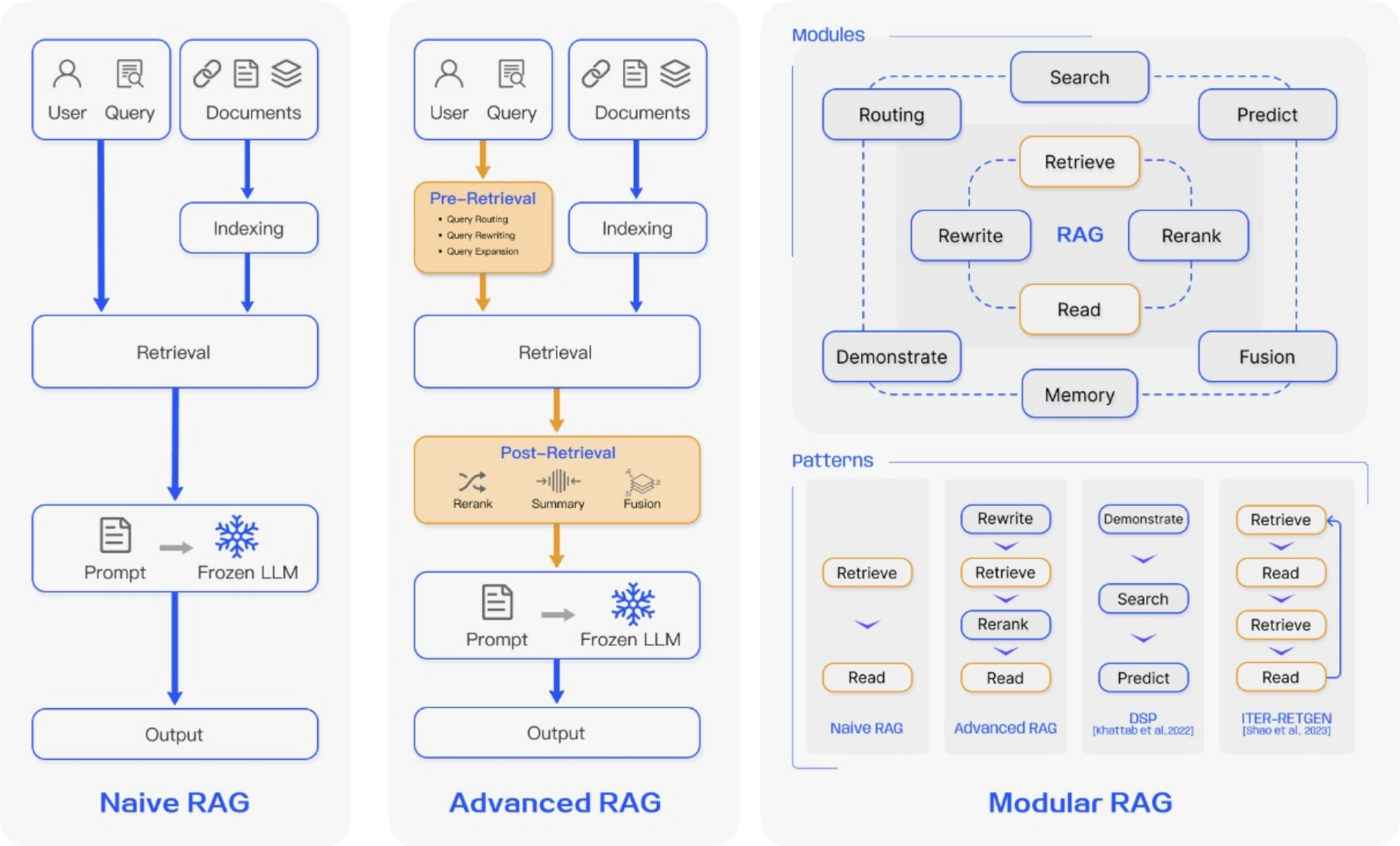
Advanced RAG

2024 하계세미나

장영준

RAG Paradigms



Retrieval-Augmented Generation for Large Language Models: A Survey(2023)

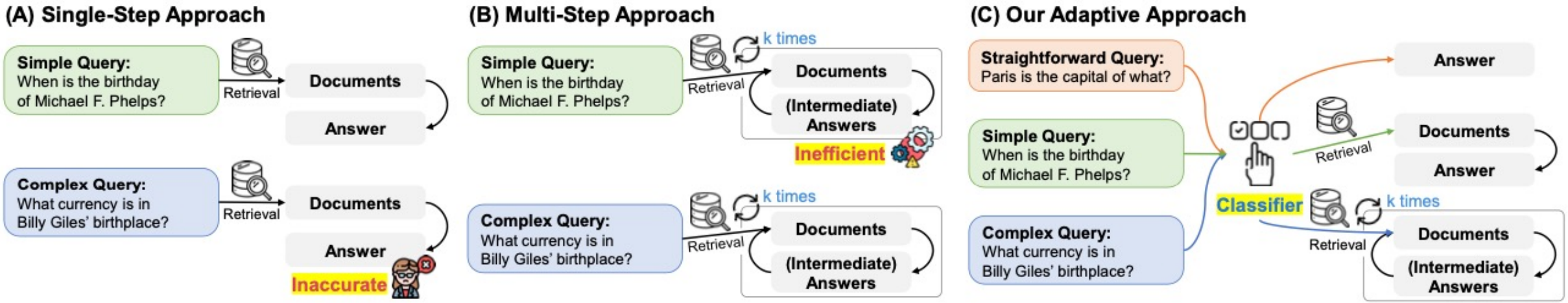
Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity

Soyeong Jeong[1] **Jinheon Baek**[2] **Sukmin Cho**[1] **Sung Ju Hwang**[1,2] **Jong C. Park**[1*]
School of Computing[1] Graduate School of AI[2]
Korea Advanced Institute of Science and Technology[1,2]
{starsuzi,jinheon.baek,nelllpic,sjhwang82,jongpark}@kaist.ac.kr

NAACL 2024

Introduction

Problem?



- 간단한 query에 복잡한 retrieve 방식을 쓰는 것은 너무 비효율적이다.
- QA task에서 query의 복잡도에 따른 분류가 필요하다
  - Query가 매우 간단 -> Retrieve 필요 X
  - Query가 적당히 간단 -> 하나의 문서만 retrieve
  - Query가 복잡 -> 여러 개의 문서 retrieve

Contribution

- **High Performance & Efficiency**

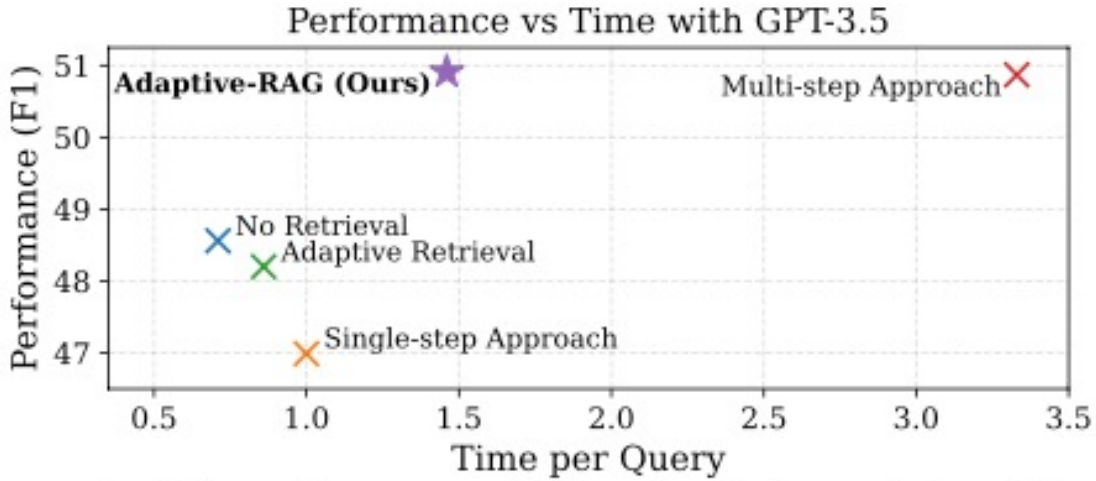


Figure 1: QA performance (F1) and efficiency (Time/Query) for different retrieval-augmented generation approaches. We use the GPT-3.5-Turbo-Instruct as the base LLM.

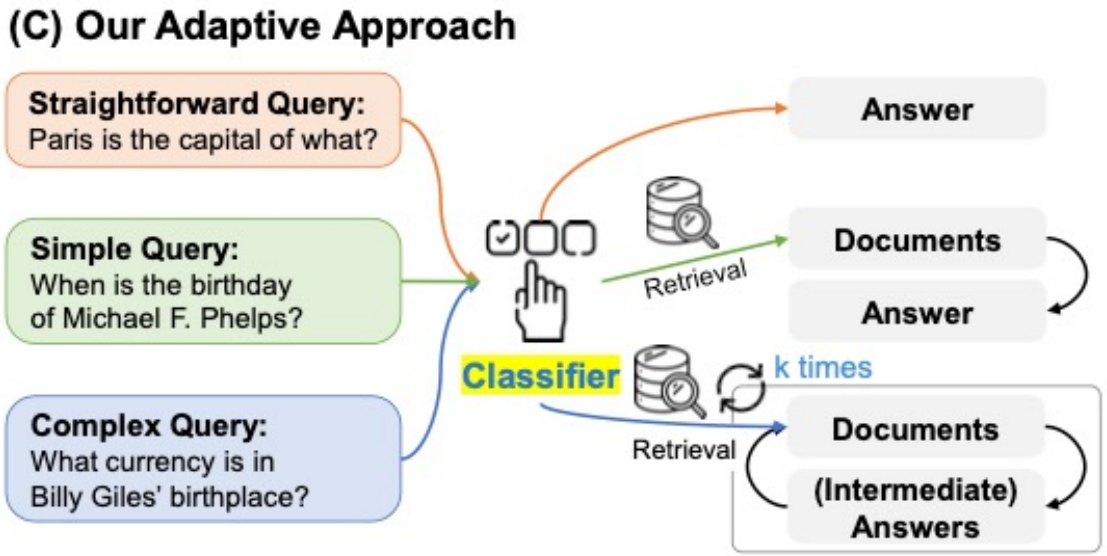
Key points of Adaptive-RAG

Query classifier

- Query의 복잡도를 판단하는 classifier를 어떻게 train 했는지

Accuracy & Efficiency

- Classifier를 이용함으로써 open-domain QA task에서 RAG의 정확성, 효율성을 얼마나 높였는지



Methods

1. Training classifier – Overall

1. 데이터를 아래와 같이 라벨링
    - **Straightforward Query**: Non-retrieval QA method -> **A**
    - **Simple Query**: Single-step approach for QA -> **B**
    - **Complex Query**: Multi-step approach for QA -> **C**
2. Labeled data를 이용하여 classifier를 학습
3. Output: classifier가 query를 보고 A, B, C 중 무엇에 해당하는지 판단

Methods

1. Training classifier – Data Labeling

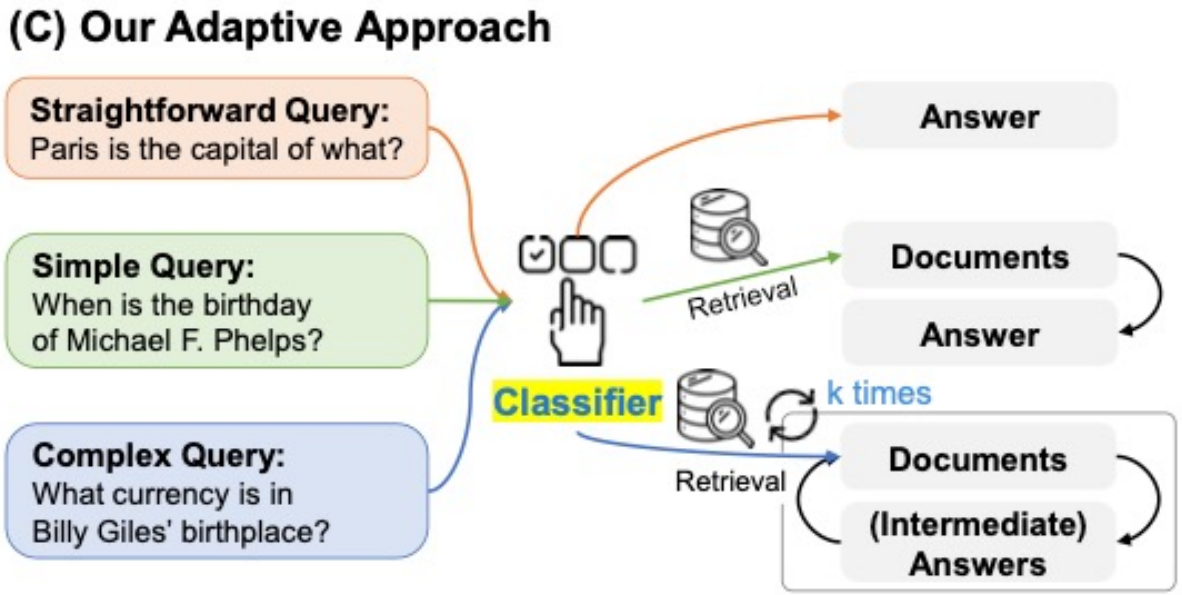
Method	Data
Single-hop QA	SQuAD v1.1
	NQ
	TriviaQA
Multi-hop QA	MuSiQue
	HotpotQA
	2WikiMultiHopQA

- Dataset: 실험 데이터셋에서 400개의 query sampling (실험 데이터와 overlap X)
- **Automated labeling strategy**
  1. **Generating silver data from predicted outcomes of models**
     - 먼저 생성 model에게 query를 주고 답을 생성해보도록 함
     - Model이 아무 문서도 참조하지 않고 생성한 답이 원래 답과 일치하면 A
     - A로 라벨링되지 않은 데이터 중, 하나의 문서만 참조해서 생성한 결과가 답과 일치하면 B
     - A,B로 라벨링되지 않은 데이터 중, 여러 문서를 참조해서 생성한 결과가 답과 일치하면 C
  2. **Inductive Biases in dataset**
     - 앞의 과정을 거치고도 label되지 않은 데이터에 대해서는, 데이터의 inductive bias를 이용
     - 출처가 Single-hop QA 데이터면 B, Multi-hop QA 데이터면 C로 라벨링

Methods

2. Inference

- 최종 추론 시
  1. Query의 complexity 판단
  2. 각 complexity에 맞춘 추론 방법 적용



Experiment Data

Datasets

1. Eval dataset
    - Single-hop QA
        - **SQuAD v1.1** – Questions created by annotators based on documents they read
        - **Natural Questions** – Constructed by user queries on Google Search
        - **TriviaQA** – Trivia questions sourced from various quiz sites
    - Multi-hop QA
        - **MuSiQue** – queries needing 2~4 hops
        - **HotpotQA** – Questions created by annotators that link multiple Wikipedia articles
        - **2WikiMultiHopQA** – derived from Wikipedia, needing 2-hops
2. Eval metric
    - **Effectiveness**: F1, EM, Acc
    - **Efficiency**: # of retrieval-and-generate steps, average time for answering each query

Experiment Data

Implementation Details

- Retriever: BM25
- External Corpus (RAG Database)
  - Single-hop: Pre-processed Wikipedia corpus for single-step questions (DPR)
  - Multi-hop: Pre-processed Wikipedia corpus for multi-step questions (IRCoT)
- LLMs
  - FLAN-T5 series (XL-3B, XXL-11B)
  - gpt-3.5-turbo-instruct
- Classifier
  - T5-Large model train
  - 100 epoch의 학습 동안 validation dataset에 대해 가장 높은 성능을 보인 epoch의 model

Experiment Data

Baselines

1. No Retrieval
2. Single-step Approach
    - 모든 query에 대해 무조건 1번 retrieve
3. Adaptive Retrieval
    - Query에 따라 retrieve 할지 말지, binary로 판단
4. Self-RAG (SOTA)
5. Adaptive-RAG
6. Multi-step Approach
    - 모든 query에 대해 무조건 여러번 retrieve
7. Adaptive-RAG w/ Oracle (complexity classification accuracy 100%)

Experiments

Overall Results

Types	Methods	FLAN-T5-XL (3B)					FLAN-T5-XXL (11B)					GPT-3.5 (Turbo)				
		EM	F1	Acc	Step	Time	EM	F1	Acc	Step	Time	EM	F1	Acc	Step	Time
Simple	No Retrieval	14.87	21.12	15.97	0.00	0.11	17.83	25.14	19.33	0.00	0.08	35.77	48.56	44.27	0.00	0.71
	Single-step Approach	34.83	44.31	38.87	1.00	1.00	37.87	47.63	41.90	1.00	1.00	34.73	46.99	45.27	1.00	1.00
Adaptive	Adaptive Retrieval	23.87	32.24	26.73	0.50	0.56	26.93	35.67	29.73	0.50	0.54	35.90	48.20	45.30	0.50	0.86
	Self-RAG*	9.90	20.79	31.57	0.72	0.43	10.87	22.98	34.13	0.74	0.23	10.87	22.98	34.13	0.74	1.50
	Adaptive-RAG (Ours)	**37.17**	**46.94**	**42.10**	**2.17**	**3.60**	**38.90**	**48.62**	**43.77**	**1.35**	**2.00**	**37.97**	**50.91**	**48.97**	**1.03**	**1.46**
Complex	Multi-step Approach	39.00	48.85	43.70	4.69	8.81	40.13	50.09	45.20	2.13	3.80	38.13	50.87	49.70	2.81	3.33
Oracle	Adaptive-RAG w/ Oracle	45.00	56.28	49.90	1.28	2.11	47.17	58.60	52.20	0.84	1.10	47.70	62.80	58.57	0.50	1.03

- Self-RAG에 한해서는 생성 모델을 finetuned LLaMA2-7B, LLaMA2-13B로 변경
- 결과 분석
  1. Simple 방식이 Complex 방식에 비해 성능이 떨어지지만, 효율적임 (당연)
  2. Adaptive-RAG > Adaptive Retrieval: 단순히 retrieve 할지 말지 판단하는건 크게 효과 없음
  3. Adaptive-RAG > Self-RAG

Experiments

Overall Results – Single hop QA

Table 2: Results on each of a collection of datasets with FLAN-T5-XL (3B) as the LLM. We emphasize our results in bold.

			SQuAD					Natural Questions					TriviaQA				
Data	**Types**	**Methods**	EM	F1	Acc	Step	Time	EM	F1	Acc	Step	Time	EM	F1	Acc	Step	Time
Single-step	**Simple**	**No Retrieval**	3.60	10.50	5.00	0.00	0.11	14.20	19.00	15.60	0.00	0.13	25.00	31.80	27.00	0.00	0.13
		Single-step Approach	27.80	39.30	34.00	1.00	1.00	37.80	47.30	44.60	1.00	1.00	53.60	62.40	60.20	1.00	1.00
	Adaptive	**Adaptive Retrieval**	13.40	23.10	17.60	0.50	0.55	28.20	36.00	33.00	0.50	0.56	38.40	46.90	42.60	0.50	0.56
		Self-RAG*	2.20	11.20	18.40	0.63	0.50	31.40	39.00	33.60	0.63	0.17	12.80	29.30	57.00	0.68	0.45
		Adaptive-RAG (Ours)	**26.80**	**38.30**	**33.00**	**1.37**	**2.02**	**37.80**	**47.30**	**44.60**	**1.00**	**1.00**	**52.20**	**60.70**	**58.20**	**1.23**	**1.54**
	Complex	**Multi-step Approach**	24.40	35.60	29.60	4.52	9.03	38.60	47.80	44.20	5.04	10.18	53.80	62.40	60.20	5.28	9.22
	Oracle	**Adaptive-RAG w/ Oracle**	32.00	45.60	38.20	1.24	1.60	47.40	57.10	53.60	1.10	1.55	61.60	70.20	66.40	0.79	1.10

1. Effectiveness: **Complex Strategy <= Adaptive-RAG <= Simple Strategy**
2. Efficiency: **Complex Strategy < Adaptive-RAG < Simple Strategy**
3. Adaptive-RAG w/ Oracle classifier -> Best Performance: **Query Complexity Classify**의 중요성

Experiments

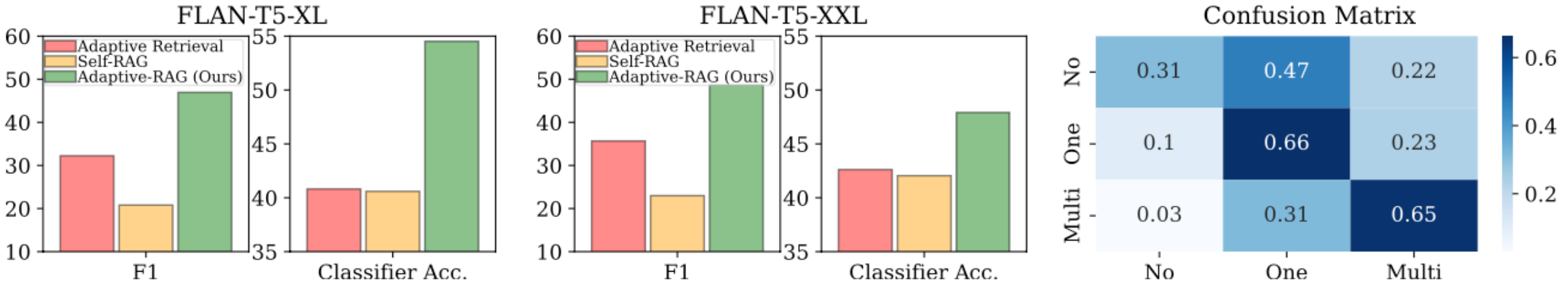
Overall Results – Multi hop QA

Data	Types	Methods	MuSiQue					HotpotQA					2WikiMultiHopQA				
			EM	F1	Acc	Step	Time	EM	F1	Acc	Step	Time	EM	F1	Acc	Step	Time
Multi-step	Simple	No Retrieval	2.40	10.70	3.20	0.00	0.11	16.60	22.71	17.20	0.00	0.11	27.40	32.04	27.80	0.00	0.10
		Single-step Approach	13.80	22.80	15.20	1.00	1.00	34.40	46.15	36.40	1.00	1.00	41.60	47.90	42.80	1.00	1.00
	Adaptive	Adaptive Retrieval	6.40	15.80	8.00	0.50	0.55	23.60	32.22	25.00	0.50	0.55	33.20	39.44	34.20	0.50	0.55
		Self-RAG*	1.60	8.10	12.00	0.73	0.51	6.80	17.53	29.60	0.73	0.45	4.60	19.59	38.80	0.93	0.49
		Adaptive-RAG (Ours)	**23.60**	**31.80**	**26.00**	**3.22**	**6.61**	**42.00**	**53.82**	**44.40**	**3.55**	**5.99**	**40.60**	**49.75**	**46.40**	**2.63**	**4.68**
	Complex	Multi-step Approach	23.00	31.90	25.80	3.60	7.58	44.60	56.54	47.00	5.53	9.38	49.60	58.85	55.40	4.17	7.37
	Oracle	Adaptive-RAG w/ Oracle	24.80	38.50	27.00	1.98	3.99	51.20	64.00	54.80	1.59	2.77	53.00	62.30	59.40	1.01	1.69

1. Effectiveness: **Simple Strategy < Adaptive-RAG <= Complex Strategy**
2. Efficiency: **Complex Strategy < Adaptive-RAG < Simple Strategy**
3. Adaptive-RAG w/ Oracle classifier -> Best Performance: **Query Complexity Classify**의 중요성

Experiments

Classifier Performance



Gold label은 앞서 labeling 한 것처럼 모든 데이터셋에 대해 silver labeling + inductive bias 거침

1. Adaptive Retrieval, Self-RAG 둘다 처음에 query를 보고 retrieve 할지 말지 결정
2. Adaptive-RAG의 Classifier가 가장 정확했다 -> QA의 F1 score 정확도까지 이어짐
3. Confusion Matrix
    - “Multi”로 예측되어야 할 query들이 가끔 “Single”로 예측되기도 하고 (31%)
    - “Single”으로 예측되어야 할 query들이 ”Multi”로 예측되기도 함 (23%)
    - **“No”가 가장 misclassified 될 확률이 높음**

Experiments

Analyses on Training Data for Classifier

Training Strategies	QA		Classifier (Accuracy)			
	F1	Step	All	No	One	Multi
Adaptive-RAG (Ours)	**46.94**	**1084**	**54.52**	**30.52**	**66.28**	**65.45**
w/o Binary	43.43	640	60.30	62.19	65.70	39.55
w/o Silver	48.79	1464	40.00	0.00	53.98	75.91

- Silver – model을 사용해서 데이터 라벨링한 방법
- Binary - 데이터의 출처 (single-hop or multi-hop)로 라벨링한 방법
- **Binary가 없는 경우**, classifier의 overall 성능은 좋아지지만, F1 score가 낮아짐
- **Silver가 없는 경우**, Classifier의 "No" (Retrieve 안해도 되는 경우)에 대한 정확성이 0임
  - Classifier의 정확성이 안좋아지고, QA task 수행 시 efficiency 감소
- 둘다 사용하는 방법이 정확도와 효율을 모두 챙길 수 있는 방법이다!

Experiments

Analyses on Classifier Size

	QA		Classifier (Accuracy)			
Sizes	**F1**	**Step**	**All**	**No**	**One**	**Multi**
Small (60M)	45.83	964	53.48	26.65	70.62	53.18
Base (223M)	45.97	983	53.41	26.42	69.46	56.82
Large (770M)	**46.94**	**1084**	**54.52**	30.52	66.28	65.45

크기가 커져도 크게 정확성에 차이는 없음 -> classifier를 resource-efficient setting에서도 잘 사용할 수 있다 !

Conclusion

- 현실적인 상황을 가정하여 Efficient RAG pipeline을 만들었다는 점에서 좋은 contribution
- 아쉬운 점
  - 진짜 real-world scenario를 가정할 것이었으면 train된 classifier를 다른 데이터셋에도 적용해 보면 좋았을 것 같다.
  - Self-RAG와 다른 retriever를 사용 (전략?) -> Self-RAG에서 쓴 Contriever-MS-Marco를 썼으면 결과가 어땠을까..

Data	Types	Methods	TriviaQA EM	TriviaQA F1	TriviaQA Acc	TriviaQA Step	TriviaQA Time
Single-step	Simple	No Retrieval	25.00	31.80	27.00	0.00	0.13
		Single-step Approach	53.60	62.40	60.20	1.00	1.00
	Adaptive	Adaptive Retrieval	38.40	46.90	42.60	0.50	0.56
		Self-RAG*	12.80	29.30	57.00	0.68	0.45
		Adaptive-RAG (Ours)	**52.20**	**60.70**	**58.20**	**1.23**	**1.54**
	Complex	Multi-step Approach	53.80	62.40	60.20	5.28	9.22
	Oracle	Adaptive-RAG w/ Oracle	61.60	70.20	66.40	0.79	1.10

LM	Short-form PopQA (acc)	Short-form TQA (acc)
Toolformer*$_{6B}$	–	48.8
Llama2$_{7B}$	38.2	42.5
Alpaca$_{7B}$	46.7	64.1
Llama2-FT$_{7B}$	48.7	57.3
SAIL*$_{7B}$	–	–
Llama2$_{13B}$	45.7	47.0
Alpaca$_{13B}$	46.1	66.9
Our SELF-RAG $_{7B}$	54.9	66.4
Our SELF-RAG $_{13B}$	**55.8**	**69.3**

ARAGOG: Advanced RAG Output Grading

Matouš Eibich
Predli
matous@predli.com

Shivay Nagpal
Predli
shivay@predli.com

Alexander Fred-Ojala
Predli & UC Berkeley
afo@berkeley.edu

April 2, 2024

Arxiv 2024.04

Introduction

ARAGOG

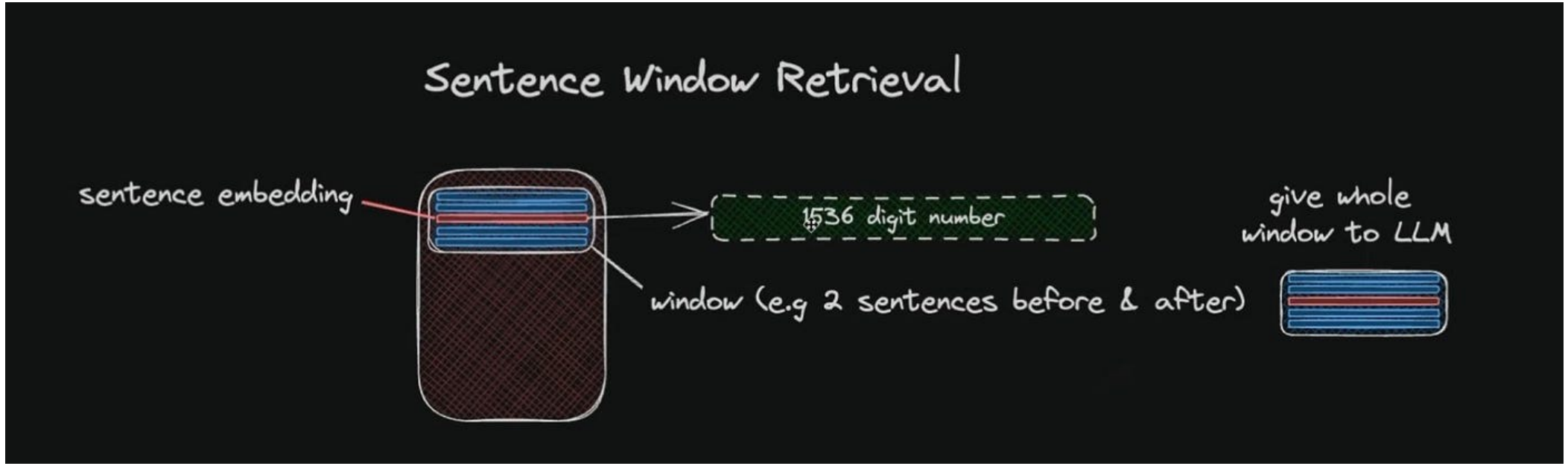
1. 7가지 다양한 RAG Techniques 소개

2. 각 RAG Techniques들을 평가 후 분석

RAG Technique	Type
Sentence-window retrieval	Decoupling of Retrieval and Generation
Document summary index	
HyDE	Query Expansion
Multi-query	
Maximal Marginal Relevance (MMR)	Enhancement Mechanism
Cohere Re-ranker	Re-rankers
LLM-based Re-ranker	

RAG Techniques

1. Sentence-window retrieval



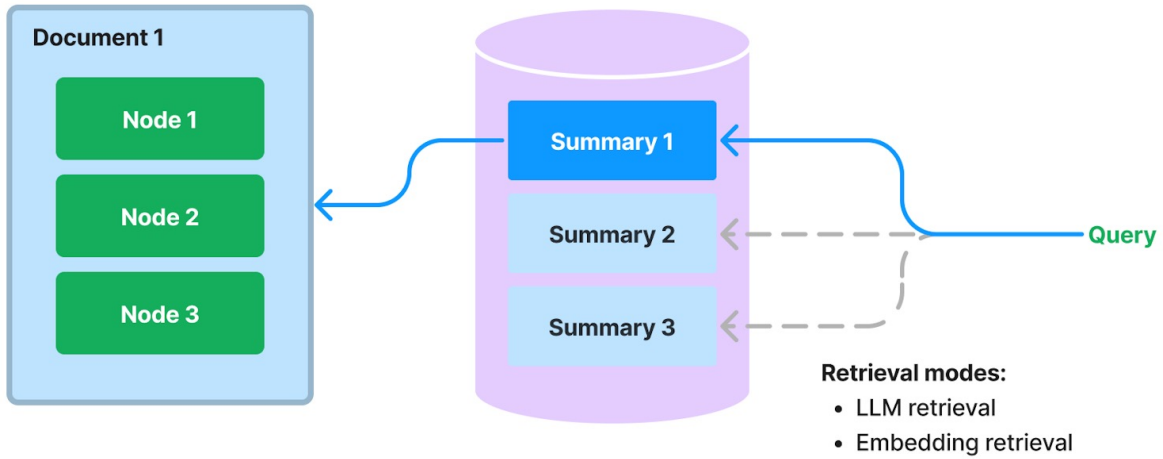
"Retrieve 단계는 간단하게, Generation 단계는 복잡하게"

1. Text를 개별 sentence 단위로 나누어 retrieve에 활용
2. 검색된 (선택된) sentence 주변에 window size 만큼의 추가적인 문장들을 가져와 generation에 사용

- 장점: 더 작은 정보로 정확하게 retrieve 하여, LLM에게 더 많은 정보 주입 가능

RAG Techniques

2. Document Summary Index

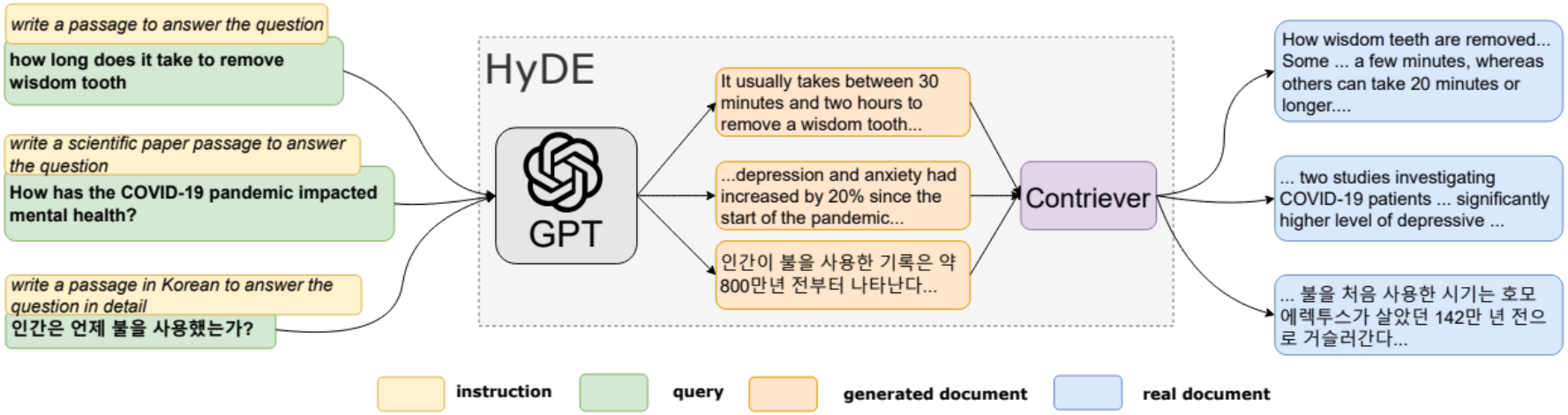


"Document summary를 활용한 retrieve"

1. LLM을 사용해서 각 document의 summary를 추출하여 DB에 저장
2. Document를 chunking하여 하나의 node 단위로 저장
3. query와 document summary의 유사도 계산해서, **가장 유사한 summary**를 1차 retrieve
4. Query와 node (text chunk)의 유사도 계산해서 **가장 유사한 node** 최종 retrieve

RAG Techniques

3. HyDE (Hypothetical Document Embedding) (2023 ACL)

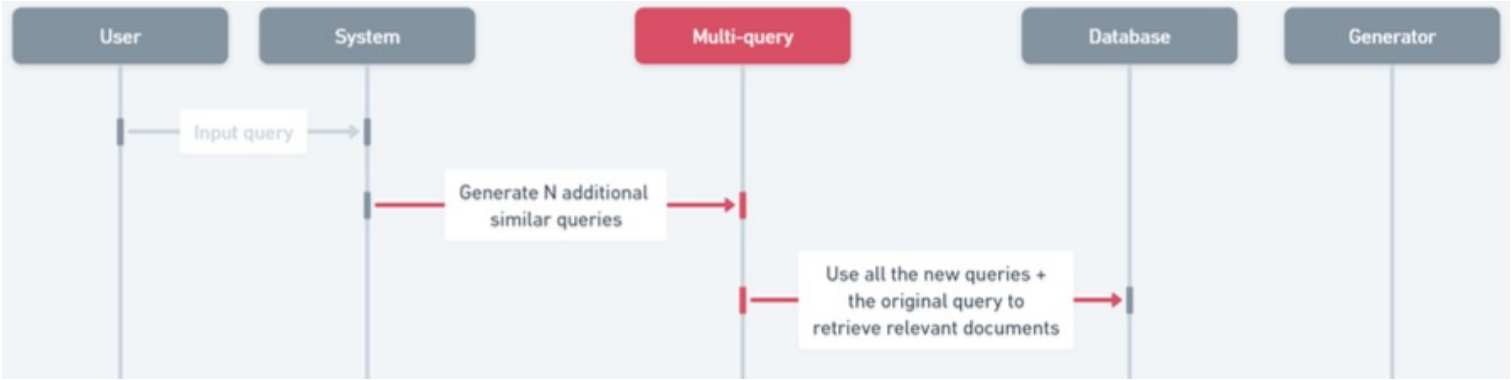


"Document-Query가 아닌, Document-{생성문들의 평균 벡터}의 유사도 비교"

1. query와 query에 대한 instruction을 LLM에게 주어 해당 query에 대한 N개의 문서/답변들을 생성하도록 함
2. 생성된 N개의 문서 각각의 임베딩과 query의 임베딩을 종합해서 평균을 구함 = v
3. v와 documents의 유사도를 기반으로 retrieve

RAG Techniques

4. Multi-Query



"Original query를 여러 개의 new queries로 나눈다"

1. Original query -> LLM을 사용하여 여러 개의 queries 생성

   ex. **Who won a championship more recently, the Red Sox or the Patriots?**

   - "When was the last time the Red Sox won a championship?"
   - "When was the last time the Patriots won a championship?"

2. 각각의 query들로 retrieve 하여, retrieve된 문서들 합침

RAG Techniques

5. Maximum Marginal Relevance (MMR)

$$\text{MMR} = \lambda \cdot \text{Sim}(d, Q) - (1 - \lambda) \cdot \max_{d' \in D'} \text{Sim}(d, d')$$

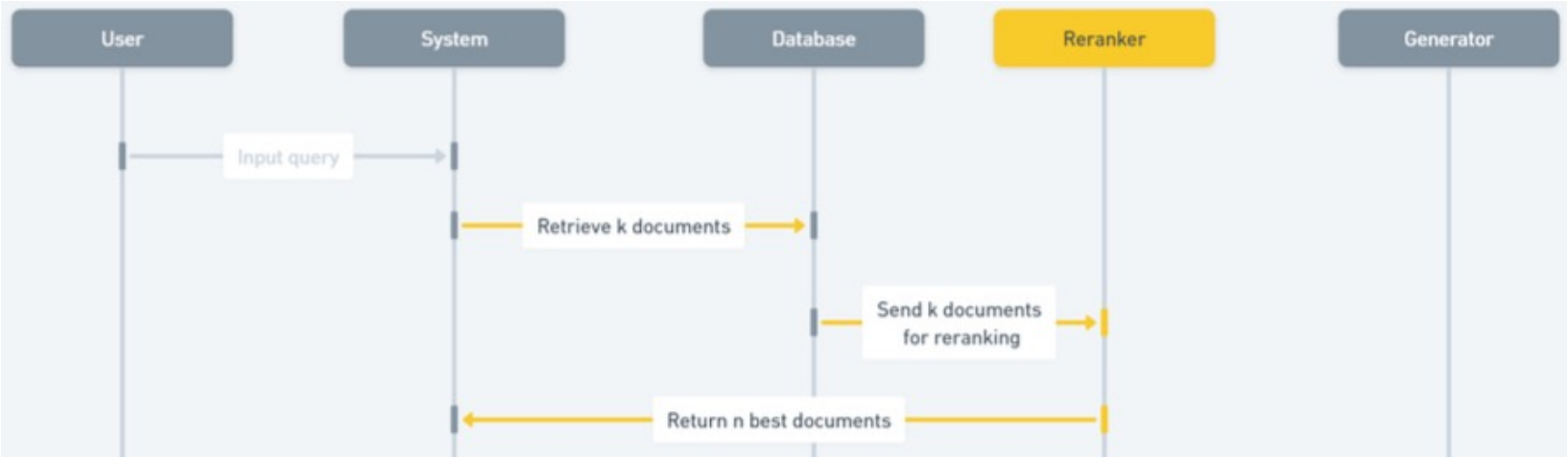
"Retrieve되는 documents의 중복 방지"

1. Query에 대한 유사도 점수와, 이미 선택된 documents 사이의 유사도 점수를 고려하여, 최종 점수 계산
2. 유사도 점수가 높은 documents를 retrieve

- 식 설명
  - Sim(d,Q)는 document와 Query와의 유사성
  - max Sim(d,d')은 이미 선택된 docs와 선택하려는 doc의 유사성
  - Query와 document는 유사하면서, 이미 retrieve한 documents와는 유사하지 않도록 (중복되지 않도록)

RAG Techniques

6. Cohere Rerank



"Retrieve Again"

1. Query를 기준으로 유사한 k개의 documents (chunks) retrieve
2. (Cohere) Reranker가 다시 k개의 documents에 대해 query와의 유사성을 판단하여 top n개의 documents만 남김

RAG Techniques

7. LLM Rerank

```
A list of documents is shown below. Each document has a number next to it along with a su
  Respond with the numbers of the documents you should consult to answer the question, in
  as the relevance score. The relevance score is a number from 1–10 based on how relevant
  Do not include any documents that are not relevant to the question.
  Example format:
  Document 1:
  <summary of document 1>
  Document 2:
  <summary of document 2>
  …
  Document 10:
  <summary of document 10>
  Question: <question>
  Answer:
  Doc: 9, Relevance: 7
  Doc: 3, Relevance: 4
  Doc: 7, Relevance: 3
  Let's try this now:
  {context_str}
  Question: {query_str}
  Answer:
```

“More cost, more accurate rerank”

- 앞서 reranking했던 것을 LLM을 사용하여 진행

RAG Techniques

정리

RAG Technique	Type
Sentence-window retrieval	Decoupling of Retrieval and Generation
Document summary index	
HyDE	Query Expansion
Multi-query	
Maximal Marginal Relevance (MMR)	Enhancement Mechanism
Cohere Re-ranker	Re-rankers
LLM-based Re-ranker	

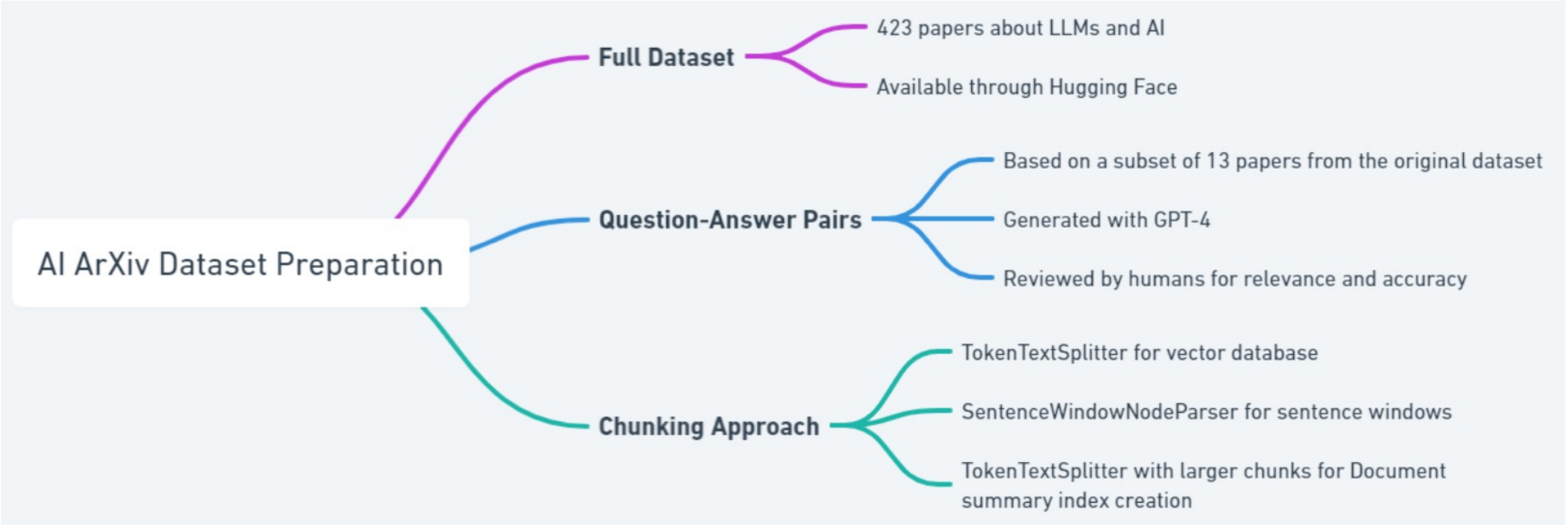
Experiment

Data

1. Data: 423개의 AI / LLM 분야 논문
2. RAG DB Construction
    - Query는 오직 13개의 key 논문에 대해서만 물어보는 query임
    - 나머지 410개는 noise (real-world와 비슷하게 하기 위함)
3. Chunking
    - Classic VDB: TokenTextSplitter (chunk size of 512 tokens & overlap of 50 tokens)
    - Sentence-window Retrieval: SentenceWindowNodeParser (window_size=3 sentences)
    - Document-summary-index: TokenTextSplitter(chunk size of 3072 tokens & overlap of 100 tokens)
        - Summarize 해야하므로 더 긴 길이 필요

Experiment

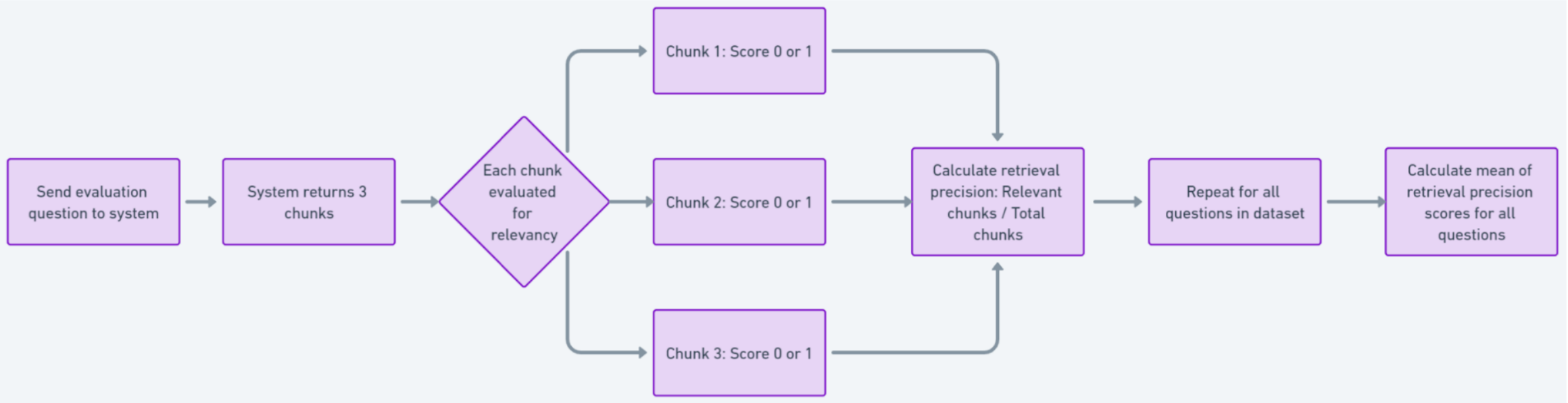
Data



- Evaluation Data Preparation
  - 13개의 key papers에 대해 GPT-4로 만들어진 107개의 QA pairs
  - 이후 human review로 relevance, accuracy 검토

Experiment

Metrics – Retrieval Precision



1. Retrieval Precision: Directly measuring efficacy of the retrieval system

- LLM에게 Retrieved chunks가 question에 얼마나 relevant한지 0~1 사이 점수로 매기도록 요청
  (점수가 높을수록 context 안에 relevant portion이 많다는 뜻임)
- Retrieve한 3개 문서에 대한 평균을 최종 점수로 계산

Experiment

Metrics – Answer Similarity (Secondary)

2. Answer Similarity

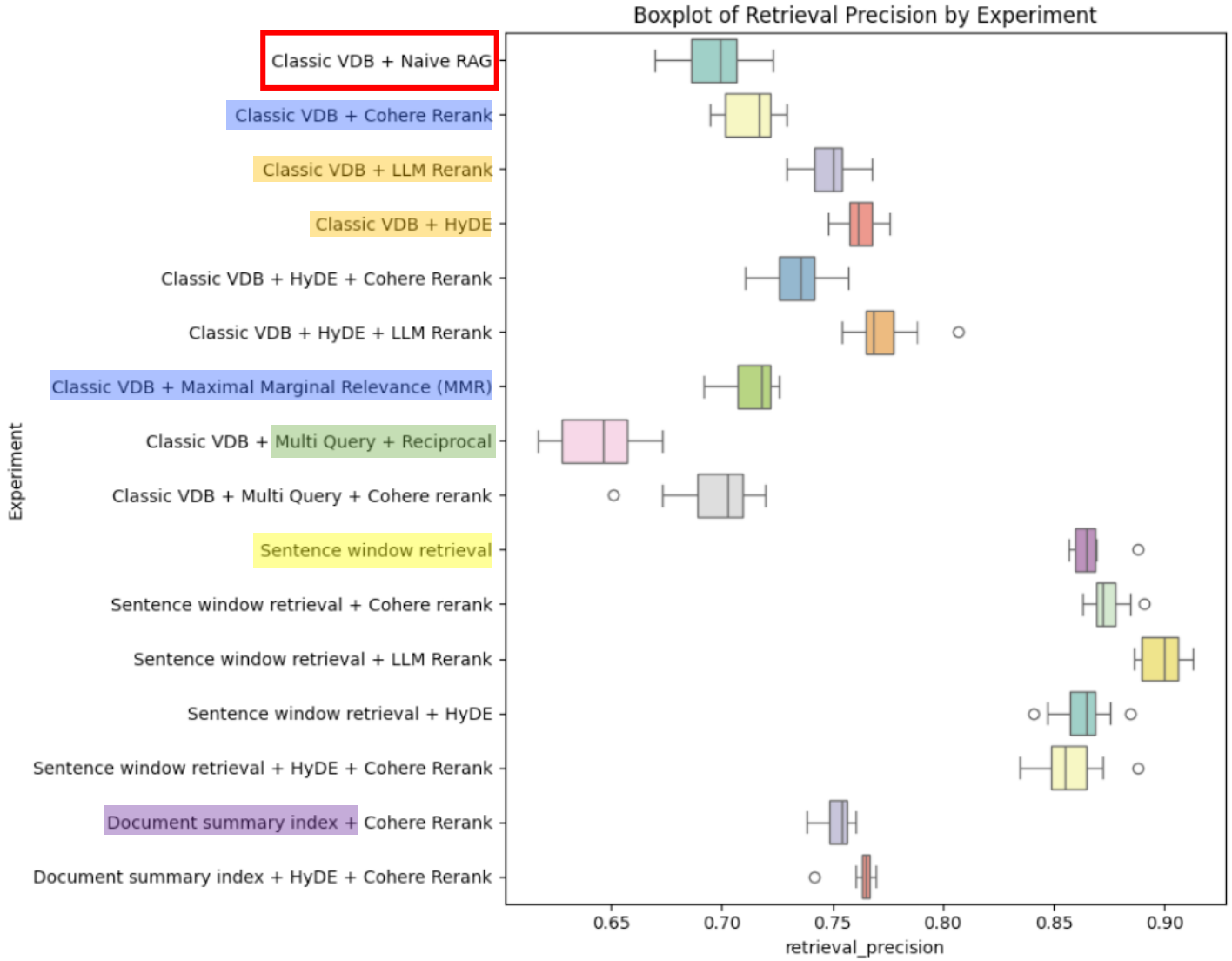
- 각 RAG system으로 생성된 answer가 reference answer와 얼마나 align 되는지 0~5 사이 점수로 평가
- 본 논문의 목적은 어떤 방식이 잘 retrieve 하는 것인지 판단하는 것이므로,

  (+ LLM 생성 시 어떤 오류가 발생할지 모르므로) answer similarity는 secondary metric으로 취급한다고 강조

- Experiment Details
  - 각 RAG techniques을 10 runs 동안 실행하여 10번의 output에 대해 모두 평가
  - 생성 모델, 평가 모델 모두 GPT-3.5-turbo

(Retrieval) Sentence-window > HyDE >= LLM Rerank > Doc-sum-index > Cohere Rerank >= MMR >= Naïve RAG > Multi-query

Results – Retrieval Precision

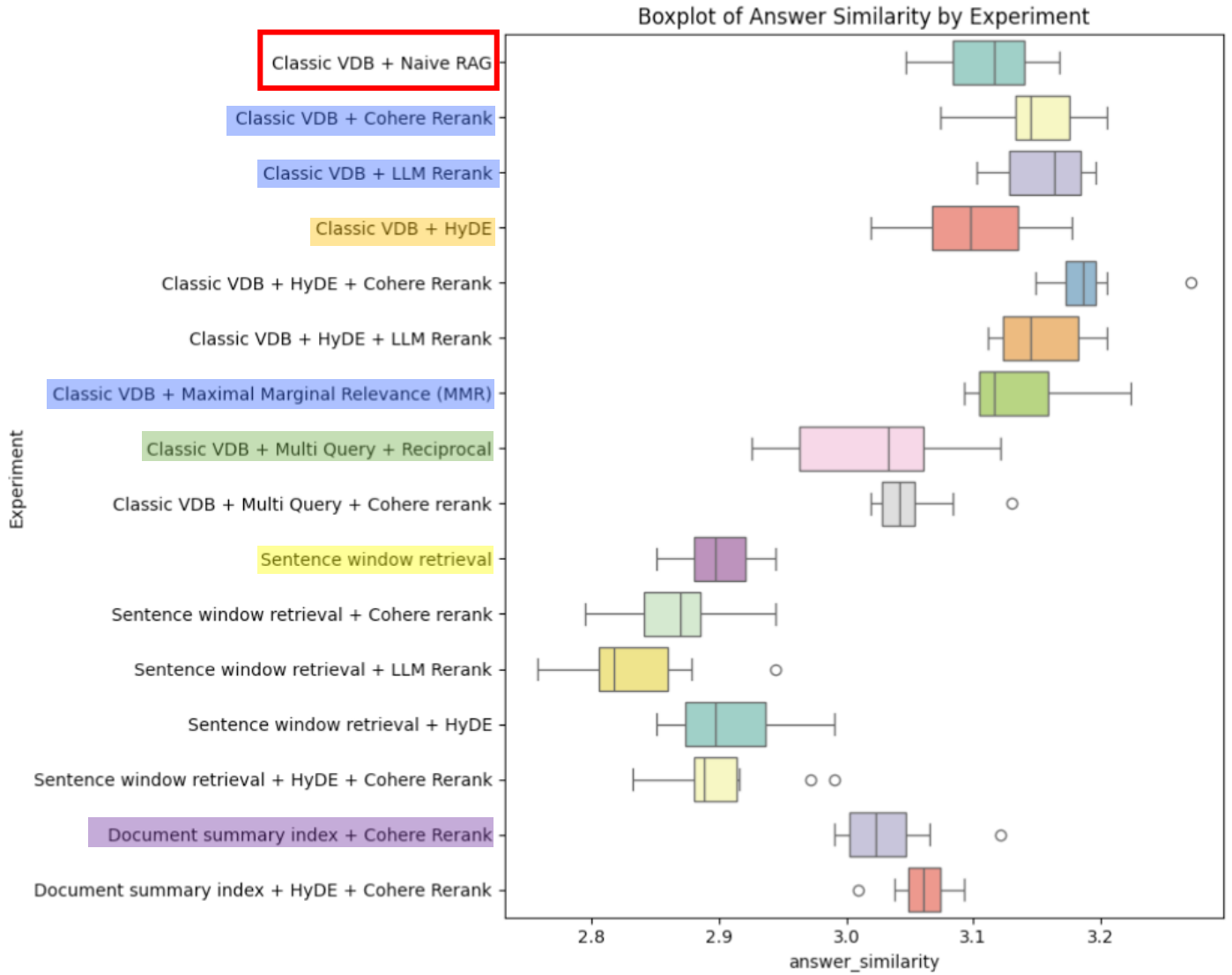


- **Sentence window retrieval**이 가장 effective
- **LLM rerank, HyDE**는 best는 아니지만, Naïve RAG보다는 뛰어남
- **MMR, Cohere Rerank**는 Naïve RAG와 비슷하거나 더 떨어지는 퍼포먼스
- **Multi-query**는 Naïve RAG와 비교하여 훨씬 떨어지는 퍼포먼스
- **Document-summary index**는 Naïve RAG 의 best score와 비슷한 퍼포먼스

(Retrieval) Sentence-window > HyDE >= LLM Rerank > Doc-sum-index > Cohere Rerank >= MMR >= Naïve RAG > Multi-query

(Answer) LLM Rerank >= Cohere Rerank >= MMR >= Naïve RAG > HyDE > Multi-query > Doc-sum-index > Sentence-window

Results – Answer Similarity



- **Sentence window retrieval**이 가장 낮은 스코어를 보임
- **HyDE**는 Naïve RAG과 비슷하거나 낮은 퍼포먼스
- **LLM rerank, MMR, Cohere Rerank**는 Naïve RAG와 비슷하거나 더 높은 퍼포먼스
- **Multi-query**는 Naïve RAG와 비교하여 훨씬 떨어지는 퍼포먼스
- **Document-summary index**도 Naïve RAG 보다 훨씬 떨어지는 퍼포먼스

(Retrieval) Sentence-window > HyDE >= LLM Rerank > Doc-sum-index > Cohere Rerank >= MMR >= Naïve RAG > Multi-query

Results – Tukey's HSD Test for Retrieval Precision - Overall

Technique	Comparison	Mean Diff.	P-adj	Reject Null
Cohere Rerank	Naive RAG	-0.0150	0.4515	False
HyDE	Naive RAG	-0.0648	0.0000	True
HyDE + Cohere Rerank	Naive RAG	-0.0371	0.0000	True
HyDE + LLM Rerank	Naive RAG	-0.0749	0.0000	True
LLM Rerank	Naive RAG	-0.0514	0.0000	True
Maximal Marginal Relevance (MMR)	Naive RAG	-0.0156	0.3787	False
Multi Query + Cohere Rerank	Naive RAG	0.0012	1.0000	False
Multi Query + Reciprocal	Naive RAG	0.0542	0.0000	True

Mean Diff가 음수면 Technique > Comparison / Reject Null이 True이면 "주목할 만한 차이가 있는" 것

1. HyDE + Rerank는 Naïve RAG보다 훨씬 높은 성능
2. MMR, Cohere Rerank는 Naïve RAG와 비교하여 주목할 만하게 성능이 높진 않음
3. Multi-Query가 들어간 방법은 Naive RAG보다 좋지 않은 성능을 보임

(Retrieval) Sentence-window > HyDE >= LLM Rerank > Doc-sum-index > Cohere Rerank >= MMR >= Naïve RAG > Multi-query

Results – Tukey's HSD Test for Retrieval Precision - HyDE

Technique	Comparison	Mean Diff.	P-adj	Reject Null
HyDE	HyDE + Cohere Rerank	-0.0277	0.0002	True
HyDE	HyDE + LLM Rerank	0.0101	0.3255	False
HyDE	LLM Rerank	-0.0134	0.1175	False
HyDE + Cohere Rerank	HyDE + LLM Rerank	0.0378	0.0000	True
HyDE + Cohere Rerank	LLM Rerank	0.0143	0.0842	False
HyDE + LLM Rerank	LLM Rerank	-0.0235	0.0015	True

Mean Diff가 양수면 Technique < Comparison / Reject Null이 True이면 "주목할 만한 차이가 있는" 것

1. HyDE 관련 technique에서는 HyDE + LLM Rerank가 가장 좋은 퍼포먼스를 보임
   - LLM 쓰는 것은 비용이 발생한다는 점에 유의
2. 그 다음으로는 HyDE 단독으로 쓰는 것이 2번째로 좋은 퍼포먼스

(Retrieval) Sentence-window > HyDE >= LLM Rerank > Doc-sum-index > Cohere Rerank >= MMR >= Naïve RAG > Multi-query

Results – Tukey's HSD Test for Retrieval Precision – Sentence window

Base Technique	Compared Technique	Mean Diff.	P-adj	Reject Null
Sentence Window	Sentence Window + Cohere Rerank	0.0090	0.9768	False
Sentence Window	Sentence Window + HyDE	-0.0025	1.0000	False
Sentence Window	Sentence Window + HyDE + Cohere Rerank	-0.0078	0.9945	False
Sentence Window	Sentence Window + LLM Rerank	0.0332	0.0000	True

Mean Diff가 양수면 Technique < Comparison / Reject Null이 True이면 "주목할 만한 차이가 있는" 것

- Sentence Window 방식에 LLM Rerank를 붙이는 것이 Sentence Window를 단독으로 썼을 때보다 주목할 만큼 좋은 퍼포먼스를 보인다.

(Retrieval) Sentence-window > HyDE >= LLM Rerank > Doc-sum-index > Cohere Rerank >= MMR >= Naïve RAG > Multi-query

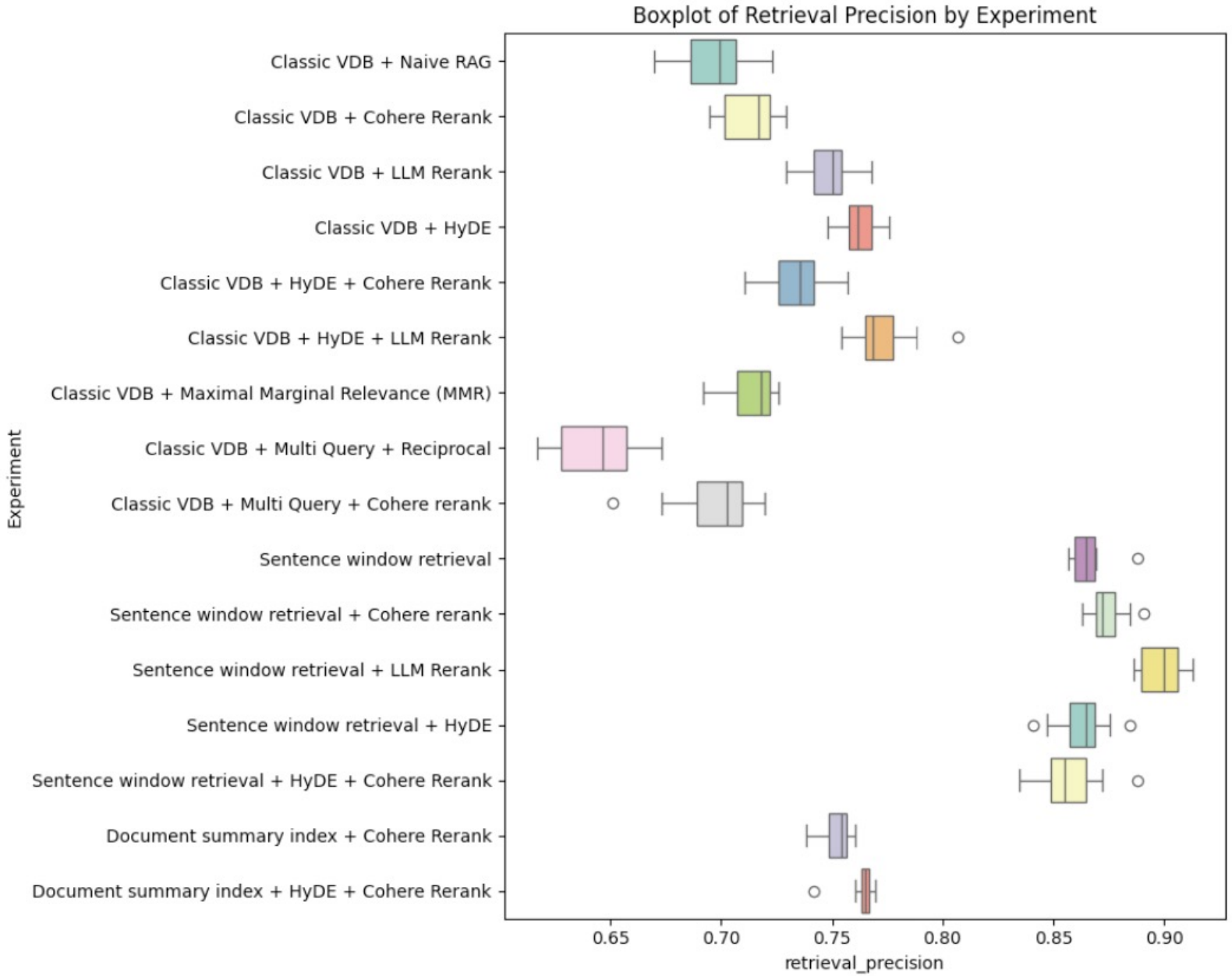
Results – Tukey's HSD Test for Retrieval Precision – Doc Sum

Technique	Comparison	Mean Diff.	P-adj	Reject Null
Doc Summ Index + Cohere Rerank	Classic VDB + Naive RAG	0.0545	0.0000	True
Doc Summ Index + Cohere Rerank	Sentence Window Retrieval	0.1679	0.0000	True
Classic VDB + Naive RAG	Sentence Window Retrieval	0.1134	0.0000	True

Mean Diff가 양수면 Technique < Comparison / Reject Null이 True이면 "주목할 만한 차이가 있는" 것

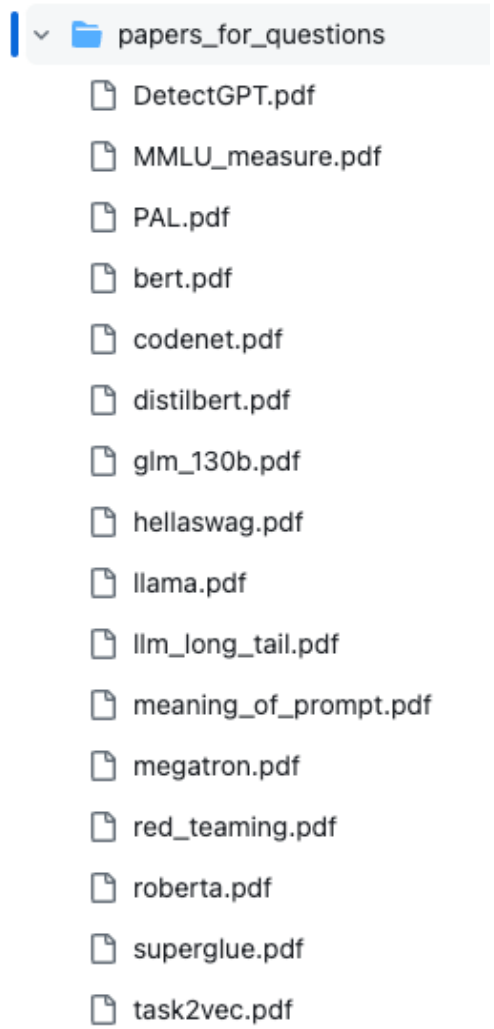
- Doc Summ에 무슨 기법을 더하든 간에, Sentence Window Retrieval이 가장 좋았다

Results


Boxplot of Retrieval Precision by Experiment
Classic VDB + Naive RAG
Classic VDB + Cohere Rerank
Classic VDB + LLM Rerank
Classic VDB + HyDE
Classic VDB + HyDE + Cohere Rerank
Classic VDB + HyDE + LLM Rerank
Classic VDB + Maximal Marginal Relevance (MMR)
Classic VDB + Multi Query + Reciprocal
Classic VDB + Multi Query + Cohere rerank
Sentence window retrieval
Sentence window retrieval + Cohere rerank
Sentence window retrieval + LLM Rerank
Sentence window retrieval + HyDE
Sentence window retrieval + HyDE + Cohere Rerank
Document summary index + Cohere Rerank
Document summary index + HyDE + Cohere Rerank
Experiment
0.65
0.70
0.75
0.80
0.85
0.90
retrieval_precision

Conclusion

- 본 논문의 기여점
  - 넘쳐나는 RAG 기법들 중 가장 많이 사용되는 핵심 기법들을
    하나의 방법론으로 비교/분석한 것이 꼭 필요하다고 생각했었음
  - 단일 방식이 아니고, 섞어서 비교한 방식이 보기 좋았음
- Retrieval precision과 Answer similarity 결과가 크게 다른 것에 관한 나의 생각
  - Query를 뽑은 논문이 대부분 유명한 논문들임 (일부로 이렇게 설정했다고 함).
    유명한 논문들이므로, LLM의 parametric knowledge가 많이 영향을 끼치지 않았을까?
  - 논문에서도 언급했지만, 더 나은 LLM을 썼으면 두 결과의 일관성을
    어느 정도 볼 수 있지 않았을까?



Thank you

Q&A