Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

Patrick Lewis†‡, Ethan Perez?,
Aleksandra Piktus†, Fabio Petroni†, Vladimir Karpukhin†, Naman Goyal†, Heinrich Kuttler†,
Mike Lewis†, Wen-tau Yih†, Tim Rocktaschel†‡, Sebastian Riedel†‡, Douwe Kiela†
†Facebook AI Research; ‡University College London; ?New York University;
plewis@fb.com

발표자 안도형

Contents

Introduction

Introduction

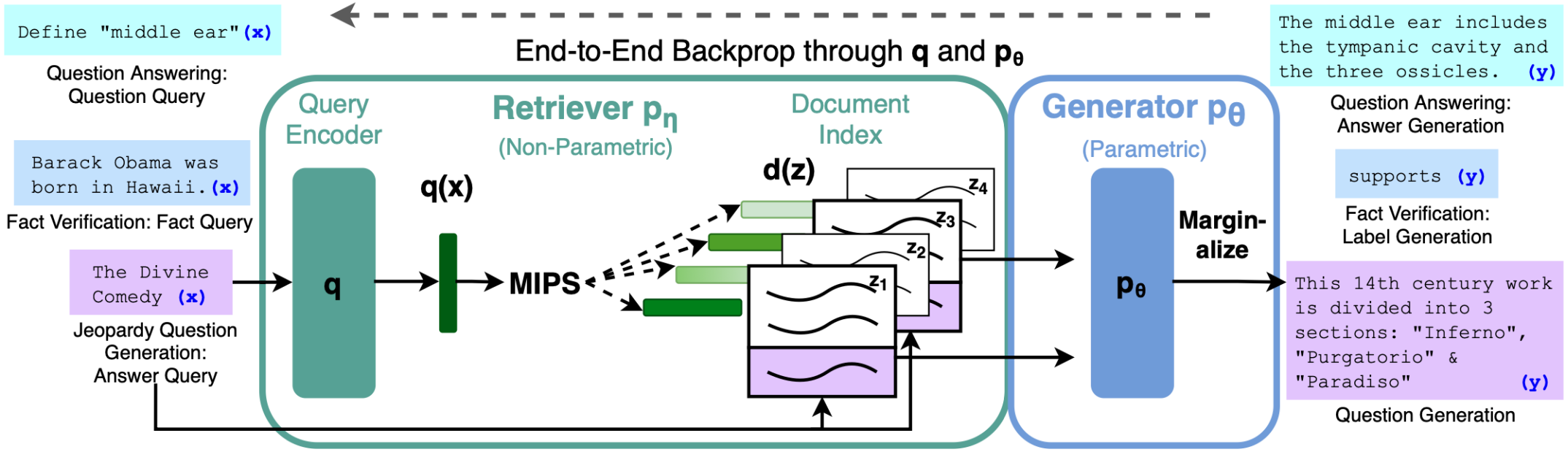
- 내부 parameter만을 사용하는 대규모 pre-trained된 언어 모델들은 지식에 접근하고 정확하게 조작하는 능력이 제한적이므로, Knowledge intensive task에서는 특정 architecture에 뒤쳐짐
- 새로운 지식에 대한 업데이트를 모델에 적용하는 것도 매우 제한적이며 예측한 결과에 명확한 Insight를 제공하지 못하며 **hallucination**을 생성할 수 있다는 단점 존재
- 이를 해결하기 위해 새로운 형태의 모델이 제시됨
    - Parametric memory + Non-Parametric memory

Introduction

- REALM과 ORQA 모델에서 parametric memory와 non-parametric memory를 결합해
  Open Domain extractive QA에서 좋은 결과를 보여줌
  - 하지만 정답이 document 내에 존재해 추출할 수 있는 경우에만 좋은 성능 보이며, document에 존재하지 않는 새로운 답변은 생성 불가
  - 또한 문서 하나와 일대일로 대응되지 않는 질문, 즉 복잡하고 복합적인 질문에는 답을 잘 하지 못하는 문제 존재
- 따라서 non-parametric memory를 활용함과 동시에 Generation task를 수행할 수 있는 BART를 generator로 활용하여 더욱 다양한 분야의 NLP task에서 활용할 수 있는 RAG 모델 제시
  - Retriever로 DPR, Generator로 BART 모델을 사용

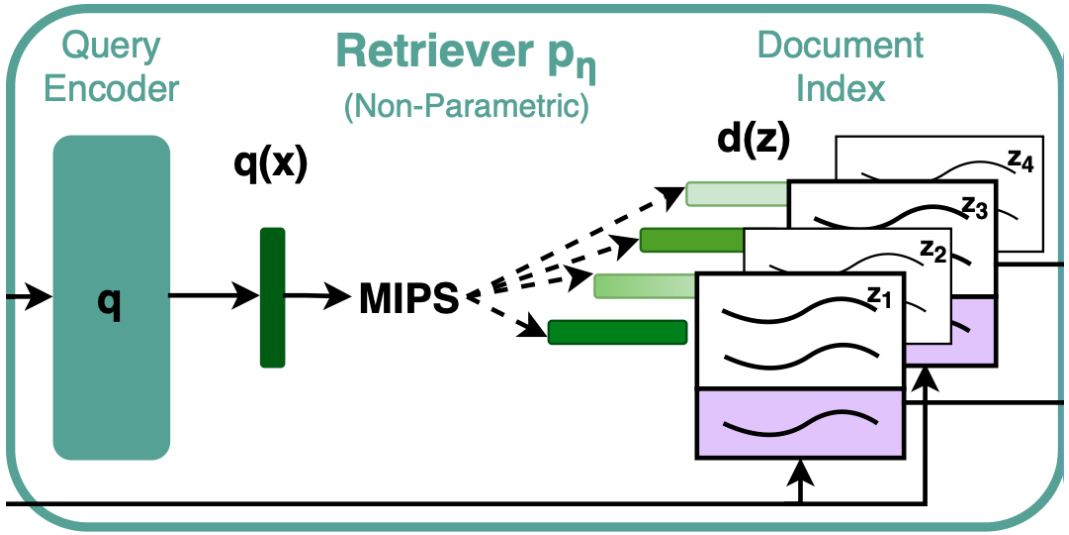
Methods

Model Architecture



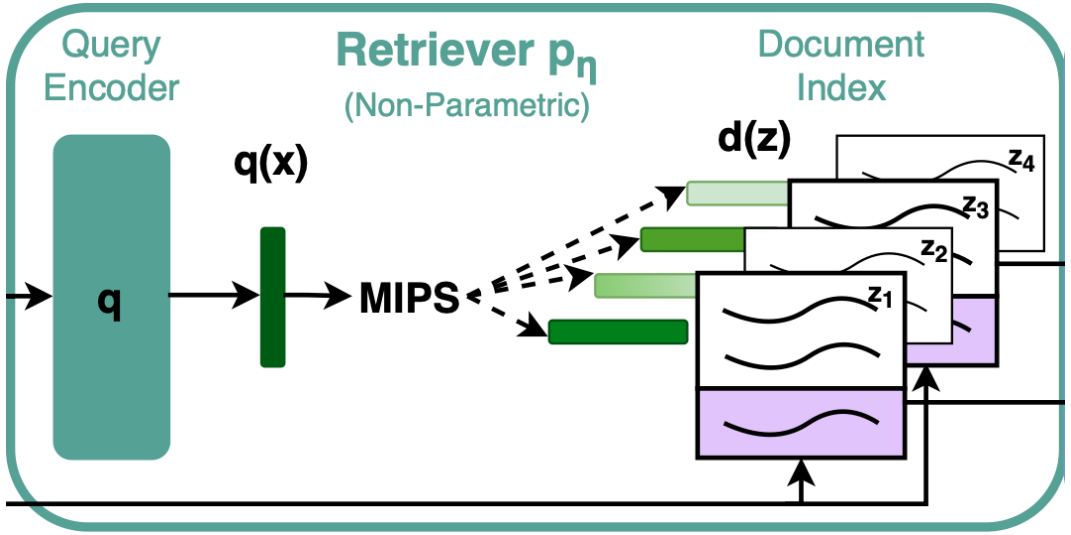
- 크게 두가지 모델로 나누어짐
  - Retriever : Question *x*를 기반으로 유사한 document *z*를 retrieve하는 모델
  - Generator : Retriever가 반환한 z와 x를 encoder에 넣고 decoder에서 answer y를 생성 하는 모델

Model Architecture - Retriever



- Retriever의 주된 역할은 주어진 입력에 대해 관련된 문서를 선택하고 추출
- RAG는 Retriever로는 **DPR**를 사용하였으며, DPR은 bi-encoder architecture를 사용
  - 두개의 Encoder는 **Bert_base** 모델을 사용하였으며, 하나는 **Query Encoder**로, 하나는 **Document Encoder**로 사용

Model Architecture - Retriever

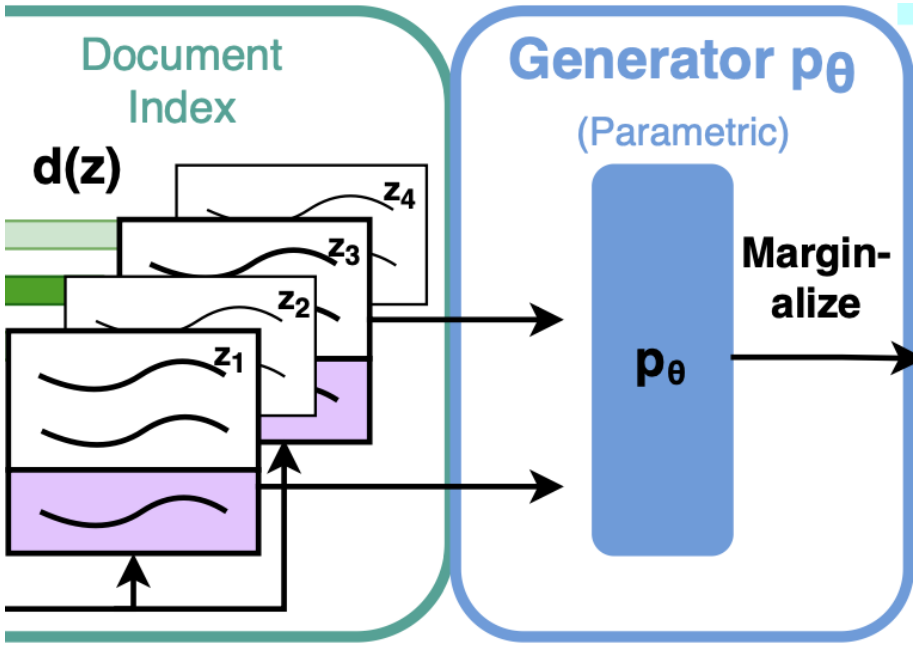


$$p_\eta(z|x) \propto \exp\left(\mathbf{d}(z)^\top \mathbf{q}(x)\right)$$

$$\mathbf{d}(z) = \mathrm{BERT}_d(z), \quad \mathbf{q}(x) = \mathrm{BERT}_q(x)$$

- Query Encoder : BERTq(x) | Document Encoder : BERTd(z)
- Question과 Document를 서로 다른 BERT를 통과해 [CLS] Token을 계산한 후 내적을 통해 내적값이 최고로 높은 document을 찾아가는 방법인 MIPS 알고리즘 활용
  - 우선 question을 BERT에 태운 후 [CLS] Token을 빼와서 query vector를 만듬
  - wikipedia 각 document 역시 BERT(question encoding한 BERT와 다른 객체)를 태워 [CLS] Token을 빼와서 DB를 구축

Model Architecture - Generator



- Generator로는 BART-large를 활용
- Passage를 기존 Query와 Concat하여 Generator Input으로 활용
- K개의 유사한 document를 어떻게 Concat해서 generate?
  - 두가지 방법의 marginalize 제시

Model Architecture – Generator (bart가 어떻게 decode 되냐)

RAG - Sequence

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

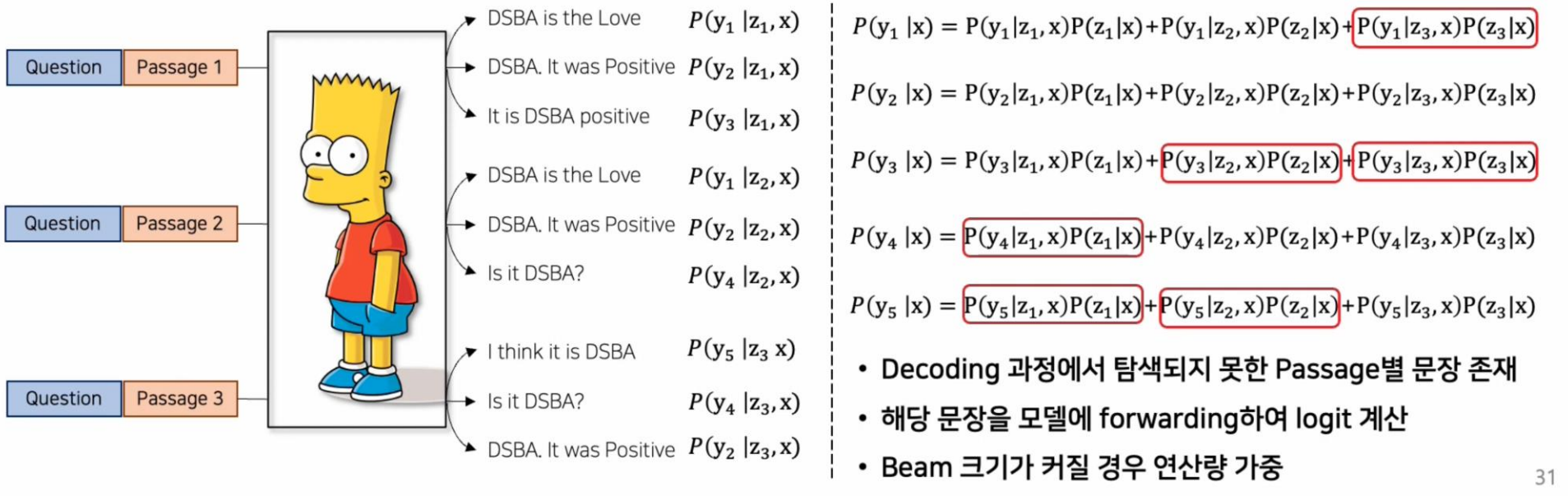
RAG - Token

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

- Marginalize 방식에 따라 두가지 모델 제안
  - **RAG – Sequence** : 각 passage 마다 별개로 complete sequence 생성, 최종 생성문 시점 단위로 marginalize
  - **RAG – Token** : 토큰 생성 때 마다 passage 별 분포 marginalize 해서 토큰 생성, 다음 토큰 생성 시 이전 marginalize 결과 이용

Model Architecture - Generator

RAG - Sequence

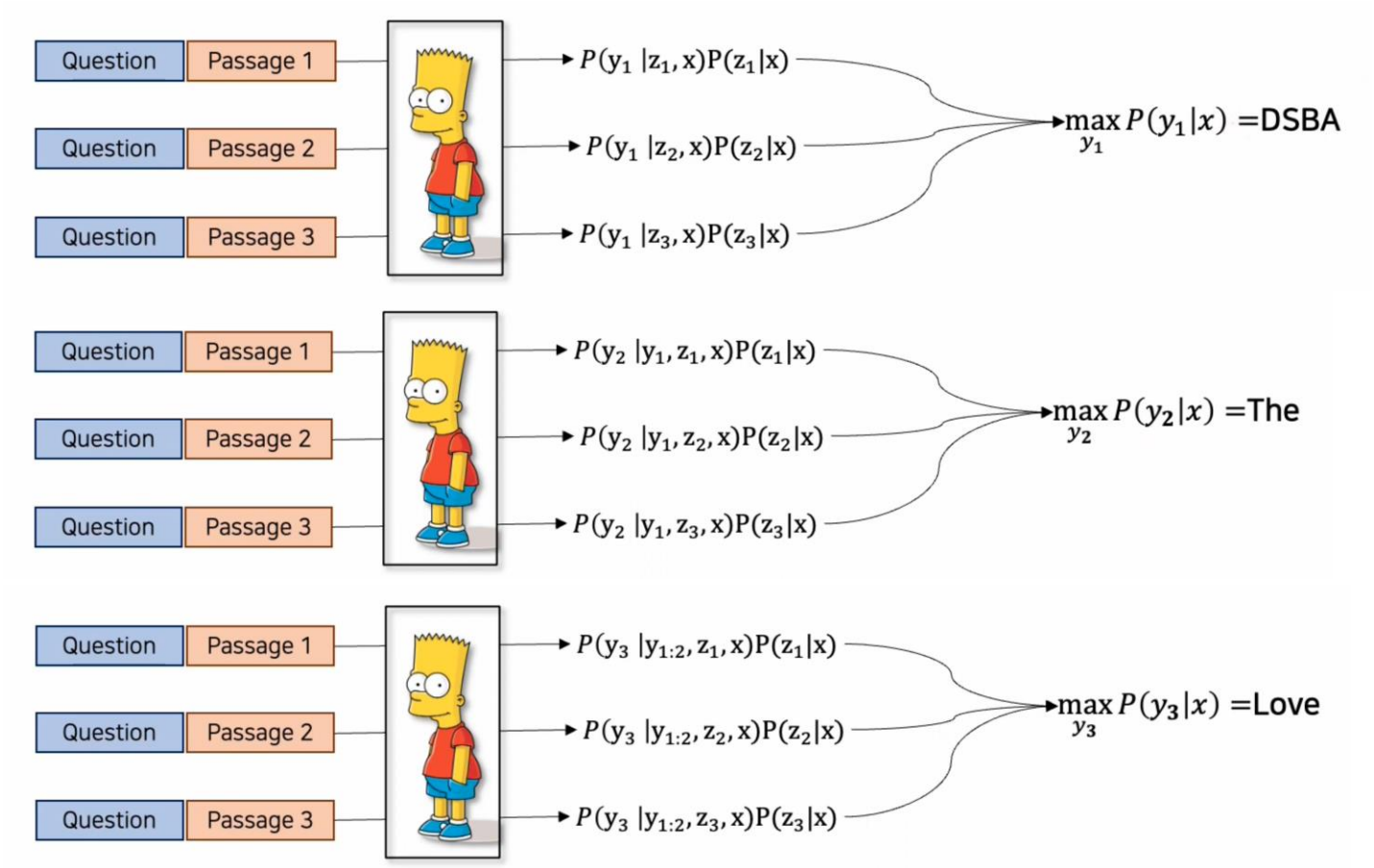


- 하나의 passage에 대해 끝까지 generation 후 모든 passage에 대해 marginalize
- 나타나지 않은 y에 대한 확률값을 계산하기 위해 문장을 다시 입력으로 forwarding 해주어 확률을 다시 구하고 더해주는 Thorough Decoding과 그냥 0으로 비워버리는 Fast Decoding 활용

출처 :
https://www.youtube.com/watch?v=qtOdvAQk6YU&t=1703s

Model Architecture - Generator

RAG - Token

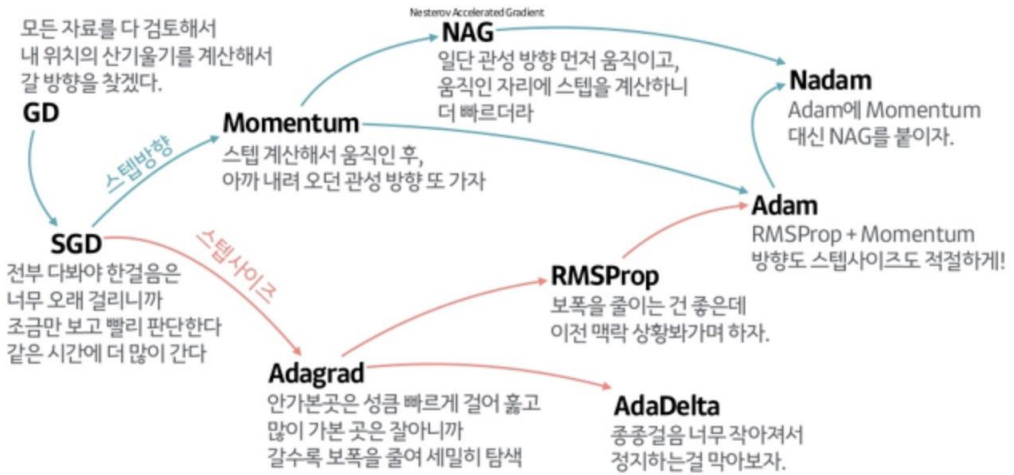


- 새로운 토큰을 생성할 때마다 모든 passage를 marginalize
- 모든 토큰에 대해 위 과정 반복
- RAG - Sequence와 달리 추가적인 연산 필요 x

출처 :
https://www.youtube.com/watch?v=gtOdvAQk6YU&t=1703s

Model Architecture - Training

- 매 question마다 유사한 K개의 document에 대한 정답 label 없이 훈련을 진행
- Document Encoder는 업데이트 하지 않고 fixed 상태로 두고 Query Encoder BERTq와 BART Generator 만 fine-tuning
    - 학습 중 Document Encoder를 업데이트하면 Docmuent Index를 정기적으로 업데이트 해야하므로 비용이 많이 소모
- 입력/출력 쌍 (x, y)가 주어지면 Adam과 stochastic gradient descent 활용하여 손실 최소화



Conclusion

Conclusion

Conclusion

- Read만 할 수 있던 기존 모델들과 달리 generate 가능한 모델로 바꾸어 기존 하이브리드 모델과 달리 더 다양한 NLP task에서 사용 가능한 모델 제시
- BART와 달리 non-parametric memory까지 사용하여 더 높은 성능 보임
- Non-parametric memory를 사용하여 지식의 업데이트 및 수정이 용이함

My Conclusion

- 외부 지식을 활용하는 하이브리드 모델은 기존에는 document에 있는 정보를 읽고 가져오는 것만 가능해 다양한 task에서 활용되지 못했지만, 이를 generator를 활용하여 더 다양한 Task에서 활용할 수 있게 된 것에 큰 의의가 있다고 본다

Background

Dense Passage Retrieval

- 초기 ODQA(Open-domain Question Answering)시스템들은 복잡하고 여러 구성요소로 이루어졌었지만 독해 이해 모델의 발전으로 간결한 두 단계의 프레임워크를 제안하게 됨.
  - Retriever: 답을 포함하는 passage를 선택
  - Reader: 검색된 passage에서 정답을 추출

- Reader모델을 중점적으로 다루는 것은 좋은 접근법이지만 이는 성능 향상에 한계가 있음
- 이는 Retrieval 방법의 개선이 필요하다는 것을 의미함

Background

Dense Passage Retrieval

- ODQA에서 Retrieval은 일반적으로 질문과 context를 Sparse vector로 표현하는 Sparse vector space 모델 TF-IDF, BM25 같은 방법을 사용하여 구현됨
- 하지만 Sparse vector를 사용하는 방식은 유사어나 동의어의 정보를 반영하기가 어렵다는 단점이 있음
- Dense 인코딩 방식을 사용하면 비슷한 의미를 가지는 단어도 가까운 벡터로 매핑 시킬 수 있음
- 그러나 이러한 Dense 인코딩 방식에서 일반적으로 좋은 Dense 벡터 표현을 학습하려면 많은 질문과 context쌍이 필요함
- 따라서 이전까지는 ODQA에서 TF-IDF, BM25보다 성능이 좋은 Dense retrieval방법이 없었음

Background

Dense Passage Retrieval

- 이후 제안된 ORQA[1]모델에서 Dense retrieval이 BM25를 능가함을 보여주며 여러 ODQA 데이터에서 SOTA를 달성함
- ORQA: retriever를 학습시킬 때 Inverse Cloze Task(ICT)로 추가적인 pretrain을 진행하고 그 후 question encoder와 reader는 질문과 답변 쌍을 사용해 fine tuning됨
  - Inverse Cloze Task(ICT) – 특정 passage에서 임의의 한 문장을 추출하고 이를 각 passage와 유사한지를 비교해 어떤 문맥에서 등장한 문장인지 예측

[1] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering.

Background

Dense Passage Retrieval

- 하지만 ORQA 모델에는 문제점이 존재해 논문에서는 이를 지적
  1. ICT pretraining의 연산 비용이 든다.
  2. 특정 passage에서 추출한 일반적인 문장이 question을 대체할 수 없다.
  3. context-encoder를 freeze시켜 fine tuning이 되지 않기 때문에 최적의 표현이 아닐 수 있다.

-> 따라서 위 논문에서는 Dense embedding model을 사용하되 추가적인 pretraining없이 질문과 답변 쌍만을 가지고 훈련을 하는 모델을 제안

Background

Dense Passage Retrieval

- ODQA의 retrieval component를 개선하는데에 집중
- DPR의 목적은 M개의 passage가 있을 때 모든 passage를 저차원 연속 공간(low-dimensional and continuous space)에 index하는 것
- 입력된 question과 관련된 top-k개의 passage를 효율적으로 검색하는 것

$$\text{sim}(q, p) = E_Q(q)^\intercal E_P(p).$$

- M : 21millon passage
- k : 20 ~ 100
- Question encoder: BERT base
- Passage encoder: BERT uncased
- d = 768

Background

Dense Passage Retrieval - Inference

- 학습 후 Inference를 진행할 때 FAISS라이브러리를 이용해 모든 passage를 passage encoder에 통과시켜 임베딩 벡터를 미리 계산하고 index함
- 새로운 질문 q가 주어질 때 가장 연관성이 높은 top-k개의 passage를 검색해서 빠르게 반환
- FAISS[2] - 대규모 dense vector 세트에 대한 빠른 유사성 검색을 위한 도구

[2] https://github.com/facebookresearch/faiss

Background

Training

- Training의 주요 목표는 서로 관련있는 Question-Passage 쌍이 서로 관련 없는 Question-Passage 쌍보다 더 가까운 거리(더 높은 유사성)을 가질 수 있는 벡터 공간을 만들기 위해 임베딩 함수를 학습시키는 것

$$
\begin{aligned}
& L(q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^-) \\
= \quad & -\log \frac{e^{\mathrm{sim}(q_i, p_i^+)}}{e^{\mathrm{sim}(q_i, p_i^+)} + \sum_{j=1}^{n} e^{\mathrm{sim}(q_i, p_{i,j}^-)}}
\end{aligned}
$$

- Q = 질문
- P+ = 질문과 관련 된(긍정적) passage
- P- = 질문과 관련 없는(부정적) passage

Background

Training – Positive and negative passages

- Positive example은 명확히 사용 가능한 반면 Negative example은 매우 큰 범위에서 선택해야한다.
- Negative example을 선택하는 방법은 인코더를 좋은 퀄리티로 학습하기 위해 중요하다.

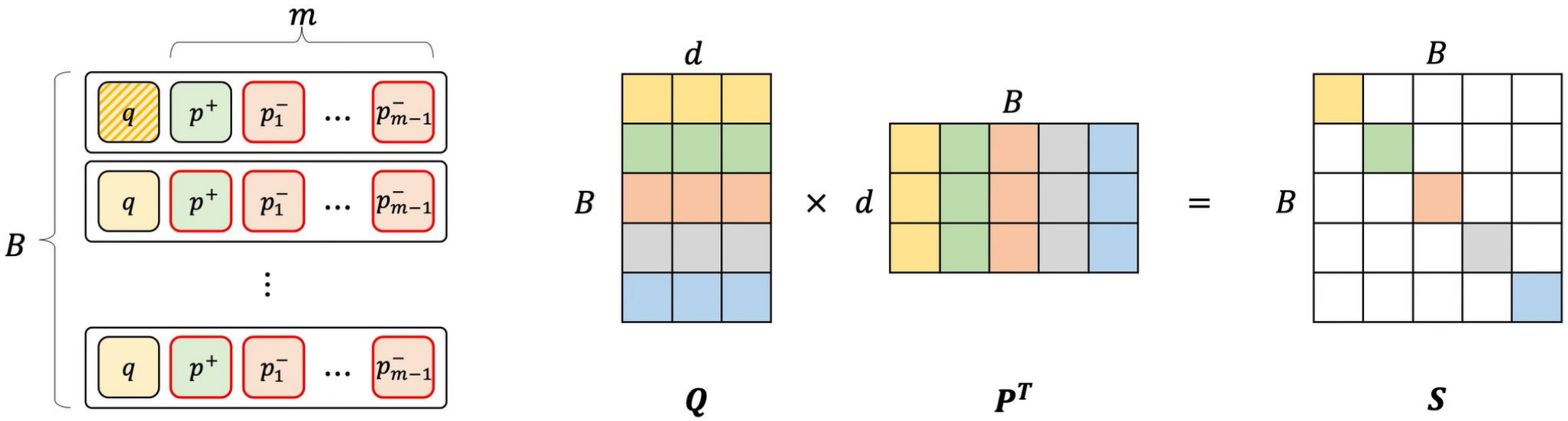
$$L(q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^-)$$

- **Random** : 코퍼스에서 임의의 패시지를 선택
- **BM25** : 답변을 포함하지 않으나 대다수의 질문 토큰과 일치하는 BM25로 검색된 상위 패시지들 선택
- **Gold** : 훈련 데이터 셋에 나타나는 다른 질문들과 연계된 Positve Passage들 선택

Background

Training – In-batch negatives

- 한 배치 내의 다른 example들을 negative example로 활용하여 모델이 positive example과 negative example을 구별하는 능력을 향상 시키는 데 도움을 준다



- 각 배치에서 모든 Question-Passage 쌍의 유사도를 계산하고,
  이를 통해 모델이 positive-passage와 negative passage를 구분하는 방법을 학습한다.

출처: https://www.youtube.com/watch?v=u4GE0sloBQU&t=883