[논문 리뷰] REPLUG: Retrieval-Augmented Black-Box Language Models

이번 게시물에서는 Retrieval-Augmented model을 GPT-3와 같이 API로만 공개되어, 내부 구조에 접근 불가능한 상황에서도 사용할 수 있게끔 연구한 논문인 REPLUG: Retrieval-Augmented Black-Box Language Models에 대해 다뤄보겠다.

해당 논문은 워싱턴 대학교에서 박사 과정을 진행중이신 민세원님과 카이스트의 서민준 교수님께서 연구에 참여하신 논문이다.

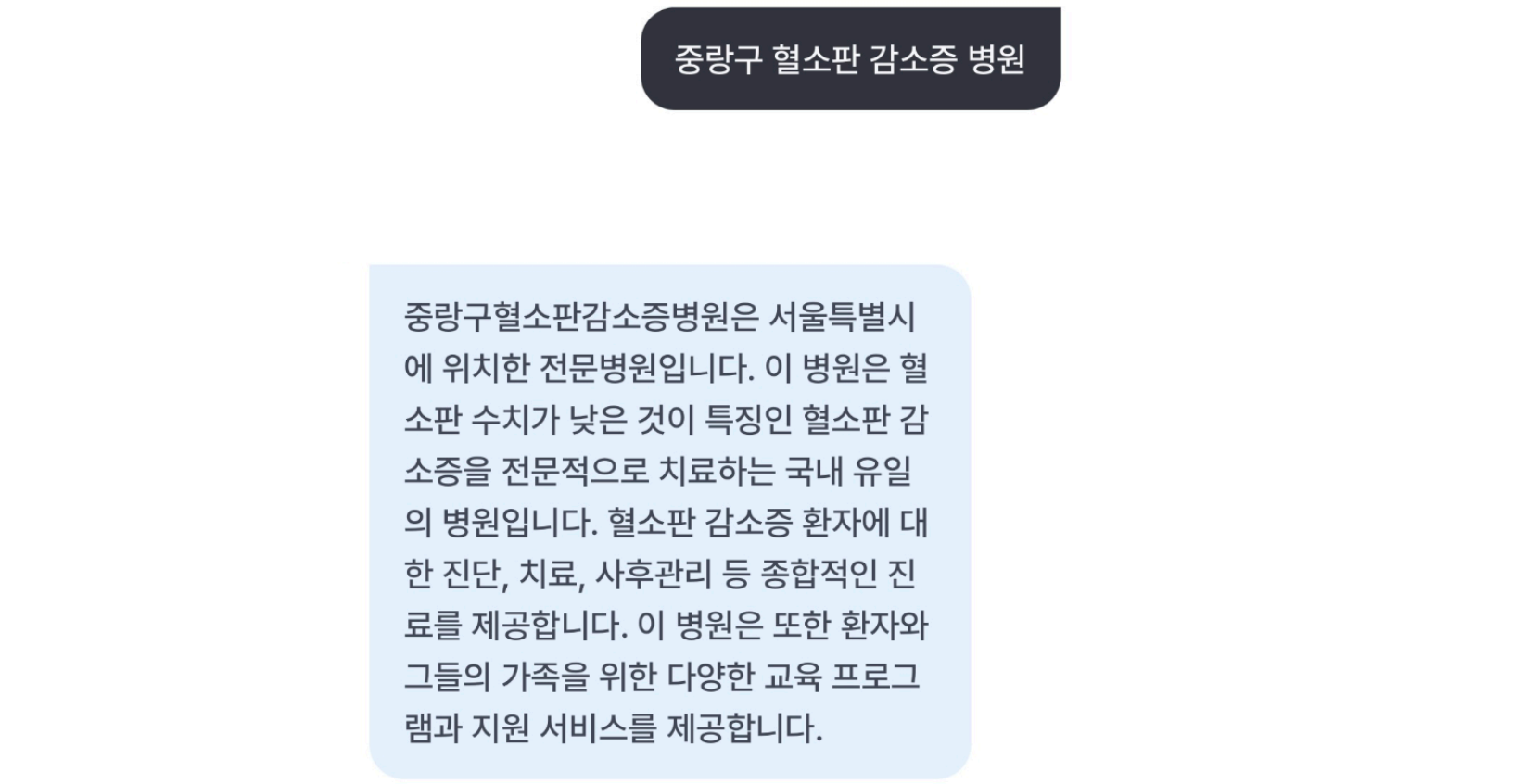
원문 링크는 다음과 같다.

REPLUG: Retrieval-Augmented Black-Box Language Models

Introduction

GPT-3와 같은 LLM들은 다양한 NLP task에서 좋은 성능을 보여왔다. 그러나, 이러한 LLM들은 hallucination이라는 큰 문제를 보이기도 한다.

Hallucination이란, **model이 사실과 다른 응답을 생성하는 문제이다**. 아래 사진을 보자



최근 GPT-3(Instruct-GPT) API를 사용하여 챗봇 서비스를 운영하고 있는 모 플랫폼을 사용해본 사진이다.

물론, 사람처럼 잘 말하고 만족스러운 답변을 주는 경우가 대다수이지만, query에 대한 output을 산출할 때, 특정 정보가 필요한 경우에는 세상에 없는 사실을 마치 사실인 것처럼 대답하는 모습을 볼 수 있다. (중랑구 혈소판 감소증 병원은 세상에 없는 병원이다.)

이러한 hallucination 문제를 해결하기 위해, Retrieval-augmented language model 관련 연구가 활발히 진행중에 있다. 이는 외부 database로부터 knowledge를 받아와 특정 정보가 필요한 경우, **즉 knowledge-intensive task에서 knowledge를 보충하여 hallucination 문제를 줄여준다는 개념**이다. 아래 사진을 보자

Context	**Topic**: Thierry Henry **Apprentice**: Thierry Henry is one of my all time favorite players. What about you?
Ground Truth Wizard Response	He was good. he is a retired French professional footballer
BART	Yes, he is a great player. He was born in 1931 and played for England's youth teams.
RAG-Token	I love him too! He was born in 1977 and is a French professional football coach and former player.
FiD-RAG DPR	I love him too! He is the current manager of French club Monaco. He was born in 1977.
Context	**Topic**: Elvis Presley **Wizard**: Oh baby.... Elvis Presley is truly The King of Rock and Roll! **Apprentice**: yes...who doesn't love Elvis Presley and his music? **Wizard**: Seriously. I have been planning a vacation to Graceland myself. I have family in Tennessee so it could be good. **Apprentice**: I would love to tour his home. Which one of his songs is your favorite?
Ground Truth Wizard Response	I always liked "Hounddog." My grandpa heard Elvis sing it live in Memphis.
BART	I love all of his albums. I think my favorite is "Love Me Do" which came out in 1999.
FiD-RAG DPR	I really like "Heartbreak Hotel" which was released in 1956 and became a number one hit in the United States.

Table 4: **Hallucination in (Non-)Retrieval-Augmented Models.** Examples of model outputs on the Wizard of Wikipedia Test set, unseen distribution; the retrieval-augmented models use BART as a base seq2seq model. Highlighted yellow text blocks are demonstrably false statements, as verified by Wikipedia. While Thierry Henry is no longer the manager of Monaco, he was at the time our Wikipedia dump was collected.

LLM은 아니지만, 형광색으로 표시된 결과가 기존 LM의 방법론인 BART의 결과이고, RAG, FiD-RAG DPR과 같은 model들은 앞서 말한 retrieval-augmented language model이다.

BART의 결과에서는, 앙리가 1931년생 잉글랜드 축구선수라는, 사실과 다른 output을 산출한 것과 달리, retrieval-augmented language model은 실제 정보를 아주 잘 산출해 낸 것을 확인할 수 있다.

이처럼, hallucination 억제를 위한 retrieval-augmented language model 연구는 REALM, RAG, FiD, ATLAS 등 활발하게 이루어지고 있다.

그러나, 이러한 기존 retrieval-augmented language model 방법론들은 retriever와 함께 붙어있는 LM부분의 내부에 접근이 가능해야만 하며, 이 부분도 retriever와 함께 한번에 train 시켜야 하기 때문에 최근 GPT-3와 같은, API로만 제공되는 LLM, 즉 black box LM에 대해서는 적용하기 힘들다는 한계가 존재하였다.

그래서, **본 연구에서는 이러한 LM 부분을 fixed-black box로 두고, retrieval에 대해서만 훈련하며, LM과 retriever을 탈부착하는 개념으로 둔 새로운 retrieval-augmented LM framework인 REPLUG(Retrieve and Plug)를 제안한다.**

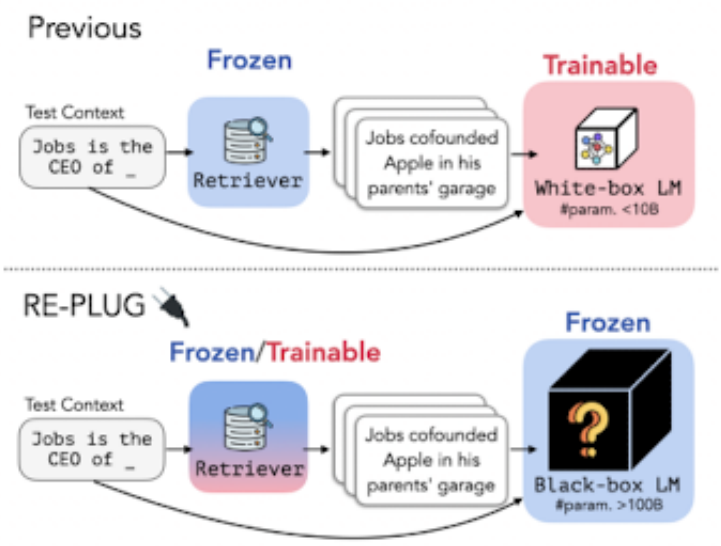


Figure 1. Different from previous retrieval-augmented approaches (Borgeaud et al., 2022) that enhance a language model with retrieval by updating the LM's parameters, REPLUG treats the language model as a black box and augments it with a frozen or tunable retriever. This black-box assumption makes REPLUG applicable to large LMs (i.e., >100B parameters), which are often served via APIs.

REPLUG는 input context x를 받으면, retriever를 이용하여 external corpus에서 x와 관련성이 높은 documents들을 retrieve 하고, 이러한 retrieved documents들은 input context 앞에 prepend 되어 black-box LM에 input으로 들어가게 된다.

다만, LM의 경우 input으로 받을 수 있는 sequence의 길이가 제한되어 있다는 문제가 있기에, 모든 document를 prepend할 수 없다는 문제가 있는데, 논문에서는 이를 해결하기 위해 새로운 ensemble scheme를 고안하였다고 한다.

해당 ensemle scheme에 대해 간략하게 말해보자면, retrieved documents들 각각 별개로 input context 앞에 prepend되고, 이렇게 만들어진 여러 개의 **prepended context가 병렬적으로 black-box LM에 input으로 들어간 뒤, 그에 따라 산출된 여러 개의 output을 ensemble** 하여 최종 결과를 산출하는 원리이다. 이에 대해서는 후반부에 더 자세하게 다루도록 하겠다.

이와 더불어, 논문에서는 REPLUG framework 내부의 retriever가 black-box LM을 일종의 supervision-signal로 활용하여 학습되게 하는 새로운 training scheme인, REPLUG-LSR(REPLUG with LM-Supervised Retrieval)을 제안한다.

해당 방법론에 대해서도 후반부에 더 자세히 설명하겠지만, 먼저 간단하게 소개해보자면 black-box LM을 retriever로부터 산출된 retrieved documents들의 score function으로 사용하는 것이다. (이때, LM 부분의 parameter는 update되지 않으며 retriever 부분만 update 된다.)

본 논문에서 제안하는 요소들을 정리해 보자면 아래와 같다.

- **새로운 retrieval-augmented language model framework인 REPLUG 제안**
- **Black-box LM을 retriever 부분의 score function으로 활용하여 training 하는 scheme, REPLUG-LSR 제안**

이제부터 논문에서 제안하는 요소들에 대해 보다 자세히 서술해 보도록 하겠다

Background

논문에서는 본 연구의 배경이 된 선행 연구들을 크게 2가지로 나누어 서술한다.

Black-box language model 파트와 retrieval-augmented model 파트가 그것인데, 하나씩 간단하게 살펴보고 넘어가겠다.

Black-box language Models

GPT-3와 같은 model들은 API로만 풀려있고, model 내부의 parameter등과 같은 요소에는 접근할 수 없다. 논문에서는 이를 black-box API라고 표현한다.

이와 다르게, BLOOM, OPT-175B, 그리고 논문에서 언급하진 않았지만 Stable LM과 같이, open source로 개방된 model들도 존재는 하지만, 이를 훈련시키기 위해서는 매우 많은 computational resource를 필요로 한다.

논문에서는 model 내부에 접근할 수 없는 black-box 문제, 혹은 높은 computational resource를 필요로 한다는 문제를 해결하고자 **black-box setting에서 retrival-augmentation을 효과적으로 진행하는 방법에 대해 연구**하였다고 밝힌다.

Retrieval-augmented Models

Retrieval-augmented language model이란, 기존 language model 내부에 retrieval 개념을 적용하여, 특정 task를 수행할 때 model parameter 내부에 저장된 knowledge뿐만 아니라 외부 knowledge를 활용하는 개념이다. 이러한 특징은 최근 LLM에서 많이 보이는 hallucination 문제를 억제할 수 있다는 장점이 있다.

이러한 retrieval-augmented language model은 크게 두 부류로 연구되어 왔는데, 두 부류 모두 black-box LM에는 적용하지 못한다는 한계가 존재했다.

경사 하강법을 통한 parameter update가 retriever뿐만 아니라 LM부분에도 적용되거나, LM의 internal representation에 접근 가능해야 하는 구조적인 한계가 있었기 때문이다.

저자들은 논문에서 제시하는 REPLUG는 이러한 기존의 한계를 극복하고, black-box LM에도 적용 가능한 retrieval-augmented language model이라고 소개한다.

REPLUG

그렇다면, REPLUG는 기존의 한계를 어떻게 극복할 수 있었을까?

우선 REPLUG의 대략적인 flow chart를 봐보자

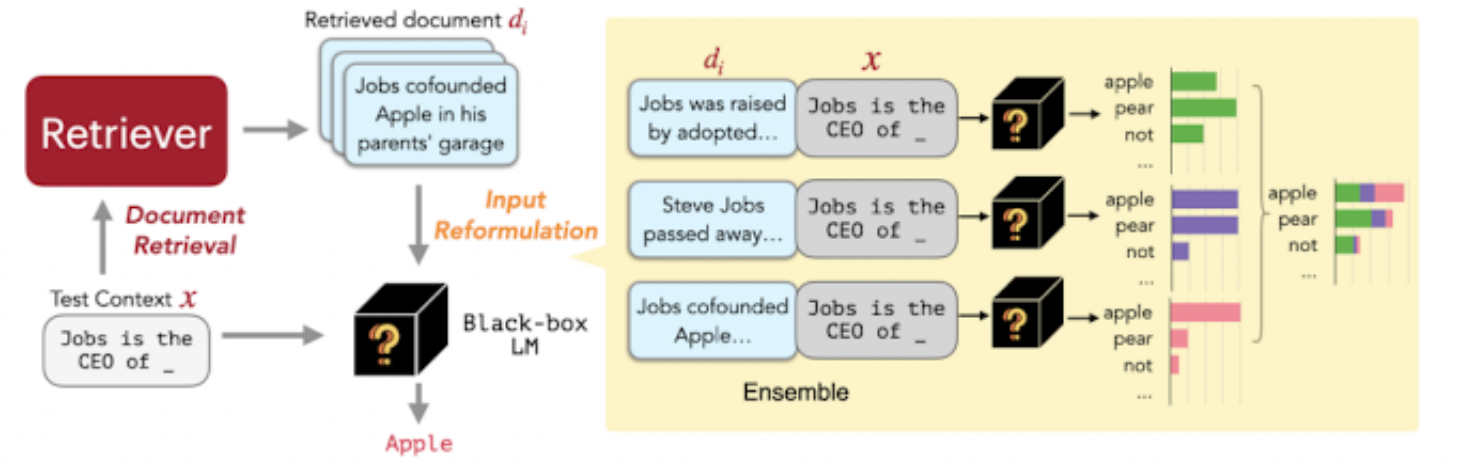


Figure 2. **REPLUG at inference** (§3). Given an input context, REPLUG first retrieves a small set of relevant documents from an external corpus using a retriever (§3.1 *Document Retrieval*). Then it prepends each document separately to the input context and ensembles output probabilities from different passes (§3.2 *Input Reformulation*).

이에 대해 크게 두 과정으로 나눠보자면

1. input text가 주어졌을 때, external corpus에서 document를 retrieve (Document Retrieval)
2. 각 document와 input context를 concat 하고, 병렬적으로 LM에 input으로 넣은 뒤 결과를 ensemble (Input Reformulation)

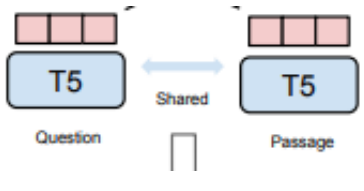
이에 대해 하나씩 살펴보도록 하겠다.

Document Retrieval

앞서 간단히 소개한 것처럼, 이 과정은 input text가 주어졌을 때, external corpus에서 document를 retrieve 하는 과정이다.

먼저, input text x가 주어지게 되면, retriever는 external corpus $\mathcal{D}$중에서 input text와 x 관계성이 있는 document set $\{d_1 \dots d_m\}$을 가져오게 된다.

이때, dual encoder에 기반한 dense retriever를 사용하게 되며, 이는 input x와 document d를 각각 encode 한다.



dual encoder의 경우, 위의 그림과 같은 구조이다. input x와 document d를 동시에 encode 하는 것이 아닌, input x따로 encode, document d 따로 encode인 것이다.

본 연구에서는, encoder가 document d에 대해 encode를 진행하고, 각 token 위치 별 last hidden representation을 mean pooling 하여 embedding $\mathbf{E}(d)$를 만든다. 이후, 금방 사용되었던 encoder가 input context x에도 동일한 방법론으로 적용되어 $\mathbf{E}(x)$를 만들어낸다.

이 두 $\mathbf{E}(x)$와 $\mathbf{E}(d)$ 사이의 유사성을 구하여 높은 몇 개의 document가 선택되게 되는데, 이때 유사성은 cosine similarity를 사용한다고 한다.

$$s(d, x) = \cos(\mathbf{E}(d), \mathbf{E}(x))$$

이후 유사도가 높은 top-k개의 document들을 선택하게 된다.

또한 효율적인 retrieval을 위해, model은 각 document별 embedding $\mathbf{E}(d)$을 미리 계산해 놓고, 해당 embedding들에 대해 FAISS index를 미리 구축해 놓는다.

(FAISS index란, 각 vector 별 유사도를 구할 때 더욱 효과적으로 구할 수 있게끔 해주는 알고리즘이다.

이에 대해 간략하게 언급하자면, vector(여기서는 $\mathbf{E}(d)$)들 간 유사도(혹은 상호 상관관계)를 미리 구해놓고, query vector와 미리 계산해 놓은 vector들과 유사도 검색이 필요한 경우 미리 구해놓은 유사도를 통해 더욱 빨리 유사한 vector를 검색할 수 있게끔 하는 기법이다.

쉽게 말하면 유사도 측정 대상이 되는 vector끼리 군집을 만들어놓고, 처음부터 끝까지 다 순회하면서 계산할 필요 없이 가장 비슷한 집단부터 측정을 하는 것이다.

FAISS의 경우 GPU 연산을 지원하기에 보다 빠른 계산을 기대할 수 있으며, 기존 연구들 중에서 보였던 MIPS와 같은 기법들도 이와 유사한 기법들이다.)

Input Reformulation

윗 단계에서 retrieved 된 top-k documents들은 input context x에 대해 풍부한 knowledge를 담고 있다. 결과적으로 LM으로 하여금 보다 우수한 output을 산출하게끔 한다.

그렇다면, 이러한 retrieved top-k documents들을 어떻게 LM에 넣어줘야 할까?

매우 간단하게 생각해 보면, 모든 top-k documents들을 input context x 앞에 붙여줌으로써 LM의 input에 풍부한 정보를 보충해 주는 방법이 있다.

그러나, 이와 같은 방법은, LM의 context size의 제한으로 인해 실행 불가능한 방법이다. 논문에서는 이러한 문제를 해결하기 위해 앞서 잠깐 소개한 ensemble strategy를 사용한다.

우선, top-k documents에서, k가 5라고 해보자. 그러면 5개의 documents가 retrieve 될 것이다.

이후 각각의 document에 대해 개별적으로 input context x앞에 prepend 한다. 이는 아래와 같은 형태가 된다.

> $d_1\ x$
> $d_2\ x$
> $d_3\ x$
> $d_4\ x$
> $d_5\ x$

이렇게 만들어진 input representation을 LM에 개별적으로 통과시키면, 5개의 개별적인 output probailities가 나오게 될 것이다.

이 과정은 아래와 같은 수식으로 표현할 수 있다.

$$p(y \mid x, \mathcal{D}') = \sum_{d \in \mathcal{D}'} p(y \mid d \circ x) \cdot \lambda(d, x),$$

이때, D'는 retrieve 된 k개의 document들의 집합이다. ○는 앞서 언급한 document d와 input context x의 prepend를 의미하며, 뒤에 붙은 $\lambda(d, x)$는 해당 document d와 input context x 사이의 유사도(cosine similarity)를 기반으로 한 가중치이다. 이 가중치는 아래와 같은 과정을 통해 산출한다.

$$\lambda(d, x) = \frac{e^{s(d,x)}}{\sum_{d \in \mathcal{D}'} e^{s(d,x)}}$$

해당 가중치는 retrieve된 k개의 document들에 대해, softmax를 취해준 값이다.

즉, 각 document별 산출된 output probabilities에 대해 해당 document가 input과 얼마나 유사한지에 기반한 가중치를 곱한 뒤, document에 대해 marginalize를 함으로써 최종적인 output probability를 구하게 되는 것이다

지금까지 REPLUG에 대해 살펴보았다. 그렇다면, 이러한 REPLUG는 학습이 어떻게 진행될까? 이를 알아보기 위해, 논문에서 제시한 요소였던 REPLUG LSR에 대해 살펴보도록 하겠다.

REPLUG LSR: Training the Dense Retriever

REPLUG LSR의 개념에 대해 간략하게 말하자면, LM을 score function 개념으로 두게 되면서 retriever의 supervision을 제공하고, 이를 토대로 retriever를 학습시킨다는 아이디어이다.

이 학습 과정은 결국, retriever가 LM의 PPL(perplexity)을 낮추게 하는 document를 retrieve 하게끔, 즉 document retrieve를 보다 잘하게끔(혹은 양질의 document를 가져오게끔) 학습된다는 의미를 가진다.

이 과정은 크게 4가지의 step으로 나눌 수 있는데, 이는 아래와 같다.

1. Document retrieve 이후 retrieval likelihood 산출 (Computing Retrieval Likelihood)
2. Language model을 통해 retrieved document의 score 산출 (Computing LM Likelihood)
3. Retriever model의 parameter를 KL-divergence 목적함수를 통해 update (Loss Function)
4. External knowledge datastore의 index를 비동기적으로 update (Asynchronous Update of the Datastore Index)

하나씩 차근차근 살펴보도록 하겠다

Computing Retrieval Likelihood

우리는 retriever model을 이용하여 전체 external knowledge corpus $\mathcal{D}$로부터 top-k document $\mathcal{D}'$를 retrieve 한다. 이 과정에서, retrieval likelihood를 구하게 되는데, 이는 아래와 같은 수식을 통해 산출해 내게 된다.

$$P_R(d \mid x) = \frac{e^{s(d,x)/\gamma}}{\sum_{d \in \mathcal{D}'} e^{s(d,x)/\gamma}}$$

간단하게, 각 document d에 대해 input context x와의 유사도를 기반으로 하여 softmax를 취함으로써 구할 수 있다.

여기서, γ는 softmax의 temperature를 조절하는 hyperparameter이다.

또한 수식을 살펴보면 전체 external knowledge corpus $\mathcal{D}$가 아닌 top-k document $\mathcal{D}'$에 대해서만 softmax를 취하는 것을 확인할 수 있다.

논문에서는 이에 대해 전체 document($\mathcal{D}$)에 대해 likelihood를 구하는 것이 이상적이지만, 이는 **현실적으로 다루기 힘들기에, top-k에 대해서만 likelihood를 구함으로써 전체 document를 "근사"한다고 밝힌다.**

(실제로, 많은 선행 연구들에서도 top-k document에 대해서만 다뤄왔었다. 그 이유로는 top-k 이외의 다른 document는 likelihood 분포에서 차지하는 비율도 적을뿐더러, 이렇게 별로 중요하지 않은 요소들을 계산하는데 요구되는 계산량이 너무 많기 때문이다.)

Computing LM Likelihood

이렇게 retrieval likelihood을 구한 이후에는, retrieve 된 top-k document들에 대해 각각의 document가 얼마나 LM의 PPL(perplexity)에 긍정적인 영향을 미쳤는지(PPL이 낮으면 LM의 성능이 좋은 것)를 측정하게 된다

이를 위해 먼저, $P_{LM}(y|d, x)$를 구하게 된다. 이는 곧 document d와 input context x가 주어졌을 때, LM이 y를 산출할 확률이다. 이때 y는 ground truth output(causal LM이라면 다음 timestep의 token)이다.

이 값이 높을수록, given d가 LM의 PPL에 긍정적인 영향을 미쳤다고 볼 수 있다. (왜냐면, x와 LM의 parameter는 변하지 않고 document d만 변하면서 다른 값을 산출하기 때문이다.)

이렇게 $P_{LM}(y|d,x)$를 구한 이후, 이를 이용하여 각 document d에 대한 LM의 likelihood를 구하게 된다. 이는 아래와 같다

$$Q(d \mid x, y) = \frac{e^{P_{LM}(y|d,x)/\beta}}{\sum_{d \in D'} e^{P_{LM}(y|d,x)/\beta}}$$

이 또한 각 document별 $P_{LM}(y|d,x)$에 대해 softmax를 취한 값이다. 또한, 여기서의 β 또한 위에서의 γ와 같은 역할을 하는 hyperparameter이다

Loss Function

우리는 지금까지 document에 대한 retriever의 likelihood와 LM의 likelihood를 구하였다.

이 두 가지의 likelihood를 통해 retriever model의 parameter update를 하게 되는데, 이에 대한 목적 함수, 즉 loss function은 아래와 같다.

$$\mathcal{L} = \frac{1}{|\mathcal{B}|} \sum_{x \in \mathcal{B}} KL\Big(P_R(d \mid x) \parallel Q_{\mathrm{LM}}(d \mid x, y)\Big),$$

두 likelihood에 대한 KL divergence가 REPLUG LSR의 loss function이다. 즉, REPLUG LSR은 이 KL divergence를 최소화하는 방향으로 학습이 진행되게 된다.

KL divergence는, 두 확률 분포 사이의 차이를 측정하는 metric이다. 즉, 두 확률 분포가 비슷할수록 KL divergence는 작아지고, 다를수록 커지는 특징을 가진다. 이에 대한 수식은 아래와 같다.

$$KL(p||q) = -\mathbb{E}_{x \sim p(x)} \left[\log \frac{q(x)}{p(x)}\right]$$

$$= -\int p(x) \log \frac{q(x)}{p(x)} dx$$

KL divergence에 대한 보다 자세한 설명은 과거에 정리해 놓은 적이 있으니, 해당 링크를 첨부하고 여기서는 더 다루지 않겠다.

KL-Divergence와 Entropy, Cross-Entropy란?

결론적으로, LM의 likelihood와 retriever의 likelihood 사이의 KL divergence를 최소화한다는 것은, 두 확률 분포를 최대한 비슷하게 맞춰간다는 의미를 가진다.

그런데, REPLUG에서의 전제는 LM이 black-box setting, parameter update가 일어나지 않는다는 상황이기 때문에, 결국 retriever의 확률 분포를 LM의 분포처럼 update해나가겠다는 의미를 가지는 것이다.

즉, retriever model에서 d,x 가 고정되었을 때, 해당 값들이 어떤 확률 분포에서 왔을지에 대한 "가능성"인 likelihood는 LM에서 x 와 y, d 가 고정되었을 때, 해당 값들이 어떤 확률 분포에서 왔을지에 대한 likelihood와 비슷해지는 것을 목적으로 training된다.

이는 즉, retriever가 LM의 PPL(perplexity)을 낮추게 하는 document를 retrieve 하게끔, 즉 document retrieve를 보다 잘하게끔(혹은 양질의 document를 가져오게끔) 학습된다는 의미를 가진다.

Asynchronous Update of the Datastore Index

해당 부분은 REALM에서 제안된 바 있는, k**nowledge corpus embedding과 그에 대한 index를 계산하는 과정과 training 과정을 비동기적으로, 병렬적으로 수행**하는 방법론이다. 이에 대한 자세한 설명은 아래 REALM 리뷰에서 더 자세히 확인해 볼 수 있다.

[논문 리뷰] REALM: Retrieval-Augmented Language Model Pre-Training

지금까지 REPLUG LSR에 대한 전반적인 과정을 살펴보았다. 아래의 figure는 위 과정을 나타낸 figure이다.

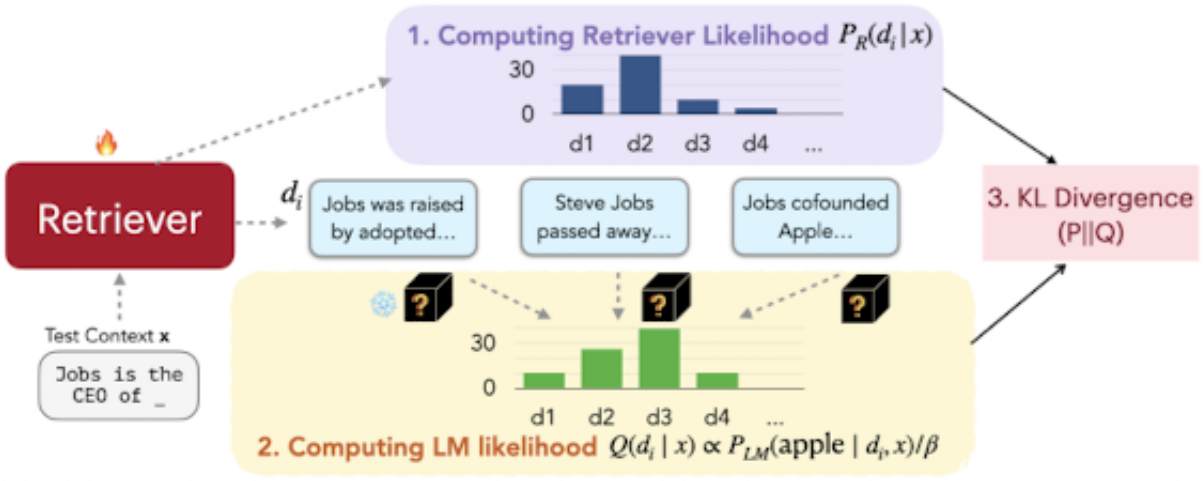


Figure 3. **REPLUG LSR training process (§4).** The retriever is trained using the output of a frozen language model as supervision signals.

Training Step

이제, 논문에서 진행된 실험들에 대해 살펴보도록 하겠다. 이번 파트는 실험이 진행된 환경에 대해 간략하게 소개한 파트이다.

REPLUG

이론적으로는 REPLUG에 sparse retriever(BM25, TF-IDF 기반 retriever)나 dense retriever 모두 사용 가능하지만, 논문에서는 실험을 진행할 때 Contriever의 구조를 따랐다고 한다.

REPLUG LSR

또한, Contriever의 parameter를 REPLUG의 retriever의 초기 parameter값으로 두었으며, LM likelihood 측정을 위한 LM으로는 GPT-3 Curie를 사용했다고 밝힌다.

Training data는 Pile training data로부터 sampling 한, 256 token으로 구성된 sequence 800K 개를 training query로 두었으며, 각 query는 첫 128 token 부분은 input context x로, 나머지 128 token에 대해서는 ground truth continuation y로 두었다 (GPT-3는 causal LM이기에 continuation이 ground truth이다)

외부 knowledge를 담당하는 external corpus $\mathcal{D}$는, 마찬가지로 Pile training data로부터 128 token으로 구성된 36M 개의 document를 sampling 하여 사용하였으며, trivial retrieval 문제를 피하기 위해, query로 sampling 한 sequence와 document로 sampling한 sequence는 중복이 없게끔 처리하였다.

(Trivial retrieval 문제란, pre-training corpus와 knowledge corpus가 같은 경우이다.

만약 query x 가 document d 에 있는 문장과 같은 문장이면, model은 x 와 d 의 관계를 학습하는 것이 아닌, 순전히 문자열이 matching 되는지 확인하는 방향으로 학습될 가능성이 있다. 이 또한 위의 REALM 리뷰에서 다룬 바 있으니 참고하면 좋을 것 같다.)

추가적으로, $k = 20$의 top-k document를 retrieve 하여 training 하였다고 한다.

Experiments

논문에서는 language modeling과 MMLU, open-domain QA와 같은 downstream task 모두에서 성능 측정을 진행하였다.

먼저, language modeling 결과이다.

Model		# Parameters	Original	+ REPLUG	Gain %	+ REPLUG LSR	Gain %
GPT-2	Small	117M	1.33	1.26	5.3	1.21	9.0
	Medium	345M	1.20	1.14	5.0	1.11	7.5
	Large	774M	1.19	1.15	3.4	1.09	8.4
	XL	1.5B	1.16	1.09	6.0	1.07	7.8
GPT-3 (black-box)	Ada	350M	1.05	0.98	6.7	0.96	8.6
	Babbage	1.3B	0.95	0.90	5.3	0.88	7.4
	Curie	6.7B	0.88	0.85	3.4	0.82	6.8
	Davinci	175B	0.80	0.77	3.8	0.75	6.3

Table 1. **Both REPLUG and REPLUG LSR consistently enhanced the performance of different language models.** Bits per byte (BPB) of the Pile using GPT-3 and GPT-2 family models (Original) and their retrieval-augmented versions (+REPLUG and +REPLUG LSR. The gain % shows the relative improvement of our models compared to the original language model.

모든 setting에서, original < REPLUG < REPLUG LSR 순으로 높은 성능을 보인 것을 확인할 수 있다.

다음으로는 MMLU에서의 결과이다. 이때, 모든 model은 5-shot in-context learning으로 실험이 진행되었다

Model	# Parameters	Humanities	Social.	STEM	Other	All
Codex	175B	74.2	76.9	57.8	70.1	68.3
PaLM	540B	77.0	81.0	55.6	69.6	69.3
Flan-PaLM	540B	-	-	-	-	72.2
Atlas	11B	46.1	54.6	38.8	52.8	47.9
Codex + REPLUG	175B	76.0	79.7	58.8	72.1	71.4
Codex + REPLUG LSR	175B	76.5	79.9	58.9	73.2	71.8

Table 2. **REPLUG and REPLUG LSR improves Codex by 4.5% and 5.1% respectively.** Performance on MMLU broken down into 4 categories. The last column averages the performance over these categories. All models are evaluated based on 5-shot in-context learning with direct prompting.

우선, Codex model부터 살펴보자. 이 또한 original < REPLUG < REPLUG LSR 순으로 높은 성능을 보인 것을 확인할 수 있다.

이어서, parameter 수에서 3배 정도의 차이를 보이는 PALM model과도 비슷한 성능을 보인 것을 확인할 수 있다.

이어서, Open-domain QA task에서의 결과이다.

	NQ		TQA	
Model	Few-shot	Full	Few-shot	Full
Chinchilla	35.5	-	64.6	-
PaLM	39.6	-	-	-
Codex	40.6	-	73.6	-
RETRO†	-	45.5	-	-
R2-D2†	-	55.9	-	69.9
Atlas†	42.4	**60.4**	74.5	**79.8**
Codex + Contriever$_{cc}$[2]	44.2	-	76.0	-
Codex + REPLUG	44.7	-	76.8	-
Codex + REPLUG LSR	**45.5**	-	**77.3**	-

Table 3. Performance on NQ and TQA. We report results for both few-shot (64 shots for Chinchilla, PaLM, and Atlas; 16 shots for Codex-based models) and full training data settings. REPLUG LSR improves Codex by 12.0% on NQ and 5.0% on TQA, making it the best-performing model in the few-shot setting. Note that models with † are finetuned using training examples, while other models use in-context learning.

Few-shot 환경에서 REPLUG와 REPLUG LSR이 가장 좋은 성능을 내는 것을 확인할 수 있다.