《Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering》

1. Introduction
2. Method
3. Used-Model
4. Evaluation
5. Conclusion

1. Introduction

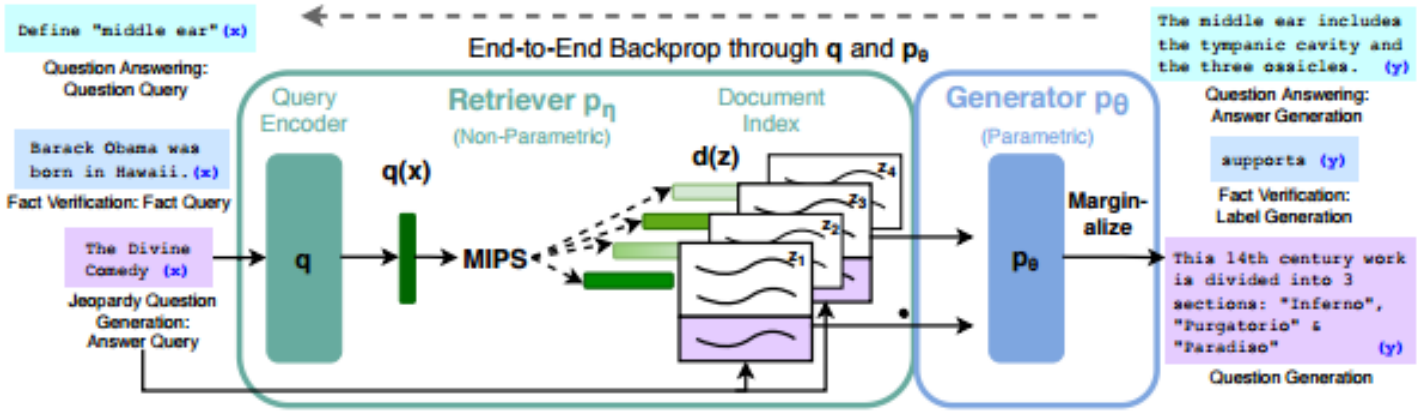
- 본 논문에서는 RAG 모델을 내부적으로 수정하여 발전시킨 FiD 모델을 제안함.

- RAG의 목적과 마찬가지로 답변시에 작은 모델을 통해서도 정확한 정보를 가져오기 위해 만들어진 기술.

  - 즉, external knowledge corpus로부터 knowldege를 추가적으로 전달받음.

- RAG와 같이 Encoder-Decoder로 구성됐지만,
인코더에서 디코더로 넘어가는 부분에서 차이가 존재한다.

Recap RAG



RAG의 Encoder-Decoder : RAG sequence / RAG token

1. 인코더를 통해 질문을 인코딩하고 독립적으로 모든 문서를 인코딩함.
2. 인코딩된 질문과 문서를 통해 문서마다 독립적으로 답변을 생성함.
   즉, 한 문서당 한 대답이 나옴 / 한 문서당 한 단어가 나옴

   RAG sequence　　RAG token
3. 이걸 marginalize, 즉 확률 높은 / 정확도가 높은것만 취급을 해서 output으로 뱉음.

Rag sequence – 문서 A/B/C: 여러분 "~~" 이 문장으로 타협하시죠

Rag token – 문서 A/B/C: 여러분 첫번째 단어는 "~" 이게 좋을 듯, 두번 째는 "~" ....

- 이 과정에서 기존 RAG는 연산과정이 복잡하고, passage 수에 따라 연산과정이 quadric 하게 증가.

- Passage가 디코딩이 될 때, 독립적으로 처리되어 passage간 정보 통합이 안됨. -> 이래서 rag-sequence/rag-token 을 사용하게 됨

  이렇게 독립적으로 처리하게 되면 문서가 많아질 경우 오히려 성능이 감소함.

2. Method

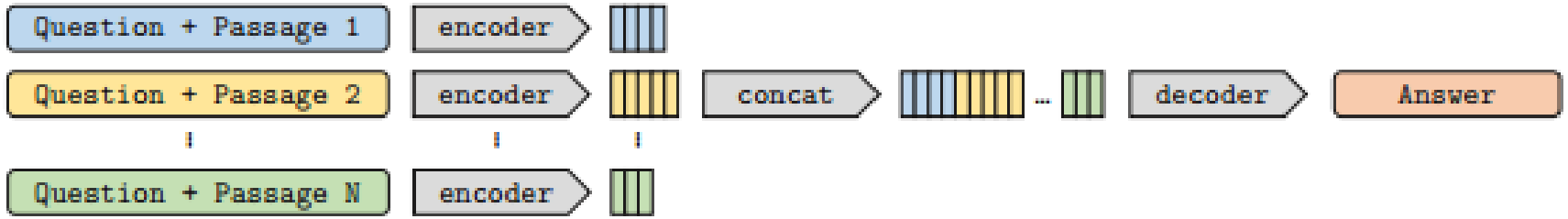
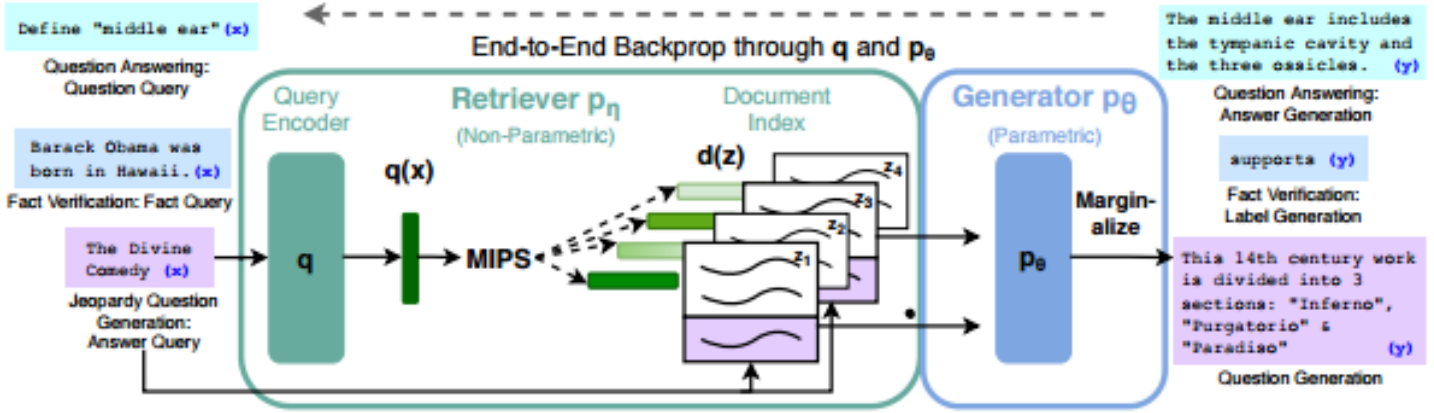


Figure 2: Architecture of the Fusion-in-Decoder method.

- FiD는 retriever을 통해corpus, 즉 참고문서 중 input과 유사한 부분을 가져와서 인코더로 hidden representation 을 얻는건 RAG와 똑같음.

- 모든 hidden representation을 concat해서 디코더에 넣어서 그걸 기반으로 답변을 생성하게 함.

RAG workflow



- RAG는 RAG-sequence/RAG-token을 통해 "입력 -> 문서 –> sequence생성"의 과정을 통해 output이 산출이 됨.
- 즉, output이 생성될 때 문서를 기반으로 생성이됨.
- 반면 FiD는 입력과 입력을 통해 retrieve된 문서가 함께 concat 되어 output을 생성하게됨.

FiD workflow

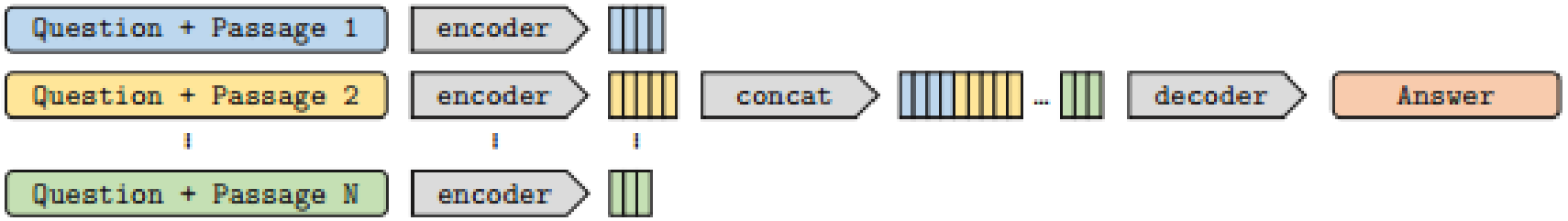


Figure 2: Architecture of the Fusion-in-Decoder method.

Workflow 요약

1. 사용자 질문 x 가 들어옴
2. 사용자 질문 x 를 사용해 retrieve로 top-k개의 문서 d_k에서 관련된 내용 n을 찾음.
3. X+n을 하여 representation 을 만들어줌. 그러면 k개의 (x+n) 형태가 나오게 되고, 이거를 다 합침. -> (x+n) * k
4. 이 (x+n)*k를 디코더에 넣어서 최종 output을 생성

장점:

1. 그렇게 답변의 계산되는 수가 linear하게 증가하여 감소하게 됨.
2. cross attention으로서 passage끼리 서로의 정보를 참고하여 답을 생성할 수 있음.

3. Used-Model

- Encoder-Decoder: T5 base(220M), T5 large(770M)
- Retriever : DPR, BM25 (학습하지 않고 일반 모델로 진행했다고함.)
- Dataset: Natural Question(NQ), TriviaQA, SQuAD

- Input: question, title, answer
- Output: Answer

4. Evaluation

FiD

Model	NQ	TriviaQA		SQuAD Open	
	EM	EM	EM	EM	F1
DrQA (Chen et al., 2017)	-	-	-	29.8	-
Multi-Passage BERT (Wang et al., 2019)	-	-	-	53.0	60.9
Path Retriever (Asai et al., 2020)	31.7	-	-	**56.5**	**63.8**
Graph Retriever (Min et al., 2019b)	34.7	55.8	-	-	-
Hard EM (Min et al., 2019a)	28.8	50.9	-	-	-
ORQA (Lee et al., 2019)	31.3	45.1	-	20.2	-
REALM (Guu et al., 2020)	40.4	-	-	-	-
DPR (Karpukhin et al., 2020)	41.5	57.9	-	36.7	-
SpanSeqGen (Min et al., 2020)	42.5	-	-	-	-
RAG (Lewis et al., 2020)	44.5	56.1	68.0	-	-
T5 (Roberts et al., 2020)	36.6	-	60.5	-	-
GPT-3 few shot (Brown et al., 2020)	29.9	-	71.2	-	-
Fusion-in-Decoder (base)	48.2	65.0	77.1	53.4	60.6
Fusion-in-Decoder (large)	**51.4**	**67.6**	**80.1**	**56.7**	63.2

RAG

Model	NQ	TQA Exact M
RAG-Token-BM25	29.7	41.5
RAG-Sequence-BM25	31.8	44.1
RAG-Token-Frozen	37.8	50.1
RAG-Sequence-Frozen	41.2	52.1
RAG-Token	43.5	54.8
RAG-Sequence	**44.0**	**55.8**

1. TriviaQA에 EM이 두개가 있는데,

왼쪽: open-domain: 이 설정에서는 모델이 외부의 매우 큰 문서 저장소(ex:위키피디아)에서 답을 검색하여 찾고, 그 문서에서 답변을 추출함.

- 검색 성능을 주로 보게됨.

(응용: 일반 지식 검색, 가상 비서, 고객 지원)

오른쪽: closed-domain: 이 설정에서는 답변이 이미 포함된 지정된 문서에서만 답을 추출하기만 하면 됨.

- 주저진 정보에서 답변 퀄리티를 보게됨.

(응용: 회사/학교 내부 문서(법률 같은거) 검색, 교육 자료 등)

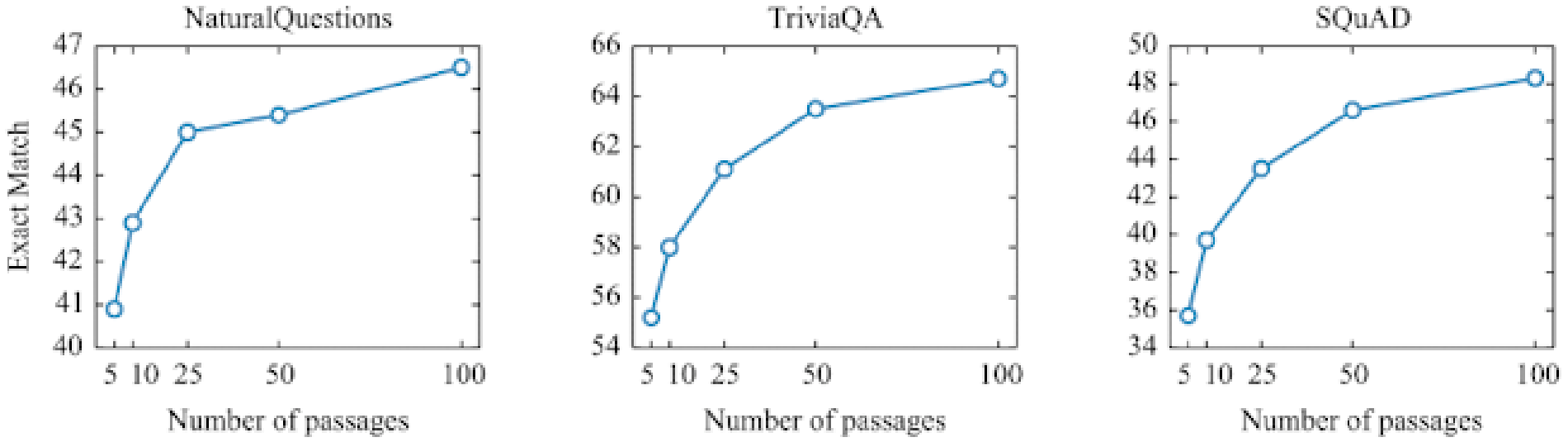
2. 결과:

- RAG에서는 open-domain에 대해서만 진행하였으며, 점수 비교 결과 FiD기법이 더 좋은 성능을 보임.
- FiD에서 모델의 크기가 클수록 더 좋은 결과가 나옴.

Top-k 갯수에 따른 성능 비교

- 논문에서는 retrieve된 문서의 개수별로 성능을 측정하여 비교하였다.
- 엘보우 포인트(?)는 대략 10-20으로 판단하고 있음.
- Sequence-to-sequence 모델이 다양한 문서를 통해 정보를 잘 빼오고 있다고 판단함.
- 훈련시에 문서를 x개 만큼 뽑았다면, evaluation 때도 x개의 문서를 사용함.

(ex: 훈련할 때 50개의 문서를 뽑아서 한다고 했으면, 평가할때도 문서 50개를 뽑아서 답을 냄.)



Training 시 사용되는 문서 수 변동에 따른 성능 평가

- 논문에서는 evaluation때 사용되는 문서 수를 100으로 고정하고, 각 훈련시에 사용되는 문서 수에 변동을 줘서 성능 측정 실험을 진행함.

	NaturalQuestions		TriviaQA	
Training Passages	w/o finetuning	w/ finetuning	w/o finetuning	w/ finetuning
5	37.8	45.0	58.1	64.2
10	42.3	45.3	61.1	63.6
25	45.3	46.0	63.2	64.2
50	45.7	46.0	64.2	64.3
100	46.5	-	64.7	-

- 이것도 당연히(?) 훈련시에 더 많은 데이터를 가지고 최종 답변의 완성도와 퀄리티를 높이는 방법으로 학습을 하기 때문에 많을 수록 좋음.

** w/o finetuning – 파인튜닝 안함 w/finetuning – 파인튜닝함

- 파인튜닝: 훈련된 모델을 가져와서 100개의 문서로 1000번의 반복해서 finetuning을 진행했고, 이 결과 모델을 처음부터 새로 훈련하는것보다 훨씬 적은 데이터를 사용하면서도 성능을 개선시켰음.

예를 들어, NQ의 경우 데이터셋에서 46.0 EM 정확도를 달성하기 위해 100개의 문서를 훈련하는 경우 425 GPU 시간이 필요하지만, 미세조정을 통해 147 GPU 시간만으로도 동일한 수준의 정확도를 얻을 수 있습니다.

5. Conclusion

- 이 논문에서는 오픈 도메인 질문 응답에 대한 간단한 접근 방식을 연구함
  * 성능적으로 우수한 모습을 보임.
  * 문서의 확장 반영이 유연함.

- 목표 및 방향성:

일련의 과정을 더 효율적으로 만들어서 **문서 검색-> 답 생성**의 모든 과정을 하나의 모델로 학습할 계획.