Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

NeurIPS 2020

Suyeon Cha
Pseudo-Lab
RAG is All you need

2024.10.07

Index

Open-Domain Question Answering (ODQA)

ODQA는 모델에게 자연어로 된 Factoid question에 답을 요구하는 언어 과제 중 한 유형이다.
정답은 객관적이므로, 모델 성능을 평가하기 간단하다.

Question: What did Albert Einstein win the Nobel Prize for?
Answer: The law of the photoelectric effect.

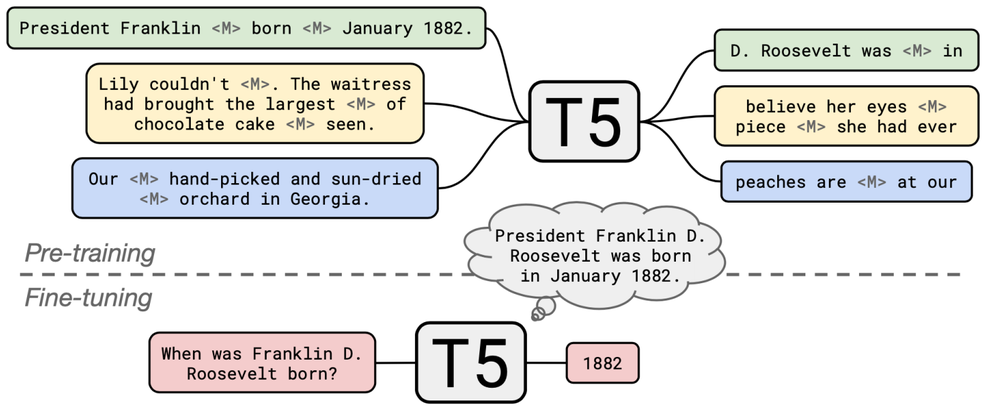
‘Open-Domain’이라는 것은 임의로 물어본 사실 기반 질문에 대해 사전 맥락 없이도 답변할 수 있음을 의미
모델은 오직 질문만 입력으로 받으며, 아인슈타인이 왜 상대성 이론으로 노벨상을 받지 못했는지에 대해서는 모른다.

Closed-Book vs. Open-Domain QA

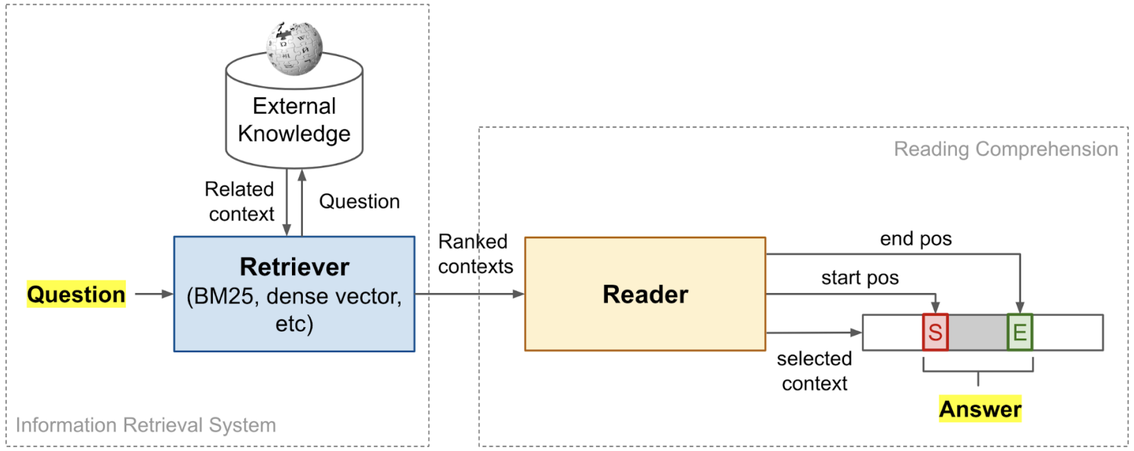
Oepn-Domian ≠ Reading Comprehension

- ODQA: 모델이 질문에 대해 답할 때 관련된 맥락이나 배경이 주어지지 않음. 질문 자체만으로 답변을 생성해야 함.
- RC: 질문과 함께 관련된 문서가 제공된다면 이는 독해에 해당. RC에서는 문서에서 답을 찾으면 되므로 ODQA보다 더 제한적 작업임.

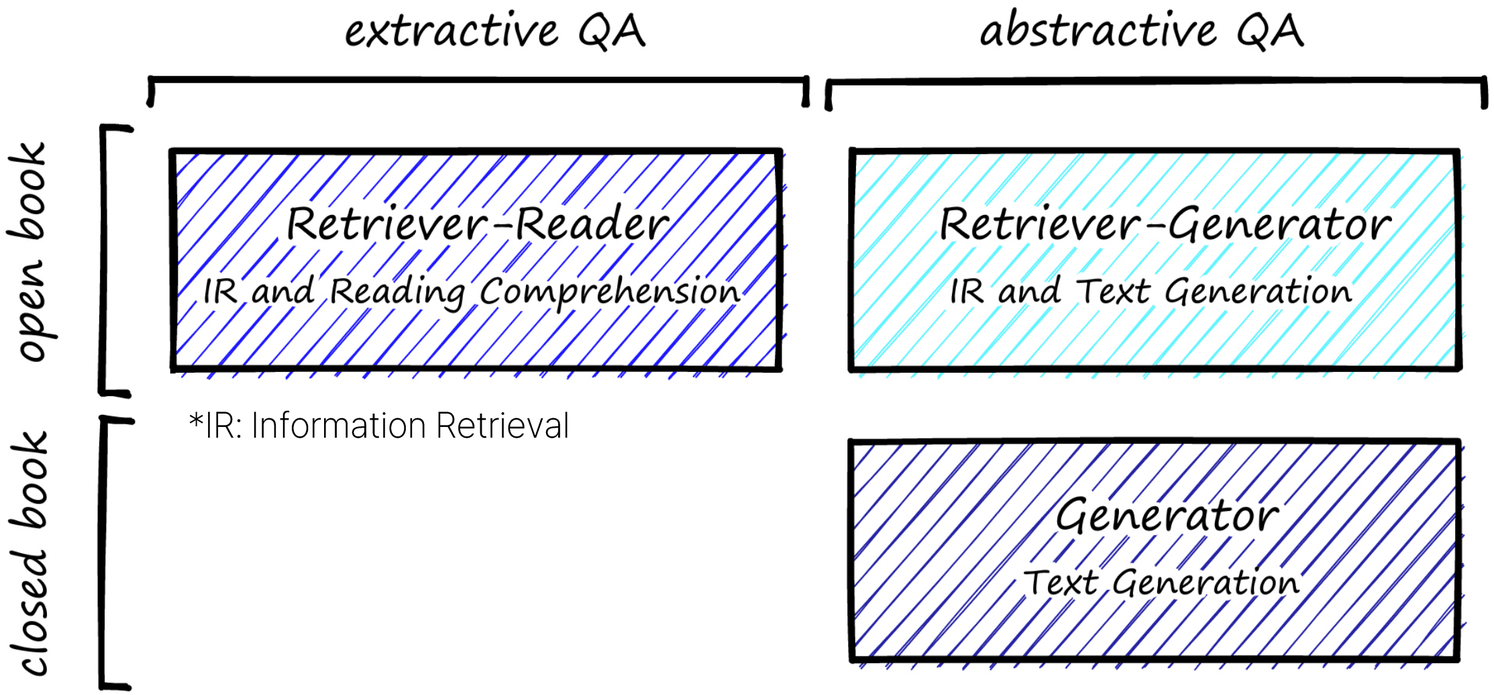
Closed-Book QA



Open-Domain QA

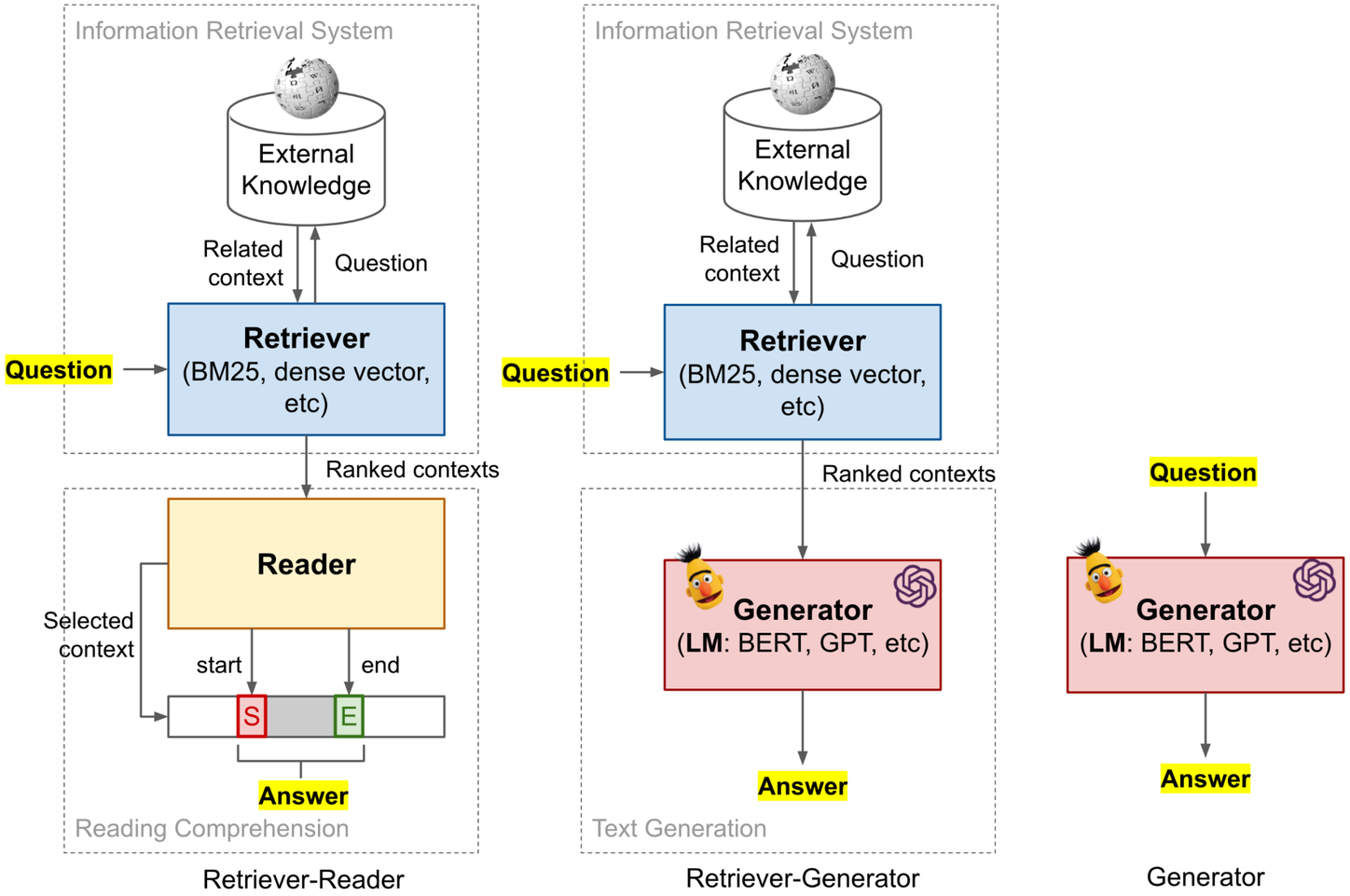


Closed-Book vs. Open-Domain QA



1. Open-book extractive QA (top-left)
2. Open-book abstractive QA (top-right)
3. Closed-book abstractive QA (Bottom)

Different types of open-domain questions



1. Generator (☆)

모델이 훈련 중에 본 질문에 대한 답을 정확히 기억하고 답변하는 것

2. Retriver-Reader (☆☆)

새로운 질문이지만, 훈련 중 본 답변 중에 선택하는 것

3. Retriver-Generator (☆☆☆)

훈련 데이터에 포함되지 않은 새로운 질문에 대해 외부 지식 소스를 검색하고, 그 정보를 바탕으로 새로운 답변을 생성하는 것

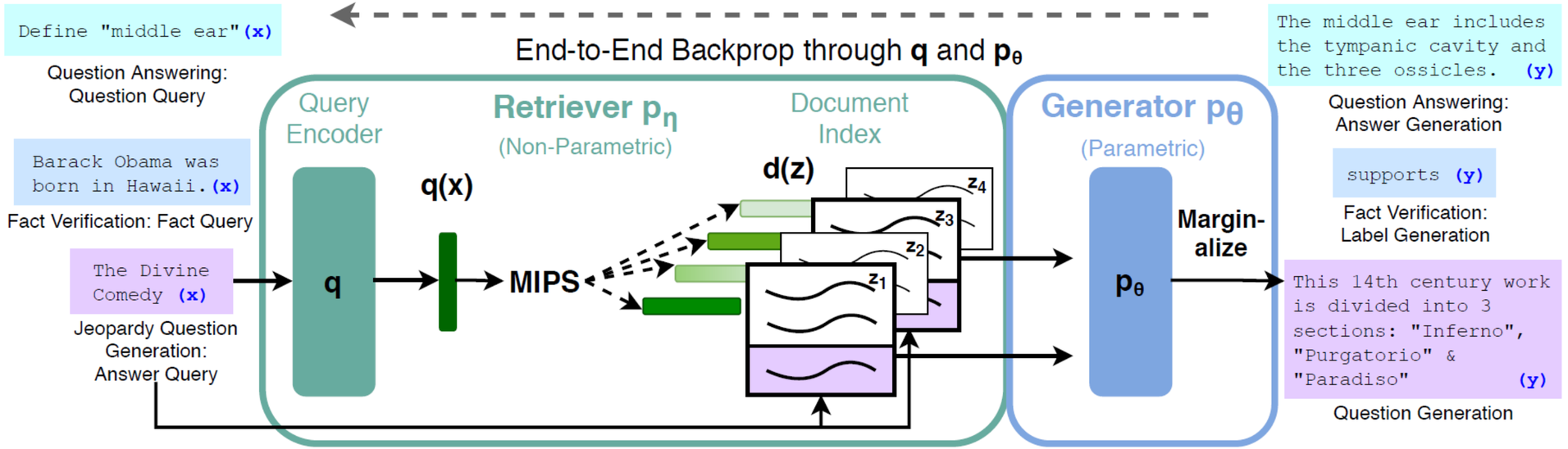
Open-Domain Question Answering (ODQA)

ODQA는 모델에게 자연어로 된 Factoid questio에 답을 요구하는 언어 과제 중 한 유형이다.
정답은 객관적이므로, 모델 성능을 평가하기 간단하다.

Question: What did Albert Einstein win the Nobel Prize for?
Answer: The law of the photoelectric effect.

‘Open-Domain’이라는 것은 임의로 물어본 사실 기반 질문에 대한 관련된 맥락이 없다는 것을 의미
모델은 오직 질문만 입력으로 받으며, 아인슈타인이 상대성 이론으로 노벨상을 받지 못했는지에 대해서는 모른다.

Model Architecture

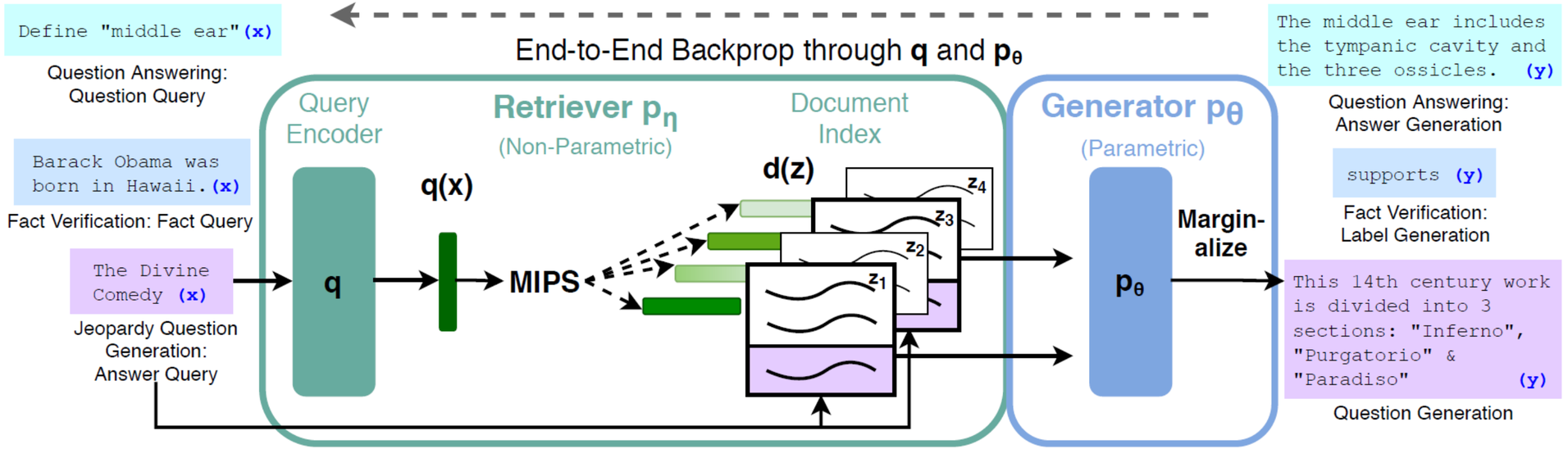


RAG는 두 가지 메모리 시스템을 결합한 모델

⟹ Pre-trained parametric-memory generation model + Non-parametric memory

- Parametric-momory: 사전 학습된 seq2seq transformer model (BART, T5)
- Non-parametric memory: 외부 데이터에서 필요한 정보를 검색하는 시스템. Dense Passage RetrieverDPR)를 사용.

Model Architecture

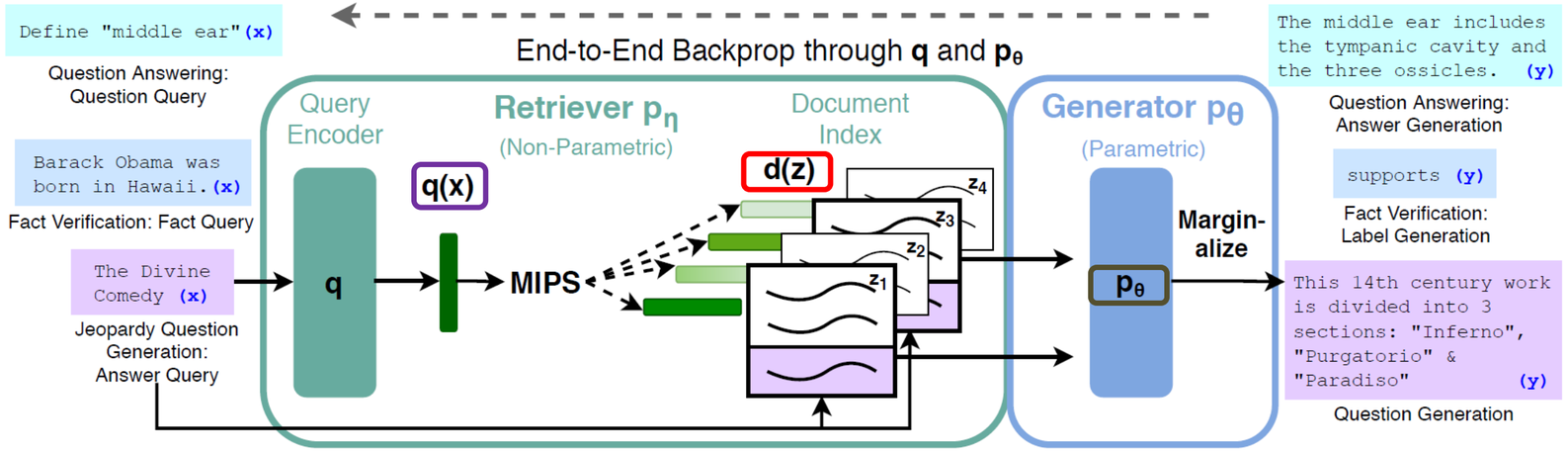


쿼리 x에 대해 MIPS를 사용하여 상위 k개의 문서 z_i를 찾음

Qeury Encoder와 Document Index를 통해 검색된 문서를 다룸

최종 예측 y를 위해 z를 latent vector로 간주하고,
다른 문서들이 주어졌을 때 seq2seq 예측을 모두 합산하여 처리함

Model Architecture



x: 입력 시퀀스 (Qeury)

z: 검색된 문서 (Retrieved Document)

y: 목표 시퀀스 (Target Sequence)

- $q(x)$: 쿼리 인코더로 입력된 질문 또는 쿼리 x를 vector representation으로 변환하는 역할
- $d(z)$: 문서 인덱스 (Document Index)로 검색기(retriever)가 관련 문서 z를 찾기 위한 문서 벡터 인덱스를 나타냄
- p_θ: 생성기(Generator)로 최종적으로 문서 z와 출력 시퀀스 y를 생성하는 seq2seq Generator임

Retriever Model - DPR

입력 시퀀스 x를 사용하여 관련 텍스트 passage z를 검색함

이 검색 과정은 **DPR**로 구현되며, 질문과 문서를 모두 벡터로 변환하여 두 벡터 간의 내적을 통해 유사도가 높은 문서를 선택

$$p_\eta(z \mid x) \propto \exp\left(d(z)^\top q(x)\right) \qquad d(z) = \mathrm{BERT}_d(z), \quad q(x) = \mathrm{BERT}_q(x)$$

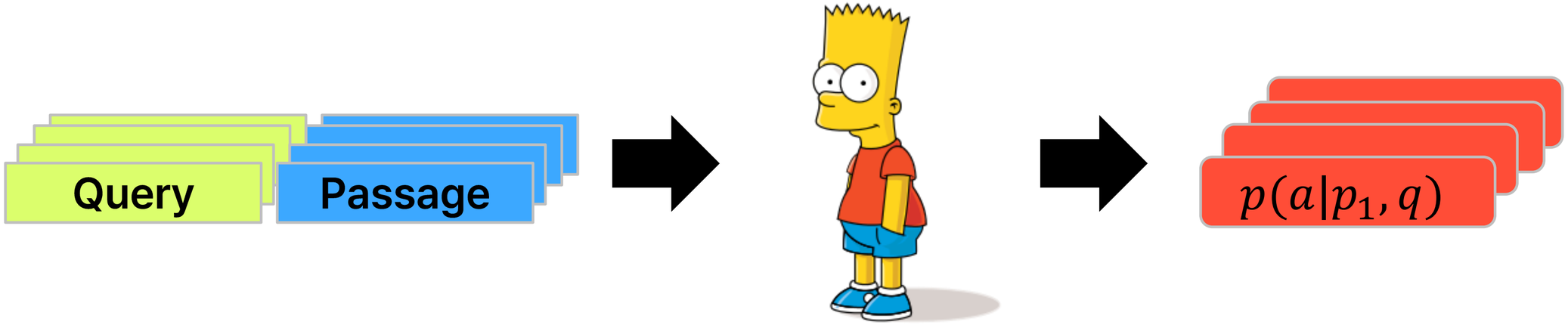
문서 인코더 d(x): 문서 z를 벡터로 변환 / 쿼리 인코더 q(x): 입력된 질문이나 쿼리를 벡터로 변환

- *Bi-encoder 구조 사용*
- *Pretrained BERT Model 사용*
- *MIPS를 사용하여 벡터 공간에서 가장 높은 내적값을 가진 문서를 빠르게 찾음*

Generator Model - BART

검색된 텍스트 passage z를 추가적인 컨텍스트로 사용하여 목표 시퀀스 y를 생성함

이때 입력 시퀀스 x와 검색된 문서 z는 단순히 연결(Concat)되어 BART 모델에 입력됨



Training

NLL (Negative Log-Likelihood) 손실을 최소화하도록 Retriever와 Geneartor를 함께 학습

$$\mathrm{NLL} = -\log \sum_{z} p_{\eta}(z|x)\, p_{\theta}(y|x,z)$$

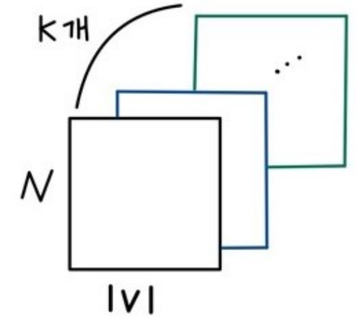
- Passage Encoder d(z)를 업데이트 하는 것은 비용이 많이 들며, MIPS를 위해 문서를 재인덱싱 해야 하기 때문에 어려움
- 그러나 RAG는 ORQA*와 달리 Passage Encoder를 fine-tuning할 필요가 없다고 보며, 대신 Query Encoder와 Generator만 업데이트 함

*Latent Retrieval for Weakly Supervised Open Domain Question Answering(2020)

How RAG Decodes

RAG-Sequence

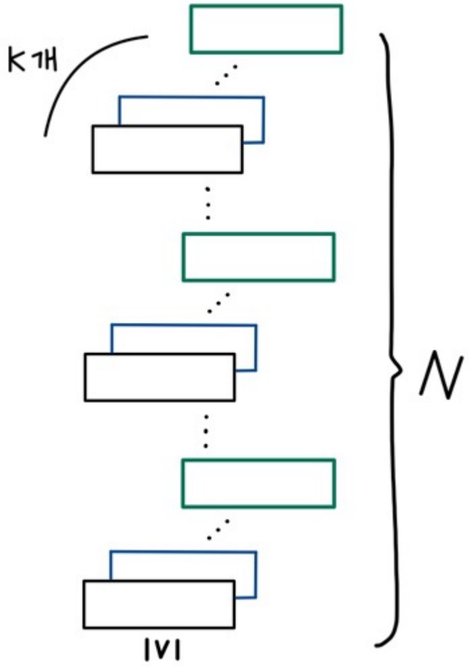
$$P_{\text{RAG-Sequence}}(y|x) \approx \sum_{z \in \text{top-k}(p(\cdot|x))} p_\eta(z|x)\, p_\theta(y|x,z) = \sum_{z \in \text{top-k}(p(\cdot|x))} p_\eta(z|x) \prod_{i=1}^{N} p_\theta(y_i|x,z,y_{1:i-1})$$



- 전체 시퀀스를 **하나의 문서에 기반**하여 생성
- 문서 하나만을 참조해 출력 토큰을 생성함 → 더 단순한 정보 통합

RAG-Token

$$P_{\text{RAG-Token}}(y|x) \approx \prod_{i=1}^{N} \sum_{z \in \text{top-k}(p(\cdot|x))} p_\eta(z|x)\, p_\theta(y_i|x,z,y_{1:i-1})$$



- **각 토큰을 서로 다른 문서에 기반**하여 생성
- 여러 문서에서 필요한 정보를 골라냄 → 세밀하고 복잡한 답변을 생성

RAG-Sequence Model

- p_η : Retriever
- p_θ : Generator

$$P_{\text{RAG-Sequence}}(y|x) \approx \sum_{z \in \text{top-k}(p(\cdot|x))} p_\eta(z|x)\, p_\theta(y|x,z) = \sum_{z \in \text{top-k}(p(\cdot|x))} p_\eta(z|x) \prod_{i=1}^{N} p_\theta(y_i|x, z, y_{1:i-1})$$

모든 Passage에 대해 가중 합을 곱함

Retriever가 결정

쿼리 x에 대해 문서 z가 선택될 확률로,
쿼리 x에 대해 검색된 문서들 중 상위 k개의 문서에 대해 확률을 부여

Generator가 계산

문서 z를 참고하면서 생성 y 의 i번째 토큰을 쿼리 x, 해당 문서 z,
그리고 이전에 생성된 토큰 $y_{1:i-1}$의 정보에 기반해 예측하는 확률

여러 문서 z 각각에 계산된 생성 확률 p_θ를 모두 더해 고려
즉, 검색된 상위 k개의 문서에 대해 확률을 계산하고 그 결과를 합쳐 최종적 P(y|x)를 구하는 방식

RAG-Token Model

각 토큰 y_i에 대해 모든 토큰을 연속적으로 생성

$$P_{\text{RAG-Token}}(y|x) \approx \prod_{i=1}^{N} \sum_{z \in \text{top-k}(p(\cdot|\ldots))} p_\eta(z|x)\, p_\theta(y_i|x, z, y_{1:i-1})$$

입력 x에 대해 top-K 문서를 선택하는 과정

Retriever가 결정

쿼리 x에 대해 문서 z가 선택될 확률로,
쿼리 x에 대해 검색된 문서들 중 상위 k개의 문서에 대해 확률을 부여

Generator가 계산

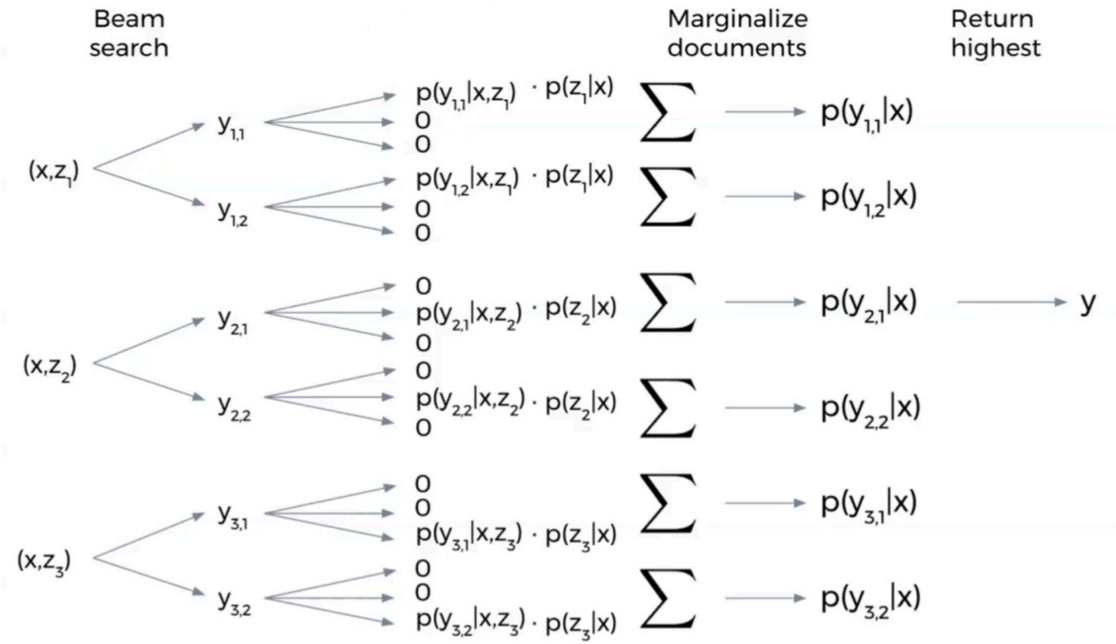
문서 z를 참고하면서 생성 y의 i번째 토큰을 쿼리 x, 해당 문서 z,
그리고 이전에 생성된 토큰 $y_{1:i-1}$의 정보에 기반해 예측하는 확률

각 출력 토큰 y_i에 대해 검색된 상위 k개의 문서 중 하나를 선택하여
각각의 생성 확률을 계산한 후, 합산하여 최종적 P(y|x)를 계산하는 방식

Decoding/Test time

1. RAG-Token은 단일 Beam search를 통해 평가됨

2. RAG-Sequence는 token 별 likelihood를 나눌 수 없으므로, 각 후보 문서에 대해 Beam search를 실행하고 최적의 문서를 선택



RAG-Sequence Model Decoding

Open-Domain Question Answering

	Model	NQ	TQA	WQ	CT
Closed Book	T5-11B [52]	34.5	- /50.1	37.4	-
	T5-11B+SSM[52]	36.6	- /60.5	44.7	-
Open Book	REALM [20]	40.4	- / -	40.7	46.8
	DPR [26]	41.5	**57.9**/ -	41.1	50.6
	RAG-Token	44.1	55.2/66.1	**45.5**	50.0
	RAG-Seq.	**44.5**	56.8/**68.0**	45.2	**52.2**

Model	Jeopardy B-1	Jeopardy QB-1	MSMARCO R-L	MSMARCO B-1	FVR3 Label Acc.	FVR2 Label Acc.
SotA	-	-	**49.8***	**49.9***	**76.8**	**92.2***
BART	15.1	19.7	38.2	41.6	64.0	81.1
RAG-Tok.	**17.3**	**22.2**	40.1	41.5	72.5	89.5
RAG-Seq.	14.7	21.4	40.8	44.2		

- RAG는 T5와 같은 다른 모델들과 비교했을 때 더 높은 성능을 보여줌
- RAG 모델은 전통적인 추출 기반 모델(extractive models)보다 우수한 성능을 기록
- 특히, RAG는 추출이 아닌 생성(Generation) 방식을 사용하여 더 정확하고 포괄적인 답변을 생성함
- RAG-Sequence와 RAG-Token의 차이점에도 불구하고, 두 모델 모두 ODQA에서 강력한 성능을 보임

Abstract Question Answering

Task	Input	Model	Generation
MS-MARCO	define middle ear	BART	?The middle ear is the part of the ear between the middle ear and the nose.
		RAG-T	The middle ear is the portion of the ear internal to the eardrum.
		RAG-S	The middle ear includes the tympanic cavity and the three ossicles.
	what currency needed in scotland	BART	The currency needed in Scotland is Pound sterling.
		RAG-T	Pound is the currency needed in Scotland.
		RAG-S	The currency needed in Scotland is the pound sterling.
Jeopardy Question Gener -ation	Washington	BART	?This state has the largest number of counties in the U.S.
		RAG-T	It's the only U.S. state named for a U.S. president
		RAG-S	It's the state where you'll find Mount Rainier National Park
	The Divine Comedy	BART	*This epic poem by Dante is divided into 3 parts: the Inferno, the Purgatorio & the Purgatorio
		RAG-T	Dante's "Inferno" is the first part of this epic poem
		RAG-S	This 14th century work is divided into 3 sections: "Inferno", "Purgatorio" & "Paradiso"

Hallucination

Diversity

RAG 모델과 BART 모델이 생성한 답변을 비교

사실적으로 더 정확한 답변을 생성하는 빈도가 높아 환각(Hallucination)이 덜 발생하였음

기존 모델보다 다양성 측면에서도 더 나은 결과를 보여줌

→ 기존 Gold passage에 의존하지 않고도 복잡한 질문에 정확하고 다양한 답변 생성

Ablation Study

Model	NQ	TQA	WQ	CT	Jeopardy-QGen		MSMarco		FVR-3	FVR-2
		Exact Match			B-1	QB-1	R-L	B-1	Label Accuracy	
RAG-Token-BM25	29.7	41.5	32.1	33.1	17.5	22.3	55.5	48.4	**75.1**	**91.6**
RAG-Sequence-BM25	31.8	44.1	36.6	33.8	11.1	19.5	56.5	46.9		
RAG-Token-Frozen	37.8	50.1	37.1	51.1	16.7	21.7	55.9	49.4	72.9	89.4
RAG-Sequence-Frozen	41.2	52.1	41.8	52.6	11.8	19.6	56.7	47.3		
RAG-Token	43.5	54.8	**46.5**	51.9	**17.9**	**22.6**	56.2	**49.4**	74.5	90.6
RAG-Sequence	**44.0**	**55.8**	44.9	**53.4**	15.3	21.5	**57.2**	47.5		

1. BM25를 사용한 모델은 상대적으로 낮은 성능

- RAG의 Retriever를 DPR 대신 BM25(TF-IDF 기반)으로 교체하였더니 대부분 task에서 기존 모델보다 낮은 성능을 보임

2. Retriver Freeze: Generator만 학습했을 때, RAG-Token/Sequence 모델 모두 낮은 성능

- Retriever과 함께 학습 가능한 상태에서 더 좋은 성능을 보임

3. RAG-Token은 여러 데이터셋에서 RAG-Sequence보다 더 높은 성능을 보임

*FVR-2/-3 task에서만 높은 성능이 나오는 이유는 사실 확인 task로 중요 token의 등장 여부가 중요하기 때문

Effect of Retrieving more documents

검색된 문서(K) 개수에 따른 성능 변화를 보여줌

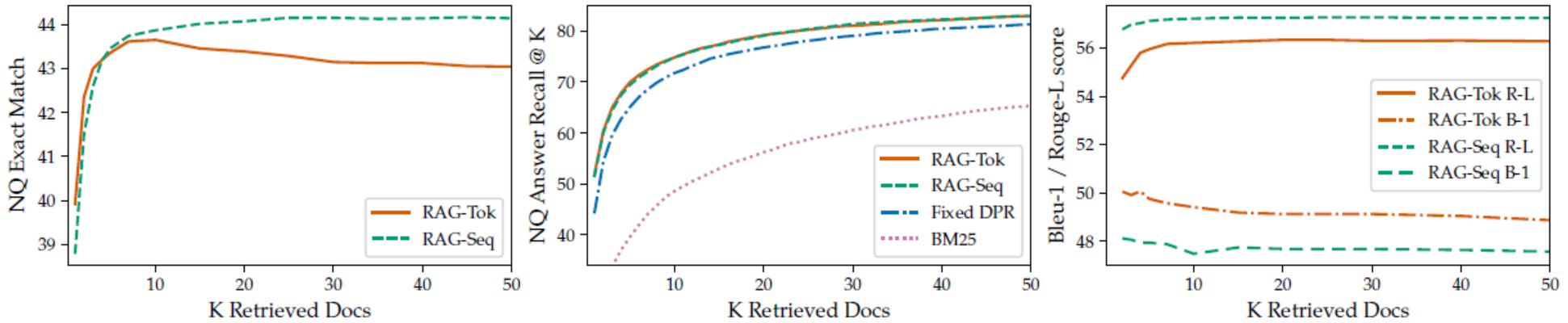


Figure 3: Left: NQ performance as more documents are retrieved. Center: Retrieval recall performance in NQ. Right: MS-MARCO Bleu-1 and Rouge-L as more documents are retrieved.

- Left(정확한 일치 비율): 문서 개수가 10개 이상으로 증가해도 성능 향상은 미미하며, RAG-Sequence가 더 나은 성능
- Middle(정답이 검색된 문서에 존재하는 비율): DPR을 사용하는 RAG 모델의 성능이 뛰어나며, 30개 이상 문서 검색 시 최적 성능 달성
- End(BLUE-1, ROUGE-L 성능): RAG-Sequence는 생성된 텍스트 품질 면에서 RAG-Token 보다 더 나은 성능

Conclusion

1. Parametric + Non-parametric Memory의 결합

- RAG 모델은 Parametric Memory와 Non-parametric Memory를 결합하여 더 유연하고 강력한 성능 발휘
- DPR을 활용한 검색과 BART 기반의 생성기를 결합하여 모델이 외부 정보를 검색하고 이를 기반으로 텍스트를 생성하는 방식은 기존 모델들보다 더 다양한 질문에 대응할 수 있게 함

2. RAG-Sequence와 RAG-Token

- RAG-Sequence는 동일한 문서를 사용해 전체 시퀀스를 생성하고, RAG-Token은 각 토큰마다 다른 문서를 사용할 수 있어, 두 가지 모델은 서로 다른 작업과 요구에 맞는 최적화된 성능을 제공함
- RAG-Token은 더 넓은 범위의 정보를 종합할 수 있는 유연성을 제공하고, RAG-Sequence는 일관성 있는 텍스트를 생성하는 데 강점을 보였음

Conclusion

3. ODQA에서의 성능 개선
- RAG는 Open-Domain Question Answering(ODQA) 작업에서 Natural Questions, TriviaQA, WebQuestions와 같은 데이터셋에서 최첨단 성능을 달성
- 특히, 생성 기반 접근이 추출 기반 접근보다 더 나은 성능을 보여주고, 다양한 질문에 대한 정확한 답변을 생성함

4. 다양한 NLP Task에서 성능 입증
- RAG 모델은 ODQA 외에도 MS-MARCO, FEVER와 같은 데이터셋에서 요약, 질문 생성, 사실 검증과 같은 다양한 NLP 작업에서 강력한 성능을 보임
- MS-MARCO에서 BART를 능가하는 성능을 기록하며, 추상적 질문 응답에서 더 적은 환각과 더 높은 사실성을 제공

5. Decoding Strategy
- 논문은 Thorough Decoding과 Fast Decoding을 제안하여, 디코딩 과정에서의 효율성과 성능 간의 균형을 맞추는 방법을 설명
- RAG 모델이 다양한 작업에서 계산 비용을 줄이면서도 높은 성능을 유지할 수 있도록 설계됨

Q&A

THANK YOU