⭐RAG 논문 리뷰

1. Introduction

대규모 사전학습 언어 모델들이 파라미터에 사실적 지식을 저장할 수 있고, downstream NLP task에 fine-tuning되면 좋은 성능을 보임이 입증되었으나, 이런 모델들은 지식에 접근하고 정확히 다루는 능력이 제한적이어서, 지식 집약적인 task에서는 task-specific한 아키텍처에 비해 성능이 뒤처짐.

explicit하고 non-parametric한 메모리에 differentiable하게 접근할 수 있는 pre trained 모델들은 extractive downstream task에 대해서만 연구되어 왔다.

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

2. Methods

2-1. 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_η(z|x): input x가 주어졌을 때 text passage에 대한 확률분포 반환 (top-k개 문서). 파라미터는 η
  - Generator p_θ(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

2-2. RAG-Sequence 모델

- 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

2-3. RAG-Token 모델

- 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
- y_i z_i

2-4. 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

2-5. Generatort 모델 - BART

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

2-6. 학습 및 추론 과정

- 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.

3. Experiments

3-1. Open-domain Question Answering(QA)

- 질문과 답변을 Input-output text pair (x, y)로 간주하고 답변의 negative log-likelihood를 최소화하도록 직접 학습
- Natural Questions (NQ), TriviaQA(TQA), WebQuestions(WQ), CuratedTrec(CT) 데이터셋 사용
- 작은 데이터셋인 CT와 WQ는 NQ로 학습한 RAG 모델로 초기화 후 fine-tuning
- 기존의 extractive QA 및 retrieval 없이 생성만 하는 closed-book QA 방식과 비교
- Exact Match(EM) score로 평가p

3-2. Abstractive QA

- MS MARCO NLG 데이터셋 사용(질문, 관련 지식, 자연어 답변으로 구성)

- 제공된 지식 없이 오직 질문-답변 쌍만 사용해 open-domain abstractive QA로 취급

3-3. Jeopardy Question Generation (QG)

- Jeopardy 퀴즈쇼의 답변으로부터 퀴즈 질문 생성하는 task
- SearchQA 데이터셋 활용(질문-답변 쌍 포함)
- Sequence-to-sequence 방식으로 모델링
- BART 모델과 비교
- SQuAD 튜닝된 BLEU-1, Q-BLEU-1 metric으로 자동평가
- 사실성(Factuaility)과 구체성(Specificity)에 대한 Human evalutiaion 수행

3-4. Fact Verification

- FEVER 데이터셋 사용
- Wikipidia 문서를 근거로 주어진 statement가 supported/refuted/not enough info 중 무엇인지 판단
- Retrieval 수행 후 entailment reasoning으로 진위 여부 판단해야 하는 복합적인 task
- Retrieval 정답에 대한 추가 supervision 없이 claim-label 쌍으로만 학습
- Statement와 label을 input-ouput pair로 간주하고 직접 학습
- 3-way classification(supports, refutes, not enough info)와 2-way classification(supports, refutes) 모두 수행
- Label accuracy로 평가

4. Results

4-1. Open-domain Qustion Answering

- Table 1에 RAG와 기존 방법들의 성능 비교 결과 정리
- RAG가 모든 open-domain QA dataset에서 새로운 SOTA 기록
- “Open book” 접근 방식의 유연한 생성 능력과 “Closed book” 방식의 우수한 성능을 결합

- 특수한 “salient span masking” 전략 없이도 강력한 성능을 보임 (vs. REALM, T5-SSM)
- BERT 기반 re-ranker나 extractive reader 없이도 우수한 성능 (vs. DPR)
- Retrieval 문서에 정답이 없는 경우에도 11.8%의 정답률을 보인 (NQ 데이터셋)

4-2. Abstractive Question Answering

- Table2에 Open-MS MARCO의 RAG 성능
- RAG-Sequence가 BART 대비 BLEU 2.6, Rough-L 2.6 높은 성능
- Gold passage 없이도 SOTA에 준하는 성능 (전체 1위 모델은 Gold passage 활용)
- 직관적으로도 RAG가 BART대비 사실에 기반하고 적은 hallucination을 보임(Table 3 예시)

4-3. Jeopardy Question Generation

- Table 2에 Jeopardy QG결과 (BLEU-1, Q-BLEU-1)
- RAG-Token이 BART, RAG-Sequence 대비 높은 성능
- 사람 평가 결과 RAG-Token이 BART 대비 사실성, 구체성 크게 개선(Table4)
- Posterior 분석을 통해 문서 별 token 생성 기여도 확인(Figure2)
- 단일 문서가 아닌 여러 문서의 정보를 결합해 생성하는 특성 확인

4-4. Fact Verification

- FEVER에서 파이프라인 방식 SOTA 대비 3-way 4.3%, 2-way 2.7% 낮은 성능 (Table2)
- 하지만 retrieval에 대한 추가적인 supervision 없이도 견줄만한 성능
- Clain과 가장 관련있는 Wikipedia article을 잘 찾아내는 것 확인(전체의 90%)

4-5. Additional Results

- RAG가 생성한 결과가 BART에 비해 더 diverse함을 확인(Table5)
- Test시 더 많은 문서를 retrieve할수록 RAG-Sequence는 monotonic한 성능 개선을 보이나, RAG-Token은 특정 개수에서 saturation되는 경향(Figure 3)
- 2018년 Wikipedia index로 학습된 모델을 2016년 index로 교체하면 2016년 당시 지식에 맞게 QA 수행이 가능함을 확인(outdated 지식 해결 가능)

5. Related Work

5-1. Single-Task Retrieval

- 다양한 NLP task에서 retrieval의 효용성이 개별적으로 입증
- Open-domain QA, fact-checking, fact completion, long-form QA, Wikipedia article generation, dialogue, translation, language modeling 등
- 본 연구는 단일 retrieval 기반 아키텍처로 여러 task를 아우를 수 있음을 보임

5-2. General-Purpose Architectures for NLP

- Retrieval 없이도 단일 사전학습 언어모델로 다양한 NLP task 수행 가능
- GPT-2는 단일 left-to-right 언어모델로 discriminative/generative task 모두 수행
- BART, T5 등의 seq2seq 모델로 성능 개선
- 본 연구는 retrieval 기반 방식으로 기존 general-purpose 아키텍처가 수행 가능한 task의 범위를 넓히고자 함

5-3. Learned Retrieval

- 딥러닝 기반 retrieval 연구가 활발히 진행중
- Downstream task 성능 향상을 위한 retrieval 최적화 연구들이 수행됨 (search, RL, latent variable 등 활용)
- 그러나 대부분 단일 task에 특화된 반면, 본 연구는 동일한 retrieval 구조로 다양한 task에 적용 가능함을 보임

5-4. Memory-based Architectures

- RAG의 문서 인덱스를 외부 메모리로 간주할 수 있음
- 기존 memory network 연구들과 유사한 면이 있으나, RAG는 distributed representation이 아닌 raw text 형태의 메모리를 사용하는 점이 특징
- 이는 해석 가능성을 제공하과, 메모리 업데이트를 용이하게 함

5-5. Retrieve-and-Edit approaches

- Retrieve-and-edit 방식과 유사한 면이 있음 (유사한 샘플을 검색 후 수정하여 활용)
- 기계번역, semantic parsing 등에서 retrieve-and-edit 방식이 활용된 바 있음
- 그러나 RAG는 retrieved sample을 직접 수행하기 보다는 여러 sample을 종합하여 생성한다는 점, latent retrieval을 활용한다는 점 등에서 차이가 있음

6. Discussion

RAG 연구의 의의와 향후 연구 방향

RAG 장점 및 기여

- Parametric memory와 non-parametric memory를 결합함으로써 SOTA 성능 달성
- 순수 parametric model 대비 사실에 기반하고 구체적인 generation 결과를 보임 (Jeopardy QG에서 human evaluation으로 입증)
- Retrieval에 기반함으로써 해석 가능성(interpretability)을 제공하고, index 교체를 통해 지식 업데이트가 용이함
- 다양한 task에 범용적으로 적용 가능(QA, QG, fact-checking 등)

이러한 장점을 바탕으로 RAG는 실제 응용에서 긍정적인 영향을 미칠 수 있음

- 의료 분야등 전문지식이 필요한 영역에서 활용 가능
- 업무 효율성 증대에 기여 가능

한편, Wikipedia가 완벽할 수 없고 편향이 존재할 수 있다는 점, 언어 모델의 misuse 가능성(fake news 생성, 스팸 메일 생성 등)등 negative social impact에 대해서도 언급하면서, 이에 대한 대응 방안 모색이 필요함을 지적하고 있다.

향후 연구 방향으로는 pre-training 단계에서부터 retriever와 generator를 jointly 학습시키는 방법을 제안. BART에서와 같이 denoising autoencoding이나 새로운 objective 설계를 통해 더 강력한 모델을 만들 수 있을 것으로 전망한다.

아울러 parametric memory와 non-parametric memory의 상호작용 메커니즘과 최적의 조합 방법 등 RAG를 발전시킬 수 있는 다양한 연구 질문들이 제기되고 있다.