Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

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

- Introduction
- Methods
- Experiments
- Results

• Introduction

이 논문에서는 사전학습된 parametric 메모리와 non-parametric 메모리를 언어 생성을 위해 결합하는 retrieval-augmented generation(RAG) 모델을 위한 범용 목적의 fine-tuning 방법을 제안.

RAG모델은 parametric 메모리로 사전학습된 seq2seq 모델을, non-parametric 메모리로는 사전학습된 neural retriever로 접근하는 위키피디아의 dense vector index를 사용한다.

두가지 RAG 공식을 제안하는데, 하나는 전체 생성 시퀀스에 대해 동일한 retrieved passage를 조건으로 사용하는 방식, 다른 하나는 각 토큰마다 다른 passage를 사용하는 방식이다.

제안 모델을 다양한 지식 집약적 NLP task에 대해 fine-tuning하고 평가했을 때, 3개의 open-domain QA task에서 기존 parametric seq2seq 모델과 task-specific한 retrieve-and-extract 아키텍처를 능가하는 SOTA 성능을 달성

- Pre-trained된 parametric 메모리(seq2seq)와 non-parametirc 메모리(dense retriever + 위키피디아 인덱스)를 결합한 RAG 모델 제안
- 두 가지 RAG 공식 : 전체 시퀀스에 동일한 문서 사용 vs. 토큰 별로 다른 문서 사용
- Open-domain QA에서 기존 모델 능가하는 SOTA 성능
- 언어 생성 task에서 BART보다 구체적이고 다양하고 사실적인 문장 생성
- Pre-training 없이도 강력한 성능을 보임
- Retrieval을 통해 지식에 직접 접근하고 업데이트 할 수 있는 장점

- Methods

RAG 모델 구조

•input sequence x를 이용해 text document z를 retrieve하고, 이를 추가적인 context로 활용해 target sequence y를 생성
•두 가지 주요 구성요소:
  - Retriever $p_\eta(z|x)$: input x가 주어졌을 때 text passage에 대한 확률분포 반환 (top-k개 문서). 파라미터는 η
  - Generator $p_\theta(y_i|x,z,y_{1:i-1})$: 이전 i-1개 토큰 $y_{1:i-1}$, 원래 input x, retrieved passage z를 context로 현재 토큰 y_i 생성. 파라미터는 θ

•Retriever와 generator를 end-to-end로 학습시키기 위해 retrieved document z를 latent variable로 취급

- Methods

RAG Sequence Model

•전체 시퀀스 생성에 동일한 retrieved document를 사용
•P(y|x) = Σ_z p_η(z|x) * p_θ(y|x,z), 즉 top-k retrieval 결과에 대해 marginalization 수행
  - p_η(z|x): retriever 확률
  - p_θ(y|x,z) = π_i p_θ(y_i|x,z,y_1:i-1): 전체 토큰에 대한 generator 확률의 곱

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

RAG Token Model

•각 target 토큰 y_i 마다 다른 latent document z_i를 사용할 수 있음
•P(y|x) = π_i Σ_z p_η(z|x) p_θ(y_i|x,z,y_1:i-1), 즉 각 토큰마다 top-k retrieval 결과에 대해 marginalization 수행

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

- Methods

Retriever 모델 - DPR

•bi-encoder 구조: query encoder BERT_q(x), document encoder BERT_d(z)
•p_η(z|x) ∝ exp(d(z)^T q(x))로 계산. d(z)와 q(x)는 각각 BERT_d(z)와 BERT_q(x)의 출력.
•사전학습된 DPR bi-encoder로 retrieverdhk document index 초기화

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

Generatort 모델 - BART

•BART-large 사용 (400M 파라미터)
•BART 입력으로 x와 retrieved document z를 concat해서 넣어줌

$$: p_\theta\left(y_i \middle| x, z, y_{1:i-1}\right)$$

- Methods

Training

- Retriever와 generator를 jointly 학습
  - 어떤 document를 retrieve해야 하는지에 대한 직접적인 supervision은 없음

- Fine-tuning 방식 사용
  - Input/output pair (x_j, y_j)로 구성된 training corpus 사용
  - Σ_j -log p(y_j|x_j) 를 최소화하는 방향으로 stochastic gradient descent (Adam) 수행

- Document encoder BERT_d는 고정 (Fixed)
  - Training 중에 document encoder를 update하려면 index를 주기적으로 새로 계산해야 함 (REALM에서처럼)
  - 그러나 RAG에서는 이 과정이 성능 향상에 필수적이지 않다고 판단
  - 따라서 document encoder (BERT_d)와 document index는 고정하고 fine-tuning에서 제외

- Query encoder BERT_q와 generator (BART)만 fine-tuning
  - Retriever의 query encoder BERT_q와 generator인 BART만 fine-tuning 과정에서 학습되도록 함

- Methods

Train & Decoding

•retriever와 generator를 retrieved document에 대한 직접적인 supervision 없이 end-to-end로 jointly 학습

•input/output pair(x_j, y_j)로 구성된 fine-tuning 데이터셋 사용

•Σ_j -log p(y_j|x_j) 최소화하도록 SGD-Adam으로 학습

•Document encoder와 index 고정, query encoder와 generator만 fine-tuning

•RAG-Token은 일반적인 beam search로 decoding 가능

•RAG-Sequence는 Thorough Decoding과 Fast Decoding 두 가지 decoding 방식 제안

즉, RAG는 retriever로 DPR을, generator로 BART를 사용하고, 두 가지 방식(Sequence/Token)으로 retrieved document를 활용해 target sequence를 생성한다.
Retrieval 결과에 대한 추가 supervision 없이 end-to-end로 모델을 학습시킬 수 있다는 것이 주요 특징.

- Experiments

  - Open-domain Question Answering (QA)
  - Abstractive Question Answering
  - Jeopardy Question Generation (QG)
  - Fact Verification

- Result

Table 1: Open-Domain QA Test Scores. For TQA, left column uses the standard test set for Open-Domain QA, right column uses the TQA-Wiki test set. See Appendix D for further details.

	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**

Table 2: Generation and classification Test Scores. MS-MARCO SotA is [4], FEVER-3 is [68] and FEVER-2 is [57] *Uses gold context/evidence. Best model without gold access underlined.

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		

Table 1:
- Closed Book 방식인 T5 모델 대비 Open Book 방식인 REALM, DPR, RAG 모델들이 전반적으로 우수한 성능을 보임
- RAG 모델들이 Natural Questions(NQ), TriviaQA(TQA), WebQuestions(WQ), CuratedTREC(CT) 데이터셋에서 가장 높은 성능 달성
- 특히 RAG-Sequence 모델이 대부분의 데이터셋에서 SOTA 기록 (TQA에서는 Wiki 버전 Test set에서만 SOTA)

Table 2:
•MS MARCO와 FEVER 데이터셋에서는 Gold context/evidence를 사용한 SOTA 모델이 가장 높은 성능
•하지만 Gold context를 사용하지 않는 모델 중에서는 RAG 모델들이 가장 우수한 성능 달성
•Jeopardy에서는 RAG-Token, MS MARCO와 FEVER에서는 RAG-Sequence가 높은 성능
•BART 대비 RAG 모델들이 대부분 더 좋은 성능을 보여주고 있음

Thank You