REPLUG: Retrieval-Augmented Black-Box Language Models

- **Weijia Shi,1 * Sewon Min,1 Michihiro Yasunaga,2 Minjoon Seo,3 Rich James, 4 Mike Lewis,4 Luke Zettlemoyer1 4 Wen-tau Yih 4**

- **NAACL 2024**

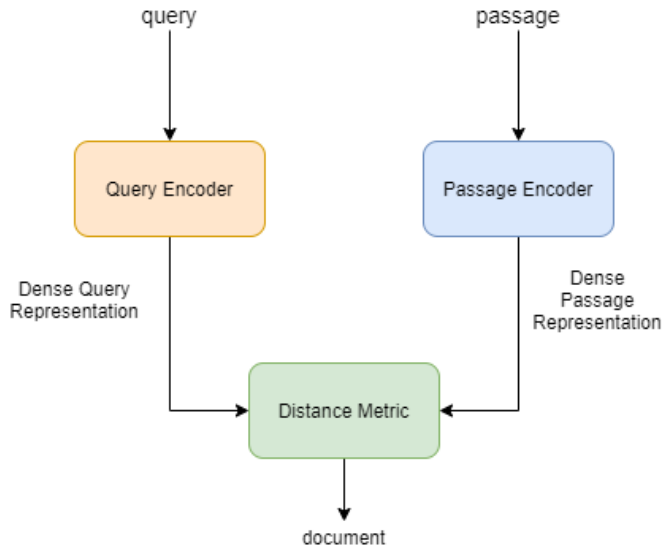
발표자: 안도형

25. 07. 03.

Background & Related Work

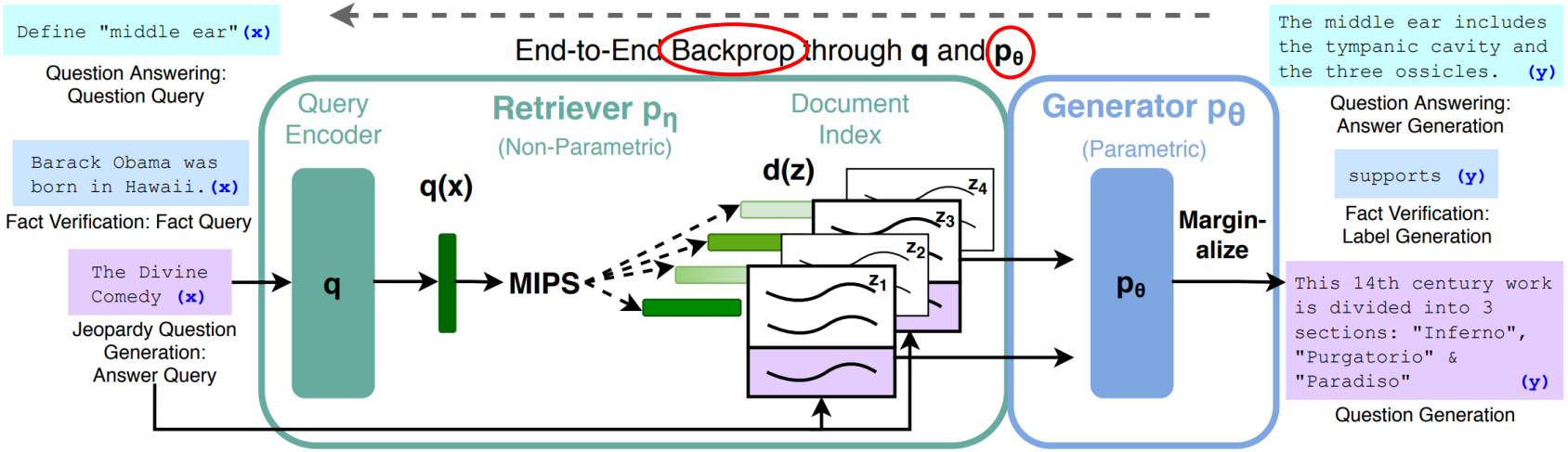
Background & Related Work

- **Black-box language Models**
  - GPT-3와 같은 LLM 모델들은 API로만 풀려 있고, 모델 내부 parameter 같은 요소에는 접근 할 수 없음
  - 논문에서는 이렇게 모델 내부 파라미터에 접근 할 수 없는 모델을 black-box language Model 이라고 말함
- **Dual Encoder**
  - 두 개의 인코더를 사용하여 쿼리와 문서를 각각 독립적으로 임베딩 벡터로 변환 한 후 각 벡터 간의 유사성을 계산하여 검색하기 위한 구조
  - Single Encoder 방식을 사용하면 쿼리와 문서를 함께 인코딩 해야 해서 새로운 쿼리가 들어올 때마다 문서를 다시 인코딩 해야 해서 비효율적



Background & Related Work

- **Retrieval augmented Language Models(RALM)**
  - Retrieval이 외부 코퍼스에서 문서(지식)들을 검색하고 LM은 검색된 문서(지식)들을 활용하여 최종 output을 만들어냄
  - RALM 방식은 retriever 뿐만 아니라 LM의 parameter도 업데이트 해야 해서 black-box LM에 적용할 수 없음



Introduction

Introduction

- **기존 LLM들의 한계**
  - GPT-3, Codex와 같은 LLM들은 hallucination이 발생하는 문제가 있음
  - Hallucination 문제를 해결하기 위해 Retrieval-augmented language models을 사용
  - RALM은 필요한 정보를 외부 데이터셋에서 검색해서 가져와 hallucination을 줄이고 LM이 다루는 정보의 범위를 넓힘

- **기존 Retrieval Augmented Language Models의 한계**
  - RALM은 필요한 정보를 외부 데이터셋에서 검색해 가져와 hallucination을 줄이는 결과를 가져옴
  - 그러나 기존 RALM은 LM의 내부 파라미터에 접근해서 업데이트를 진행해야 함
  - 최근 GPT-3 같이 API로만 제공되어 내부 파라미터에 접근이 불가능한 LM에는 RALM을 적용하기 힘들다는 한계가 존재

>> 논문의 연구에서는 retrieval을 통해 black-box LLM을 개선하는 방법을 찾으려 함

Introduction

- **REPLUG**(Retrieve and Plug)
  - LM을 black-box로 두고 Retrieval을 활용하여 black-box language model의 성능을 향상 시키는 retrieval-augmented LM **framework**
  - 기존 RALM은 LM 내부 파라미터를 업데이트
  - LM을 black-box로 두기 때문에 **REPLUG**는 이러한 과정이 필요하지 않음

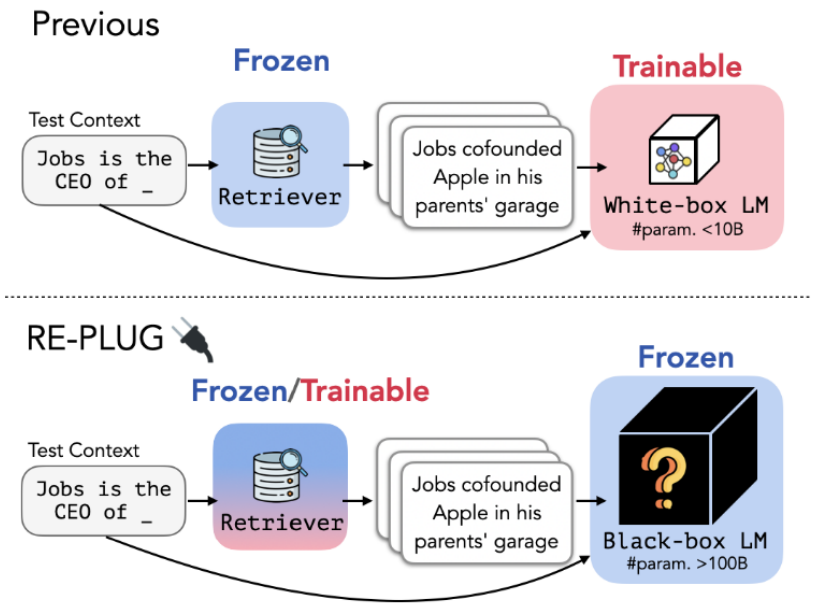


Figure 1. Different from previous retrieval-augmented approaches (Borgeaud et al., 2022) that enhance a language model with retrieval by updating the LM's parameters, REPLUG treats the language model as a black box and augments it with a frozen or tunable retriever. This black-box assumption makes REPLUG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

Methods

REPLUG

Method : REPLUG

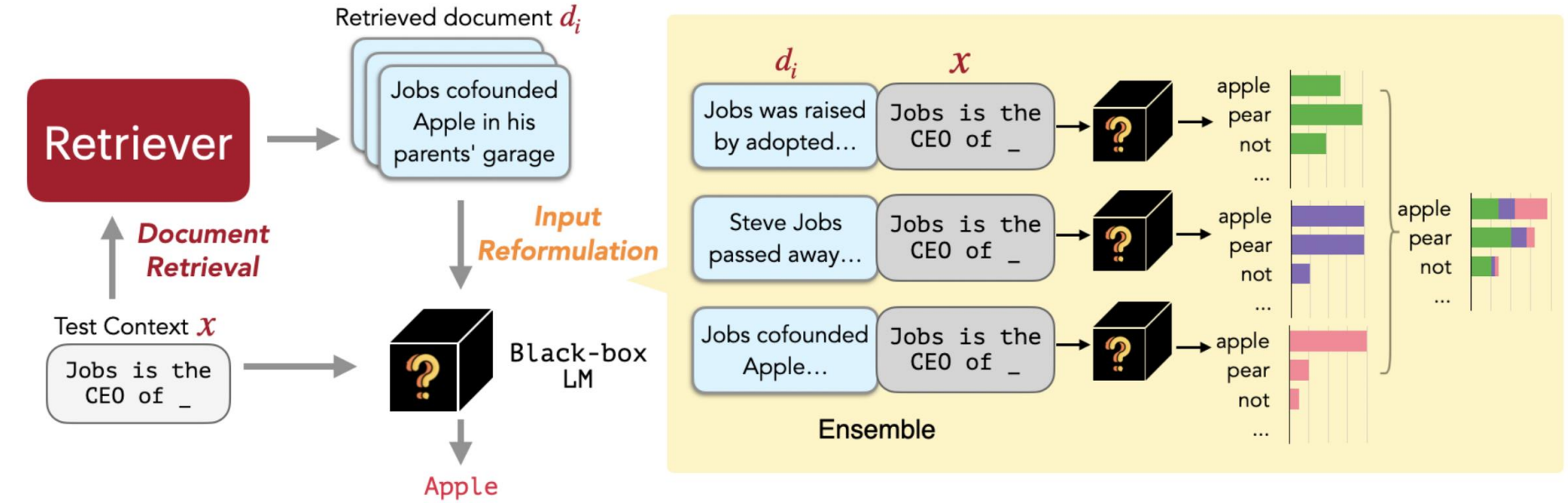


Figure 2. **REPLUG at inference** (§3). Given an input context, REPLUG first retrieves a small set of relevant documents from an external corpus using a retriever (§3.1 *Document Retrieval*). Then it prepends each document separately to the input context and ensembles output probabilities from different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

- REPLUG
  - Input context가 주어지면 REPLUG는 먼저 외부 코퍼스에서 관련 문서들을 검색
  - 각 문서들을 input context 앞에 붙이고 병렬로 입력 후 나온 각각의 출력 확률들을 앙상블

Method : REPLUG

D : external corpus $\{d1, \ldots dm\}$
d : document
x : input context
$E(d)$: 문서에 대한 embedding vector
$E(x)$: input에 대한 embedding vector

- **Document Retrieval**
  - input context가 주어졌을 때 외부 코퍼스에서 문서를 검색하는 과정

1. **Input context가 주어지면 retriever는 D중에서 x와 관련 있는 documents을 가져옴**
2. **dual encoder에 기반한 dense retriever을 사용하게 되고, x와 문서 d를 각각 encode**
   - Dual encoder는 입력과 문서를 따로 encode함
   - Encoder가 문서 d에 대해 encode를 진행하고, 문서 d의 각 token 별 hidden representation을 mean pooling하여 embedding vector $\boldsymbol{E}(\boldsymbol{d})$를 만듬
   - 입력 x에 대해서도 동일한 방법으로 $\boldsymbol{E}(\boldsymbol{x})$를 만듬
   - $E(d)$와 $E(x)$ 사이의 유사성을 구하여 top-k개의 문서가 선택되고, 이때 유사성은 코사인 유사도를 사용

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

   - 또한 효율적인 retrieval을 위해, 모델은 각 문서 별 embedding $E(d)$를 미리 계산해놓고, 해당 embedding들에 대한 FAISS index를 구축 해놓음
   - 논문에서는 훈련 시 쿼리 x가 주어지면 FAISS 인덱스로 부터 top-20개의 문서를 검색

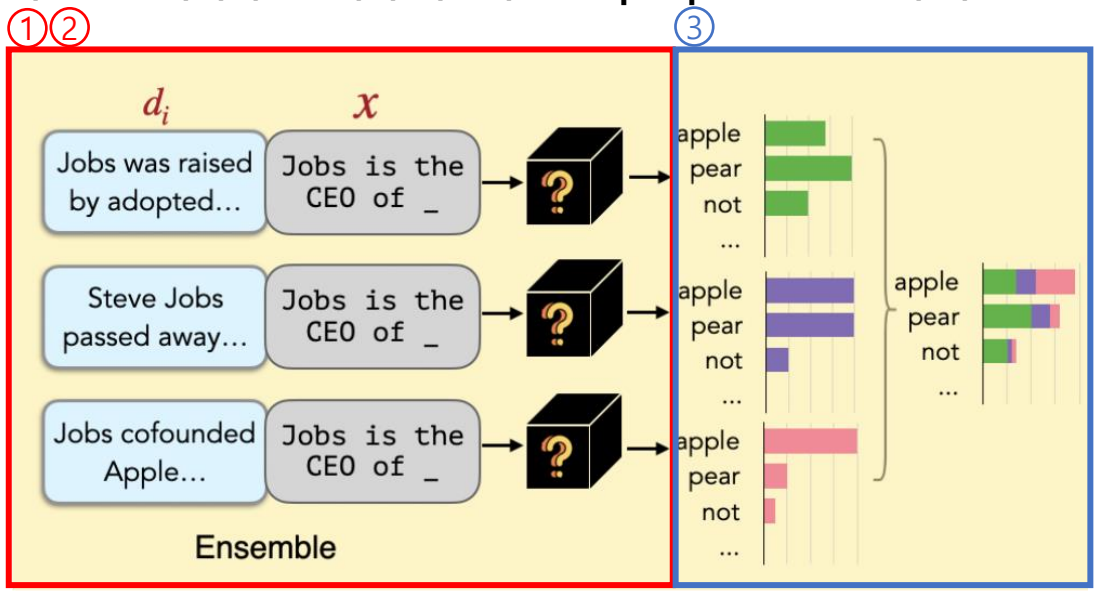
Method : REPLUG

- **Input Reformulation**
  - **input context에 top-k개의 문서들을 병렬로 붙여서 LM의 입력으로 주고 앙상블 기법을 사용**

Why?

- top-k개의 문서들은 LM이 더 좋은 output을 나오게끔 함
- 모든 top-k개의 문서들을 input context x 앞에 붙여서 LM의 input으로 제공해주기에는 LM의 context size 제한으로 실행 불가능

1. **Top-k개 만큼의 문서가 retrieve 됨**
2. **각각 문서에 대해 개별적으로 input context x 앞에 붙임**
3. **input representation을 LM에 개별적으로 통과시키면 3개의 개별적인 output probabilities가 나옴**



Method : REPLUG

D' : retrieve된 k개의 문서들의 집합
x : input context
d : document
$d \circ x$: x 앞에 d(문서)를 추가

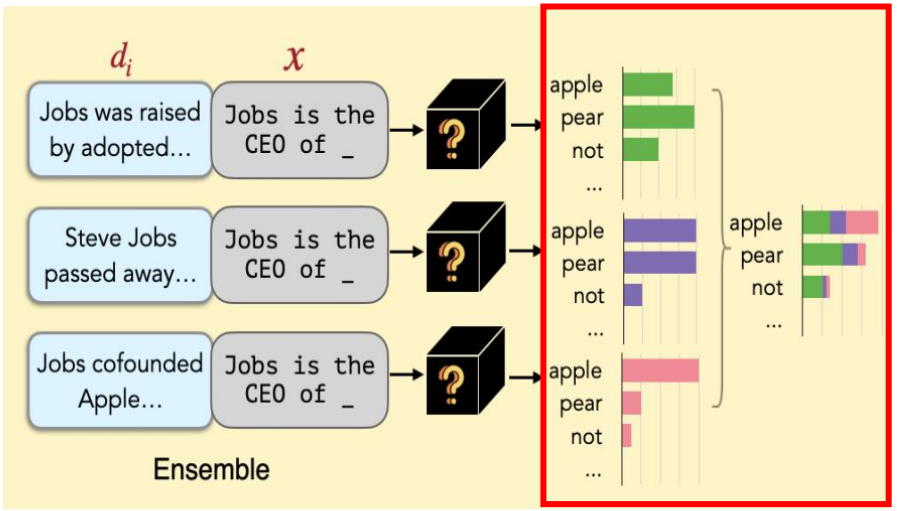
4. D'에 대한 output probabilities는 다음과 같이 구함

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

- D'는 retrieve 된 k개의 문서들의 집합, $d \circ x$ 는 x 앞에 문서 d를 추가함을 의미
- $\lambda(d,x)$는 문서 d와 input context x 사이의 코사인 유사도를 기반으로 한 가중치

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

- 해당 가중치는 retrieve된 k개의 문서들에 대해, $softmax$를 취해준 값



>> 각 문서 별 산출된 output probabilities을 해당 문서와 input의 유사도를 기반으로 한 가중치와 곱한 뒤,

각 문서별 probabilities들을 앙상블 하여 최종 output probabilities를 구하게 되는 것

Experiments

Language Modeling

MMLU

Open Domain QA

Experiments

Language Modeling

- **Datasets**
  - Pile : 웹페이지, 코드, 학술 논문 등의 다양한 도메인으로 구성된 text source를 포함한 데이터셋
- **Result**
  - BPB는 텍스트의 각 바이트를 예측하는데 필요한 평균 비트 수를 나타내는 지표
  - BPB가 낮을수록 모델이 텍스트를 예측하기 위해 필요한 정보를 더 효율적으로 압축하거나 인코딩한다는 것을 의미하며, 이는 더 높은 정확도와 더 나은 모델 성능을 의미

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.

Experiments

MMLU(Massive Multi-task Language Understanding)

- **Datasets**
  - MMLU는 57개 task들의 시험 문제(수학,컴퓨터과학,법,역사 등등)를 다루는 다중 QA 데이터셋
- **Results**
  - Codex < Codex + REPLUG < Codex + REPLUG LSR 순으로 성능이 향상 되는 것을 확인할 수 있음
  - parameter 수에서 3배 정도 차이를 보이는 PaLM 모델과도 비슷한 성능을 보인 것을 확인 할 수 있음
  - Atlas에서 낮은 성능이 나옴으로써 black box를 사용하는 REPLUG의 효과가 있음을 보여주고 있음

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.

Experiments

Open Domain QA

- **Datasets**
  - NQ와 TriviaQA는 Wikipedia와 웹으로 부터 수집된 질문,답변들을 포함하는 Open-domain QA 데이터셋
- **Results**
  - REPLUG LSR은 NQ에서 Codex의 성능을 40.6 -> 45.5, TQA에서 73.6 -> 77.3 향상시킴
  - 64개 훈련 예제로 fine tuning된 이전의 SOTA 모델인 Atlas를 능가하여 few-shot 세팅에서 새로운 SOTA를 달성
  - 여전히 전체 훈련 데이터로 fine-tuning된 retrieval augmented language model의 성능에는 못 미침
  - 훈련 dataset에 중복된 테스트 질문이 다수 존재하기 때문일 가능성이 높다고 예측함

	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.

Conclusion

Conclusion

- **요약 및 의의**
  - 논문의 저자는 LM을 black box 취급하고 조정 가능한 retrieval model을 추가하는 retrieval-augmented language modeling framework인 REPLUG와 훈련 방법 REPLUG-LSR을 소개
  - 논문의 평가에 따르면 REPLUG는 기존 LM에서 언어 모델링이나 downstream task의 성능을 향상 시킬 수 있었음
  - GPT-3, Codex 같이 API로만 접근이 가능한 LLM 모델들에 대해 LM에 접근 없이 성능 향상을 불러온 방법이라는 점에서 향후 개발될 다양한 LLM에도 적용이 가능할 수 있다는 것이 의미가 있어 보임

- **한계점 및 향후 연구 방향**
  - REPLUG는 모델이 검색된 지식에 의존하는지, parametric 지식에 의존하는지 명확하지 않음
  - 특정 작업에서 여전히 완전한 성능을 보장하지 못하는 경우가 있음
  - 향후 연구는 REPLUG의 적용 범위를 확장 시키고 검색된 정보와 모델의 내부지식을 구분할 수 있는 방법을 개발하는 방향이 될 예정
  - REPLUG(LSR)을 추가하는 기준이 되는 모델이 Codex인데 이는 코드를 fine-tuning한 모델이라서 GPT-3 기반이긴 하지만 실험에서 사용한 Task의 카테고리가 조금 다른게 아쉬운 부분으로 느껴짐

PPL(Perplexity)

- **PPL**
  - PPL은 텍스트 생성 언어 모델의 성능 평가지표 중 하나로 모델이 주어진 텍스트를 얼마나 잘 예측하는지 나타내는 지표
  - PPL은 언어 모델의 성능 평가 시 이전 단어로 다음 단어를 예측할 때 몇 개의 단어 후보를 고려하는지를 의미
    - 여기서 고려해야 할 단어 후보가 많다는 것은 그 만큼 언어 모델이 쉽게 정답을 못 내고 있다고 해석할 수 있음
    - **즉 PPL 값이 낮을수록 언어 모델이 쉽게 정답을 찾아내는 것이므로 성능이 우수하다고 평가**

$$Perplexity = \sqrt[N]{\frac{1}{P(w_1, w_2, w_3, \ldots, w_N)}} = \sqrt[N]{\frac{1}{\prod_{i=1}^{N} P(w_i|w_1, w_2, \ldots, w_{i-1})}}$$

- **단어 개수가 N인 문장 W의 수식**
- $P(wi|w1, w2, \ldots, wi-1)$는 **각 단어 wi가 주어진 이전 단어들 $w1, w2, \ldots, wi-1$ 에 기반하여 나타날 확률**
- $Product$ **값이 작으면 각 단어가 나올 조건부 확률이 낮다는 것을 나타내고, 따라서 선택의 폭이 넓어짐 = 모델의 불확실성이 높음 = PPL 값이 높음**
- **이전 단어들을 기반으로 다음 단어를 예측할 때 마다 평균적으로 N개의 단어 후보 중 정답을 찾는다고 해석 가능**

KL-Divergence

- **KL-Divergence**
- KL-Divergence은 **두 확률 분포 간의 차이를 측정하는 지표**

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

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

  - REPLUG에서는 KL-Divergence를 사용하여 검색 모델을 언어 모델의 필요에 맞추어 조정
    - LM은 매개변수를 업데이트 할 수 없는 Black-box 상태이기 때문
    - 그래서 수식 상에서 **P**(Retrieval Likelihood)와 **Q**(LM Retriever)의 위치가 바뀔 수 없음
  - KL-Divergence를 최소화하여 검색 모델이 언어 모델이 필요로 하는 문서를 더 잘 반환하게 하여 언어모델이 예측을 더 잘하도록 학습
  - $\mathbf{B}$ 는 input context의 집합, 전체 입력에 대한 Loss 값을 표준화 하고 일관된 평가가 되도록 함

BPB (Bits-per-byte)

- **BPB**
  - 언어 모델의 성능을 평가하는 데 사용되는 지표
  - 모델이 텍스트 데이터를 얼마나 효율적으로 표현할 수 있는지를 나타냄
  - BPB 값이 낮을수록 모델이 예측을 더 효율적으로 수행할 수 있음을 의미 = 다음 토큰을 예측하는데 더 적은 비트가 필요하므로 성능이 더 좋음

- **BPB은 시퀀스에서 다음 토큰(일반적으로 바이트)을 예측하는 데 필요한 평균 비트 수를 측정**

$$\text{BPB} = \frac{L}{\log_2(e)}$$