교육학

*Role*

- 팀원들과 함께 모든 프로젝트에 성실히 참여.

*In Upstage AI Lab*

- AI 관련 기본기 숙달
- AI 분야에서 함께 발전해갈 동료 확보

80%의 힘으로 10배로 노력하자!  
저는 트리플트리플(33)입니다.



*Interested in*

- NLP를 활용한 Knowledge Tracking

*Introduction*

- 전공: 노어노문학 & 국어교육학

*Role*

- 팀원들과 함께 모든 프로젝트에 성실히 참여.

*In Upstage AI Lab*

- AI 관련 기본기 숙달  
- AI 분야에서 함께 발전해갈 동료 확보



저는 박건민 입니다.



*Interested in*

- 사랑과 평화

*Introduction*

- 전공: 빅데이터비즈니스

*Role*

- 팀원들과 함께 모든 프로젝트에 성실히 참여.

*In Upstage AI Lab*

- 많이 배우고 갑니다.

저는 개발하는 돌아이입니다.



Interested in

- 사랑과 평화

Introduction

- 전공: 빅데이터비즈니스

Role

- 팀원들과 함께 모든 프로젝트에 성실히 참여.

In Upstage AI Lab

- 많이 배우고 갑니다.





# 트리플 트리플의 발표 공약





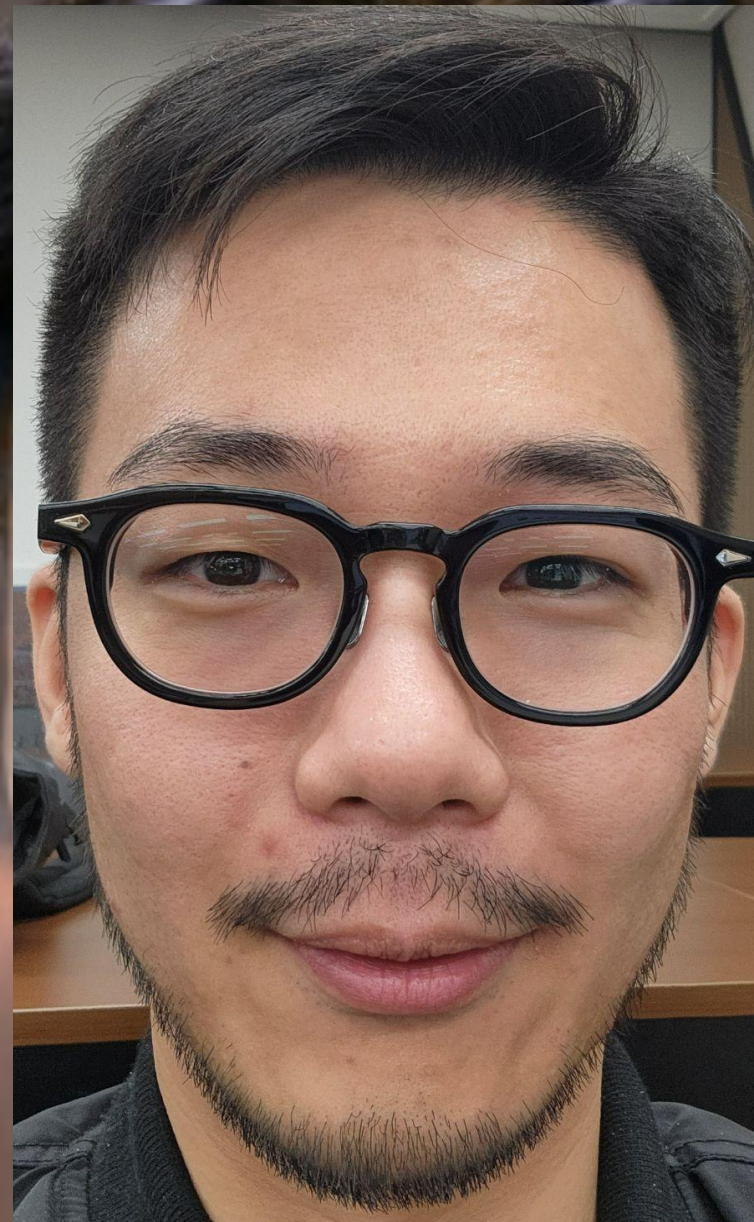
제가 제일 중요하게 생각하는 거는





발표의 익힘 정도인 것 같아요





근데 그 익힘이 굉장히 타이트해요, 맛있어요



맛있게  
발표하겠습니다!

# 프로젝트 진행 방법

프로젝트 진행 장소	오프라인 강의장
스크럼 진행 횟수 및 일정	모여서 진행하며 그때그때 상황 공유
프로젝트 진행 방법	<div>그냥 마피아: 모든 역할 + GEMINI api 구현</div> <div>트리플 트리플: 모든 역할 + 프롬프트 엔지니어링</div> <div>개발하는 돌아이: 모든 역할 + 각종 알고리즘 구현</div>



02

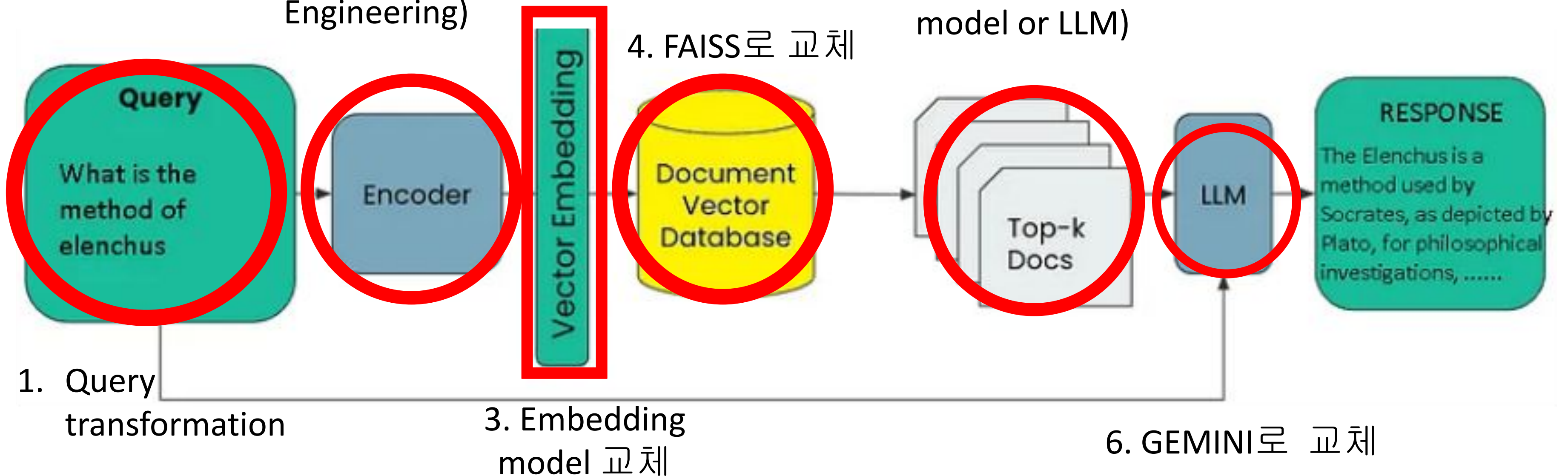
# Total Process



# Total Process

2. Standalone  
Query Generation  
(Prompt  
Engineering)

5. 문서 Re-rank  
방식 적용(Cohere  
model or LLM)



02-01

# 1. Query Transformation



# Query Transformation

eval\_jsonl query 수정

---

질문자의 query가 맥락이 난해한 경우가 많고,

한글과 알파벳이 혼용된 부분이 있어 이를 적절히 수정하는  
시도를 해 봄.

→ 성능 향상!









# Query Transformation

eval\_jsonl query 수정

```
{  
  "eval_id": 280,  
  "msg": [{"role": "user", "content": "Dmitri Ivanovsky가 누구야?"}]  
}
```



```
{  
  "eval_id": 280,  
  "msg": [{"role": "user", "parts": "바이러스를 발견한 드미트리 이바노프스키에 대해서 검색해줘"}]  
}
```

	baseline_code실행		<div>0.71210.7167</div>	2024.10.02 20:05	완료	
	basicmodel_p...ing_3		<div>0.77420.7788</div>	2024.10.04 21:32	완료	



02-02

## 2. Standalone Query Generation



# Prompt Engineering



# Standalone Query Generation

## Prompt Engineering

```
# RAG 구현에 필요한 질의 분석 및 검색 이외의 일반 질의 대응을 위한 LLM 프롬프트
persona_function_calling = """
## Role: 과학 상식 전문가

## Instruction
- 사용자가 대화를 통해 과학 지식에 관한 주제로 질문하면 search api를 호출할 수 있어야 한다.
- 과학 상식과 관련되지 않은 나머지 대화 메시지에는 적절한 대답을 생성한다.
"""
```

- **역할 부여 오류**(과학 외 모든 지식에 대해서도 답할 수 있어야 함), **추상적인 지시문**으로 인해 function\_call이 필요한 상황에 제대로 일어나지 않음.
- 이에 따라 standalone\_query가 생성되지 않고, topk도 아예 불러오지 않는 문제 발생.
- 베이스라인 코드 gpt-4o 기준 40개 이상의 문서에서 topk가 생성되지 않았음.
- 생성된 standalone\_query의 퀄리티도 좋지 않았음.



# Standalone Query Generation

## Prompt Engineering

```
persona function calling = ""
```

```
## Role: 인문, 사회, 과학, 공학 등 모든 분야의 지식에 통달한 전문가
```

역할 부여 수정

```
## Instruction
```

```
- Classify the last query into question, advisal request or casual conversation
```

```
query: 연구의 과정과 결과를 잘 기록해야 하는 이유? // question
```

구체적 지시문(one-shot prompting)

```
연구실에서 무엇인지 파악이 안된 가루를 저울로 옮기는 좋은 방법은? // advisal request
```

```
요새 너무 힘드네.. // casual conversation
```

```
- 사용자와 여러 번 대화를 나눈 경우, 대화 맥락을 바탕으로 마지막으로 나눈 대화의 의미를 추론한다.
```

```
- 사용자가 각종 지식을 주제로 질문을 하거나 설명을 요청하면, 시스템은 검색 API를 반드시 호출한다.
```

```
- 지식과 관련되지 않은 다른 일상적인 대화 메시지에 대해서는 적절한 응답을 생성한다.
```

```
""
```

반드시 불러오도록 하기 위한 조치..



# Standalone Query Generation

## Prompt Engineering

<input type="checkbox"/>	GPT4o_Kor_Kn...xpert		<div>0.7015 –</div> <div>0.7030 –</div>	2024.10.08 17:32	완료	
						
<input type="checkbox"/>	GPT4o_Kor_Kn...ied_2		<div>0.8129 –</div> <div>0.8152 –</div>	2024.10.08 18:45	완료	

# Standalone query에 ‘?’ 추가



# Standalone Query Generation

standalone query 마지막에 '?' 추가

FAISS_Flat_cosine	건민	0.5697 -	0.5742 -
↓			
FAISS_Flat_c...dd'?	건민	0.6727 -	0.6758 -

- 쿼리의 마지막에 '?'문자를 넣는 것 만으로 검색 정확도가 눈에 띄게 향상하는걸 우연히 확인함. MAP 점수가 0.5697-> 0.6727
- 특정한 표현이 문맥적 정보를 추가할 수 있음을 알게됨.  
이 경우에는 질의응답 task이기 때문에 쿼리가 대부분 질문이고 '?', '검색해줘' 라는 표현이 도움이 되었을 거라 추측함.

Q. standalone query가  
아니라 문서 자체를  
embedding하면 어떨까?



# HyDE(Hypothetical Document Embedding)



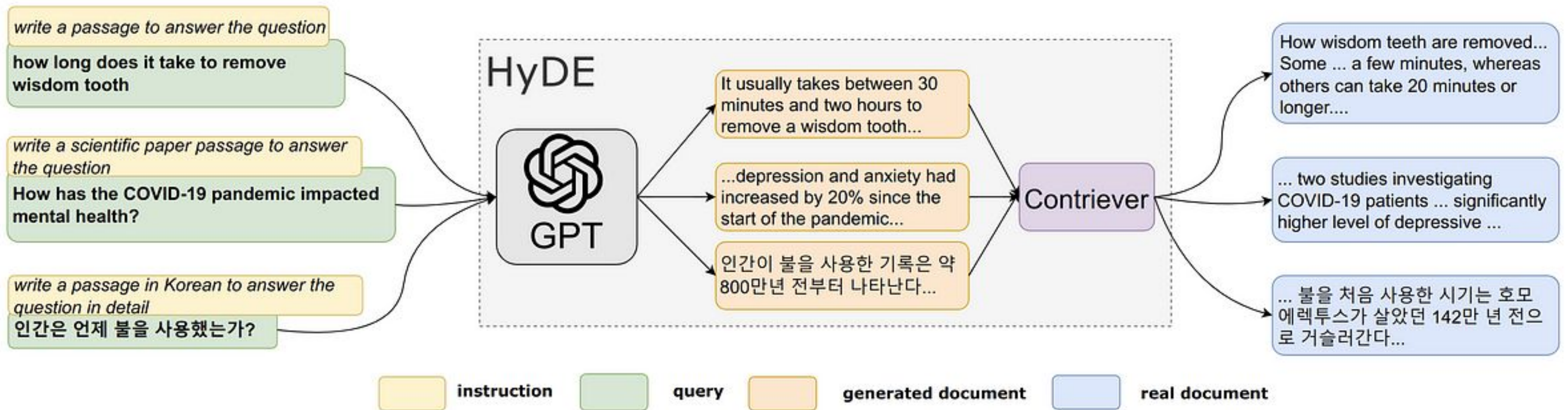


Figure 1: An illustration of the HyDE model. Documents snippets are shown. HyDE serves all types of queries without changing the underlying GPT-3 and Contriever/mContriever models.



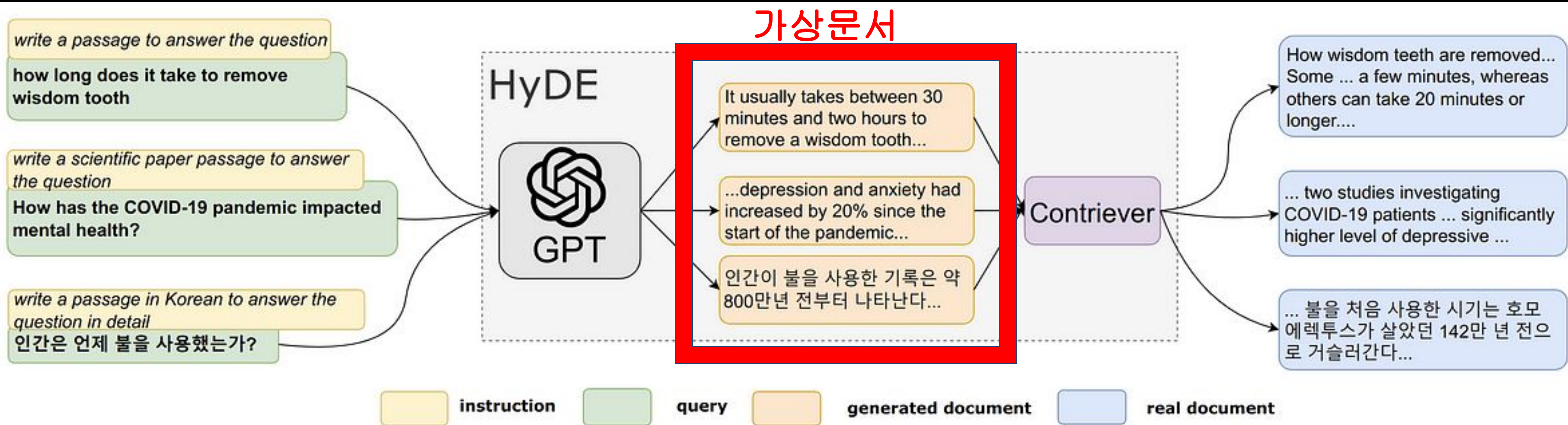


Figure 1: An illustration of the HyDE model. Documents snippets are shown. HyDE serves all types of queries without changing the underlying GPT-3 and Contriever/mContriever models.



# Standalone Query Generation

Hyde(Hypothetical Document Embeddings)

## HyDE (가상문서생성을 통한 검색) develop 과정

1. Query에 대한 가상 문서를 생성(by LLM). → 이 문서가 곧 standalone query의 기능을 함.

-> 기존 방식에 비해 오히려 낮은 정확도를 보임

<input type="checkbox"/>	HyDE	<div>건민</div>	0.7992	0.8030	2024.10.17 13:41	완료	<div>↓</div>
--------------------------	------	---------------	--------	--------	------------------	----	--------------



# Standalone Query Generation

Hyde(Hypothetical Document Embeddings)

## HyDE (가상문서생성을 통한 검색) develop 과정

2. 구체적인 instruction을 주어 가상 문서를 생성함.

-> 첫번째 시도보다 조금 개선되었으나 여전히 성능이 낮음.



hyde\_it



0.8189

-

0.8212

-

2024.10.17 18:02

완료



# Standalone Query Generation

Hyde(Hypothetical Document Embeddings)

## HyDE (가상문서생성을 통한 검색) develop 과정

3. 모델을 HyDE에 맞게 파인튜닝하여 가상문서를 생성함.

: LLM으로 모든 문서에 대한 질문을 생성하여 질문-문서 쌍을 만들었고 이를 활용해 gpt-4o-mini 모델을 파인튜닝함.

-> 만족스러운 결과가 나오지 않아서 결국 폐기



hyde\_finetuning



0.8114

0.8152

2024.10.18 16:34

완료





# Standalone Query Generation

Hyde(Hypothetical Document Embeddings)

---

## Q. 왜 성능이 나오지 않았을까?

- LLM이 생성한 가상문서는 기존의 standalone query보다 문서에 대해서 높은 유사도를 가지게 되었음.

- 그러나 관련없는 문서도 함께 유사도가 높아짐.

→ 즉, 정답에 해당하는 문서와의 유사도만 높아진 것이 아니라, 모든 문서와의 유사도 점수가 전반적으로 높아져서 변별력이 떨어짐.

(모든 문서에 대한 유사도 점수 평균은 상승하고, 표준편차는 떨어짐..)

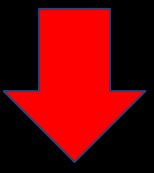
02-03

## 3. Embedding model 교체



# Embedding model 교체

Dense Retrieval을 위한 조치

baseline_dense	건민	0.4394 -	0.4409 -
			
진짜 upstage_api_dense	건민	0.8394 -	0.8455 -

- 동의어 문제를 해결하기 위해서는 Dense retrieve의 성능 향상이 반드시 필요했음
- 구글링으로 한국어 데이터에서 성능이 가장 높은 upstage 임베딩 모델을 발견하게 됨.
- 모델을 개선한 것 만으로 sparse retrieve보다 높은 성능을 확보함.

# Embedding model 교체

Dense Retrieval을 위한 조치

---

- API로 임베딩을 함으로서 메모리 문제 해결
- 4096 차원까지 인덱싱을 지원하도록 Elasticsearch의 버전을 업데이트
- 전체 문서를 인덱싱하는데 약 0.05달러 비용 발생  
(IF 오픈소스 모델을 fine-tuning 한다면 더 경제적 일 수 있음.)

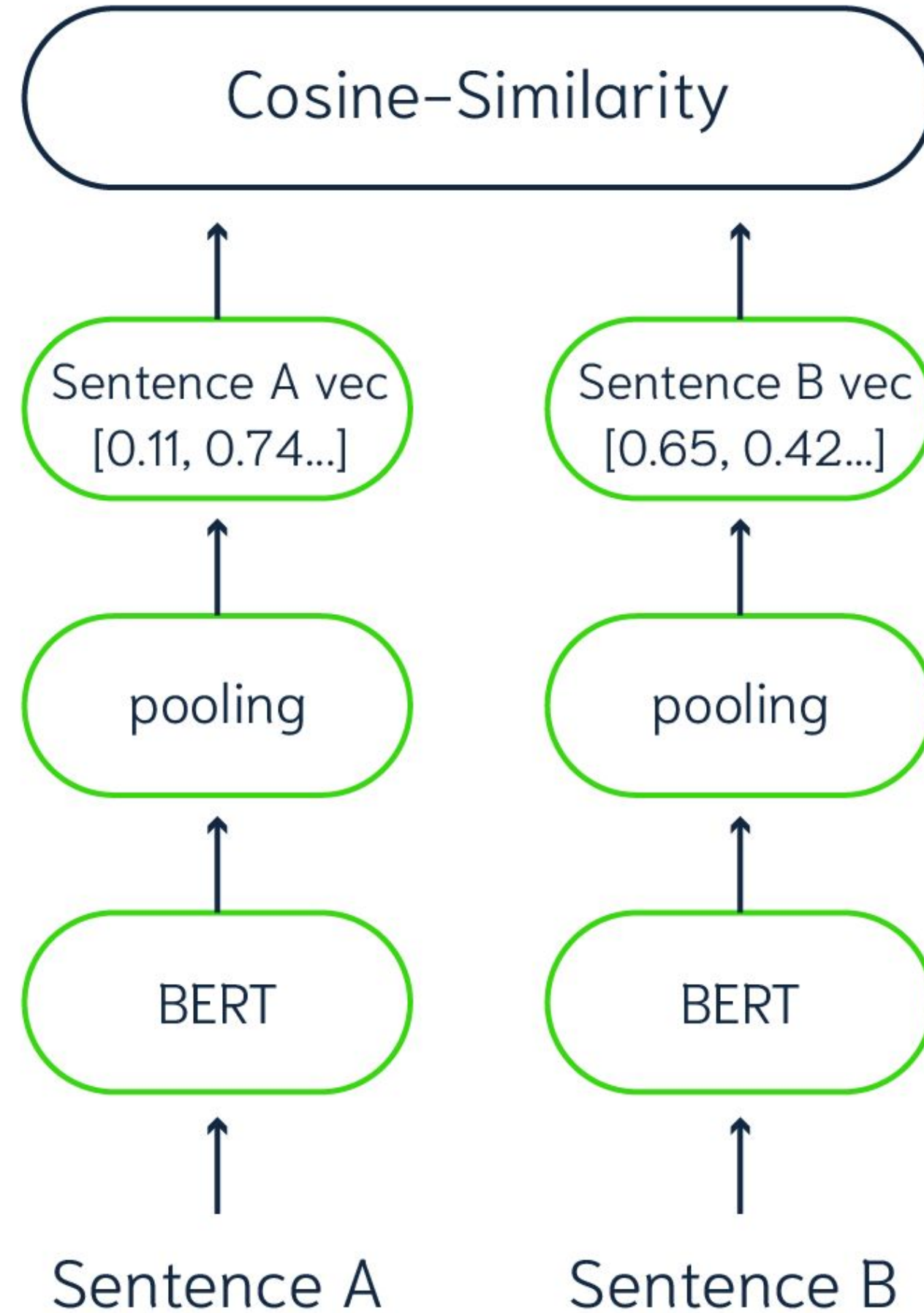


02-04

## 4. Topk re-rank



# Bi-Encoder

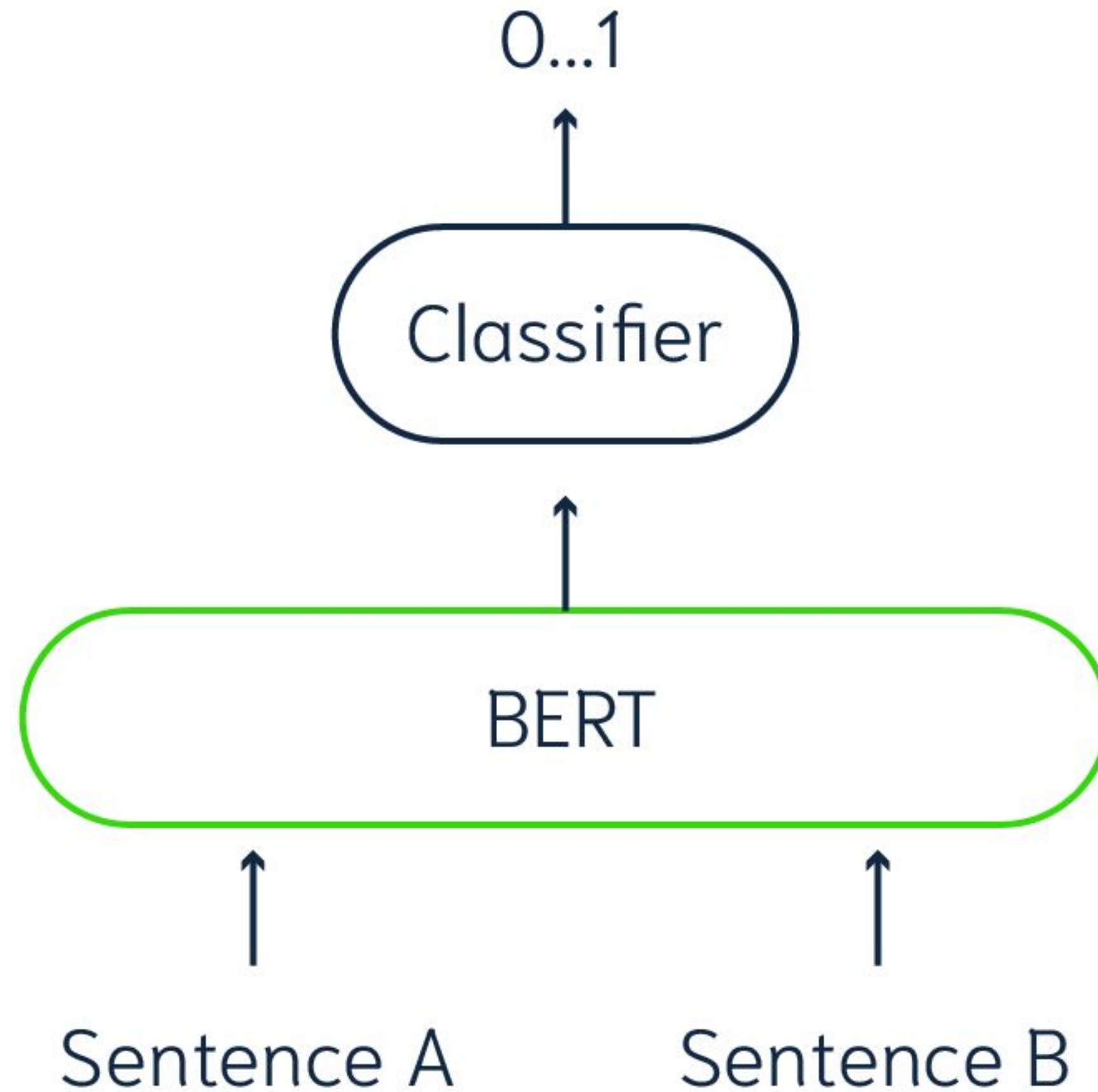




1. 이 방식으로 먼저 dense retrieval을  
진행하여 문서 10개 생성



# Cross-Encoder






## 2. 이 방식으로 10개의 문서를 re-rank한 뒤 상위 3개 추출

# Topk Re-rank

Reranker를 통한 성능 향상

## Rerank

upstage_api_dense수정	건민	0.8750	0.8758
		-	-
			
rerank	건민	0.9167	0.9212
		-	-

Rerank를 적용함으로  
MAP 0.8750 -> 0.9167로 향상



02-05

## 5. LLM을 GEMINI로 교체

# gemini api 코드 비교

## 1. open ai와 prompt를 이해하는 방식이 다름

---

```
{ "eval_id": 264, "standalone_query": "", "topk": [], "answer": "죄송하지만 성적으로 노골적인 주제에 대해서는 답변할 수 없습니다." }
```

- 증오심 표현, 괴롭힘, 음란물, 위험한 콘텐츠에 대한 규정이 **openai** 보다 명확하게 규정되어 **standalone\_query**를 못 찾아 옴. ('성적인 자극이 발생하는 과정을 알려 줘.')



# gemini api 코드 비교

## 1. open ai와 prompt를 이해하는 방식이 다름

```
# RAG 구현에 필요한 질의 분석 및 검색 이외의 일반 질의 대응을 위한 LLM 프롬프트
```

```
persona_function_calling = """
```

```
## Role: 검색을 사용할 것인지, 대화를 생성할 것인지 판단하는 역할
```

```
## Instruction
```

```
* **주요 기능:** 사용자의 질문을 분석하여 관련된 정보를 검색하고 지식에 관련된 질문은 searchapi를 사용하여 검색을 하고, 일상 질문은 바로 생성합니다.
```

```
* **검색 범위:** 인문학, 사회과학, 자연과학, 공학 등 모든 분야를 포괄합니다.
```

```
* **검색 기준:**
```

```
* **관련성:** 사용자의 질문과 가장 관련성이 높은 정보를 우선적으로 제공합니다.
```

```
* **판단기준** 성적인 암시인 경우 같은 부적절한 것 같은 경우도 스스로 판단하지 말고, 검색이 가능하면 검색을 한다. "우울하다" 같이 감정을 나눈다면 적절한 대답을 한국말로 생성한다.
```

```
* **응답 형식:**
```

```
* **요약:** 검색 결과를 요약하여 간결하게 제공합니다.
```

```
"""
```

- 판단기준을 정해주어, GEMINI가 직접 판단하여 필터링하는 현상을 개선함.

# gemini api 코드 비교

## 1. open ai와 prompt를 이해하는 방식이 다름

```
# RAG 구현에 필요한 질의 분석 및 검색 이외의 일반 질의 대응을 위한 LLM 프롬프트
```

```
persona_function_calling = ""
```



gemini\_upsta...dense



0.7750

-

0.7773

-

2024.10.15 22:52

완료



```
***주요 기능:** 사용자의 질문을 분석하여 관련된 정보를 검색하고, 검색에 관련된 질문은 searchapi를 사용하여 검색을 하고, 검색 결과는 바로 제공합니다.  
***검색 범위:** 인문학, 사회과학, 자연과학, 공학 등 모든 분야를 포괄합니다.  
***검색 기준:**
```



gpt4o\_upstag...rompt



0.8773

-

0.8833

-

2024.10.18 17:54

완료



```
... **요약:** 검색 결과를 요약하여 간결하게 제공합니다.  
""
```

- 판단기준을 정해주어, GEMINI가 직접 판단하여 필터링하는 현상을 개선함.
- 성능 MAP점수 0.8773으로 나쁘지 않은 성능을 보여주었으나, 시간 문제로 더 develop하지 못함 ㅠㅠ



# gemini api 코드 비교

## 2. open ai와 tools를 작성하는 방식이 다름

```
tools = {  
  "name": "search",  
  "description": "관련 문서를 검색합니다. 캐주얼한 대화를 제외한 모든 질문이나 요청 시 이 함수를 호출하세요. 예: '지구 자전의 원인은?', '세종대왕에 대해 알려줘.'",  
  "parameters": {  
    "type_": "OBJECT",  
    "properties": {  
      "standalone_query": {  
        "type_": "STRING",  
        "description": "사용자 메시지 기록을 기반으로 문서 검색에 적합한 최종 쿼리. 항상 한국어로 작성하세요."  
      }  
    }  
  },  
  "required": ["standalone_query"]  
}
```

“type\_”: “STRING” 과 “type\_”: “OBJECT”: tools를 정의하는 방식도 달랐다.  
찾느라 고생

## gemini api 코드 비교

3. open ai와 role을 정의하는 방식도 다르다.

---

```
"role": "model", "parts": "네 맞습니다."},
```

content는 parts로 role:assistant는 role:model로 바꿔주어야한다.



# gemini api 코드 비교

## 4. start\_chat과 send\_message

```
response = {"standalone_query": "", "topk": [], "references": [], "answer": ""}
j = json.loads(line)
last_j= j['msg'].pop()['parts']
chat = llm_model.start_chat(
    history=j['msg']
)
try:
    result = chat.send_message(last_j+persona_function_calling)
```

openai는 client.messages.create로 text\_generation을 만들지만, gemini는 start\_chat과 send\_message형태로 나누어 text를 generate한다.



03

# 결과 및 인사이트 공유



# 프로젝트 인사이트 공유

: RAG 시스템

- (인웅) 프롬프트가 이해할 수 있도록 상세한 정보를 적는 것이 LLM모델에 성능에 가장 중요한 요인인 것 같다.
- (범희) 1) Validation set의 필요성을 느꼈다. 모델 성능을 개선하기 위해서 원인 파악이 가장 중요한데, validation이 이루어지지 않다 보니 진단을 내리기가 어려웠다. 2) LLM을 활용함에 있어 프롬프트가 매우 중요했다. zero-shot 보다는 one-shot, few-shot처럼 예시를 제시하여야 더 안정적인 기능을 했다. 또한 '반드시', '항상'과 같은 강조용 단어가 들어가면 더 안정적으로 작동했다.
- (건민) 정확도와 비용의 균형을 잘 고려해야한다. RAG시스템의 모든 부분이 비용과 밀접하게 관련되어 있었다. vector store, LLM, embedding모델 등, 서비스의 목적에 최적화하여 구성해야한다.

04

# 아쉬운 점 및 질문



# 프로젝트 아쉬운 점 및 질문들

## : Scientific Knowledge Answering(RAG)

---

1. Q. ‘?’가 ‘검색해줘’ 등이 standalone query에 포함되면 성능이 향상되는 이유가 무엇인가?

→ 이미 문장에서 의문문임을 파악할 수 있는 문맥 정보가 충분히 있을텐데, ‘?’는 무슨 기능을 하는 걸까?

2. BM25보다 dense retrieval이 더 좋은 성능을 보이는 것 같다. 그럼에도 BM25를 사용하는 이유는?

→ 현실에서 서비스할 때는 속도도 무시할 수 없기 때문에 빠른 속도의 BM25를 사용하는 것으로 예상된다.

3. RAG 시스템의 특성상, 각 도메인에 전문화되도록 embedding 모델에 MoE패턴 적용이 효율적일 것 같고 구현해보고싶다.

Life-Changing Education

감사합니다.

---