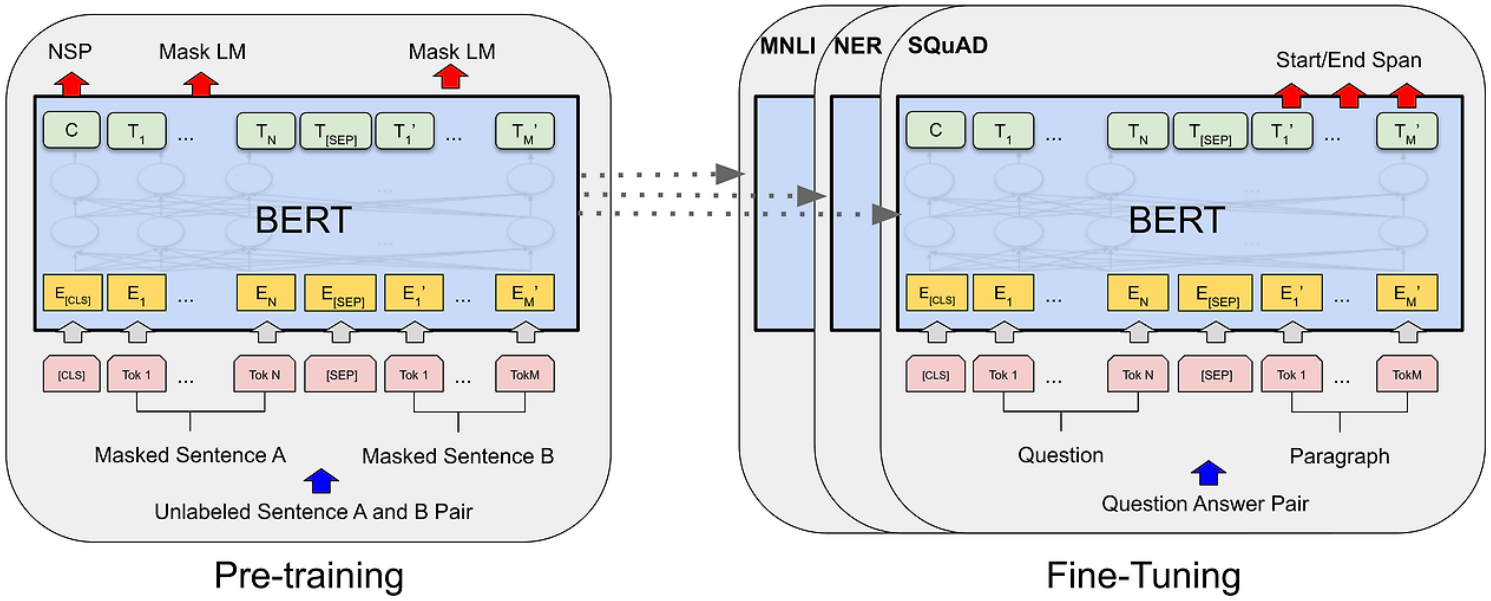
A domain-specific LLM and RAG

jh-cho

Contents

Large Language Models

- Large Language Model(LLM)은 방대한 양의 데이터를 기반으로 **사전 학습된**(pre-trained) 초대형 **딥 러닝 모델**
- 기반이 되는 **트랜스포머(transformer)**는 셀프 어텐션(self-attention) 기능을 갖춘 인코더와 디코더로 구성된 신경망 집합
- 인코더와 디코더는 **텍스트 시퀀스**에서 의미를 추출하고 텍스트 내의 단어와 구문 간의 관계를 이해함

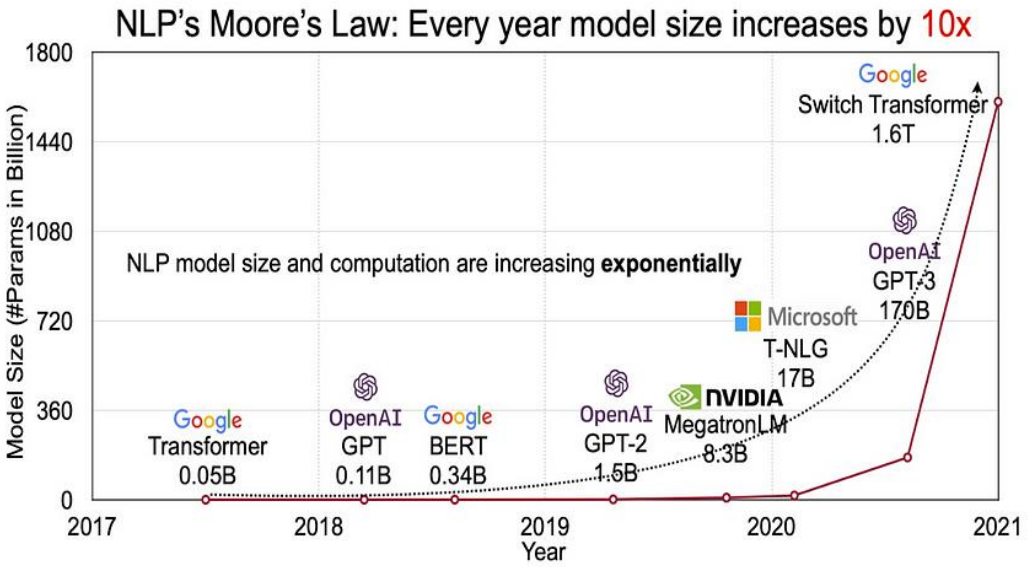


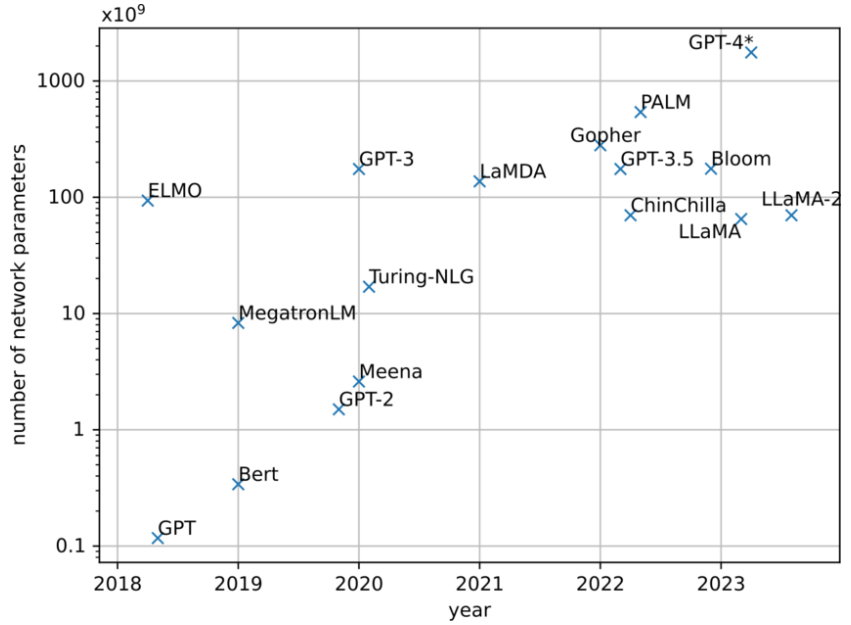
BERT 구조

Large Language Models

- Large Language Model(LLM)의 task
  - Text generation, machine translation, summarization, sentiment analysis, QA, text classification 등
- 언어모형 연대기 (~2024)

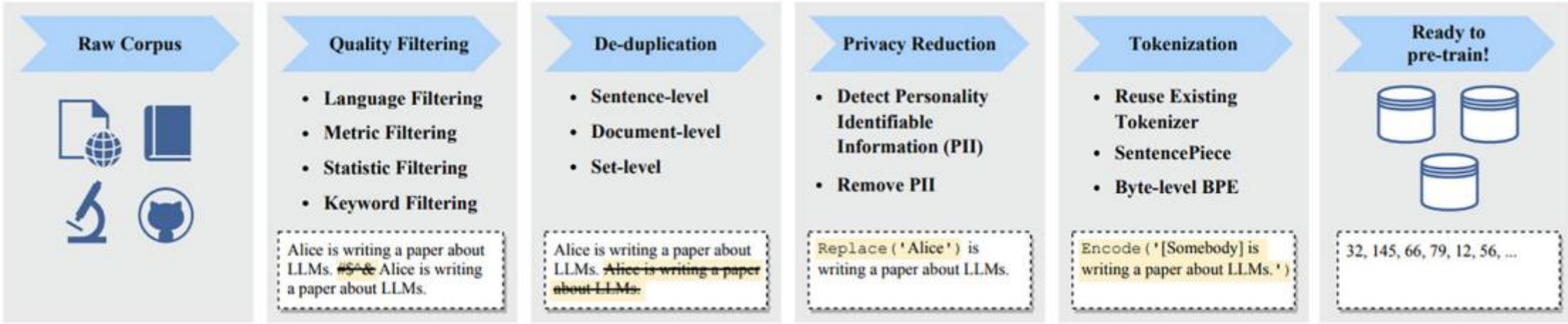
  RNN, LSTM (딥러닝 도입, LM) → Transformer → GPT, BERT (transformer 기반 언어모델)→ GPT2, MegatronLM, T-NLG → GPT3, LLaMA, LaMDA … (모델 크기 확대, LLM) → GPT-4, LLaMA3 … (학습데이터 확대)





Large Language Models

- **파운데이션 모델(Foundation model)** 및 LLM 사전학습(Pre-training)
  - **일반 Text 데이터**(wikipedia 등)를 수집하여 자기지도학습(Self-Supervised Learning)이나 반자기지도학습(Semi-Supervised Learning)을 사용하여 **레이블링되지 않은** 상당한 양의 텍스트로 **사전학습된 언어 모델**(Pre-trained Language Model, PLM) (ex. GPT(generative pretrained transformer))
  - 방대한 양의 데이터를 비지도 학습(unsupervised learning)을 통해 모델을 학습시킨 후 배포, 사용자가 원하는 목적에 맞게 다운스트림(downstream) 작업에 대해 파인튜닝(fine-tuning)이나 in-context learning과 같은 과정을 거쳐 완성



A Domain-specific LLM

- 생성하고자 하는 도메인(ex. 금융, 법률, 의료) 선택 → 관련 데이터 수집 → 일반적인 LLM에 특정 도메인 데이터 정보 주입(Adaptive Pre-training) → 특정 도메인에 대한 대화 모델 학습 → 모델 생성
- 도메인 특화 LLM을 만들기 위해, 사전학습부터 학습하는 **"train from scratch"** 방식과 파인튜닝 과정에서만 도메인 데이터를 사용하는 **"finetune"** 방식을 사용할 수 있음
  - **Train from scratch**: 해당 도메인의 데이터를 사용하여 언어 모델을 처음부터 완전히 새롭게 학습

Pretrained LLM	Corpus size(tokens)	Training budget(A100·hours)	Model architecture	Release time
BloomBergGPT	363B Finance tokens + 345B public tokens	1,300,000	50B-BLOOM	May 2023
XuanYuan2.0	366B for pre-training + 13B for finetuning	Not released	176B-BLOOM	May 2023
Fin-T5	80B Finance tokens	Days/weeks	770M-T5	Feb 2023

  - **Fine-tuned LLM:** 이미 일반적인 데이터로 사전 학습된 언어 모델을 가져와 특정 도메인의 데이터로 추가 학습

Model Name	Finetune data size (samples)	Training budget	Model architecture	Release time
FinMA-7B	Raw: 70k, Instruction: 136k	8 A100 40GB GPUs	LLaMA-7B	Jun 2023
FinMA-30B	Raw: 70k, Instruction: 136k	128 A100 40GB GPUs	LLaMA-30B	Jun 2023
Fin-GPT(V1/V2/V3)	50K	< $300 per training	ChatGLM, LLaMA	July 2023
Instruct-FinGPT	10K Instruction	8 A100 40GB GPUs, ~1 hr	LLaMA-7B	Jun 2023
Fin-LLaMA[53]	16.9K Instruction	NA	LLaMA-33B	Jun 2023
Cornucopia(Chinese)[61]	12M instruction	NA	LLaMA-7B	Jun 2023

A Domain-specific LLM

- Train from scratch 방식은 여러 면에서 **비용**이 상당함

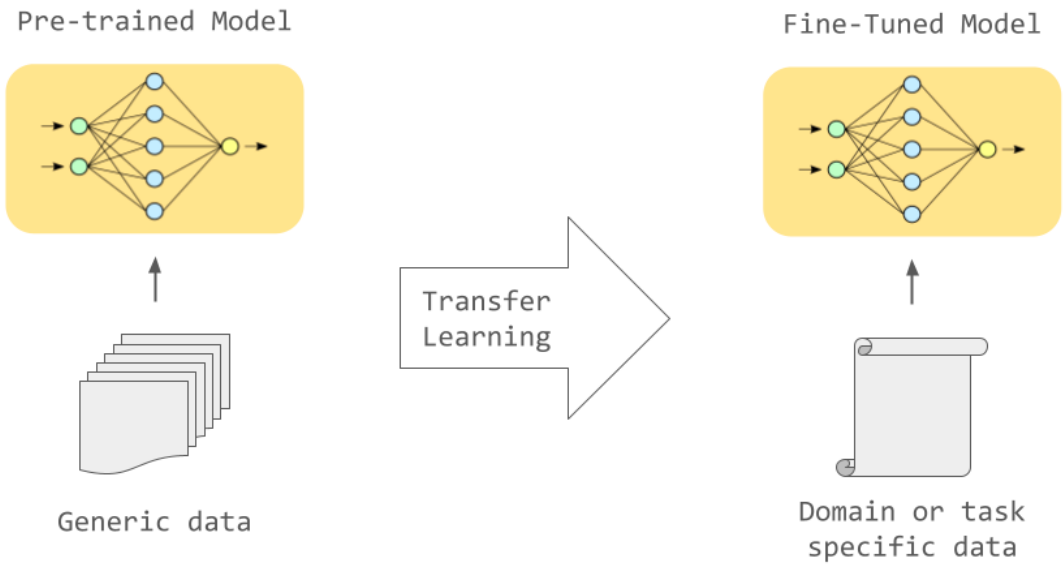
  → **fine-tuning** 혹은 **Tool augmented generation**(ex. RAG) 사용

Options	Development Computational Cost($)	Development Data Cost(samples)	Deployment Computational Cost ($/1k tokens generated)
OpenSource-ZeroShot	-	-	0.006 - 0.037
3rd party-ZeroShot	-	-	0.002 - 0.12
OpenSource-FewShot	-	-	0.006 - 0.037
3rd party-FewShot	-	-	0.002 - 0.12
OpenSource Tool Augmented Generation	Cost of developing tools	-	0.006 - 0.037
3rd party Tool Augmented Generation	Cost of developing tools	-	0.002 - 0.12
OpenSource-Finetune	4-360,000	10,000 - 12,000,000	0.0016 - 0.12
3rd party-Finetune	30-30,000	10,000 - 12,000,000	0.002 - 0.12
Train from Scratch	5,000,000	700,000,000	0.0016 - 0.12

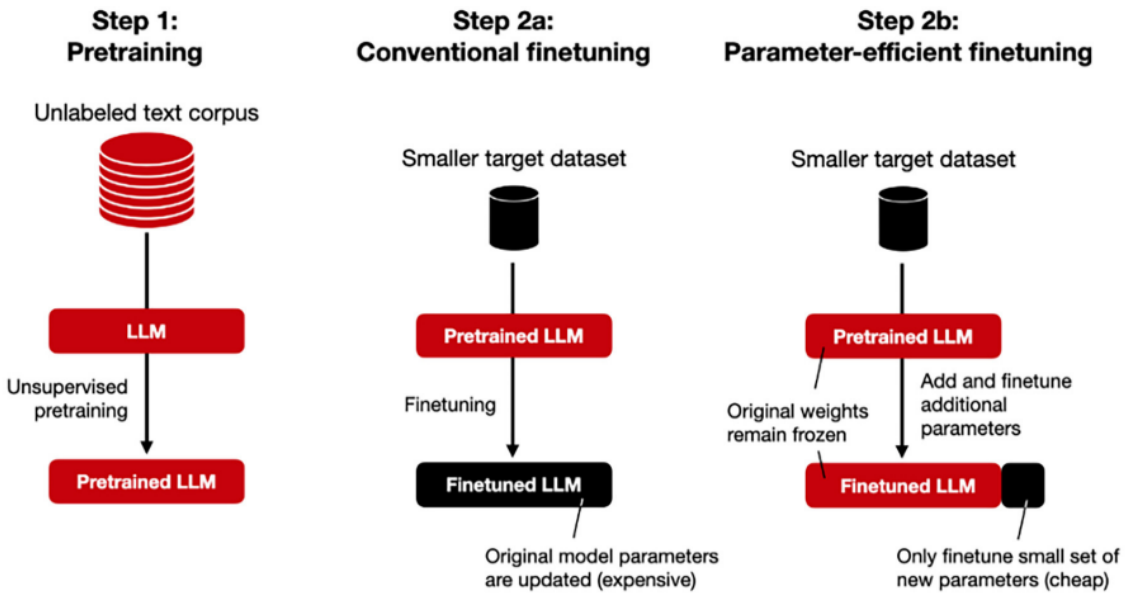
Costs of Different LLM Options

LLM Fine-tuning

- Fine-tuning이란?
  - Pre-Training을 완료한 모델은 모든 학습을 끝낸 것이 아닌, 최종 문제에 맞는 Fine Tuning작업을 해야 함. 따라서 Pre-Training 작업은 반드시 다양한 Task의 Fine Tuning에 적합한 형태여야 함
  - Fine Tuning은 앞선 Pre-Training과 달리 **Supervised Learning** 방법으로 이루어지며, 데이터셋은 **Labeled Dataset**으로 구성.



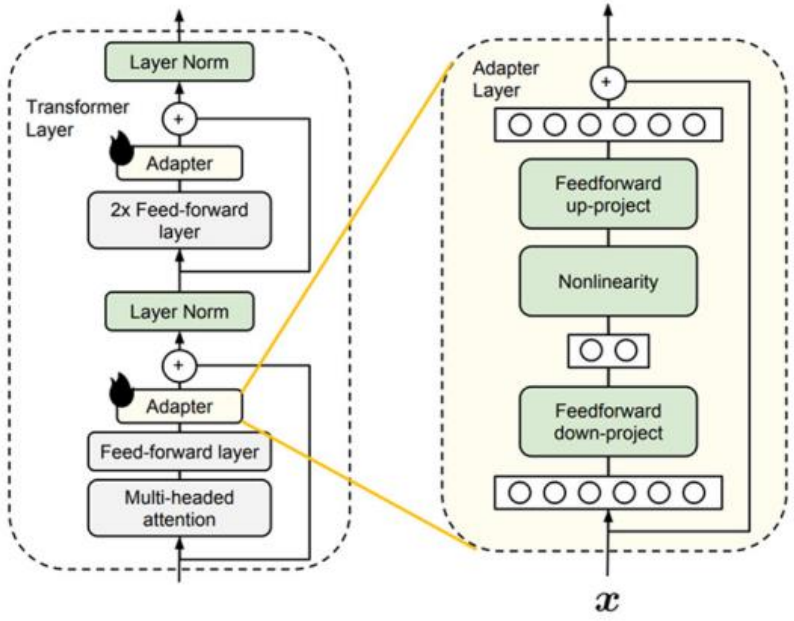
LLM Fine-tuning



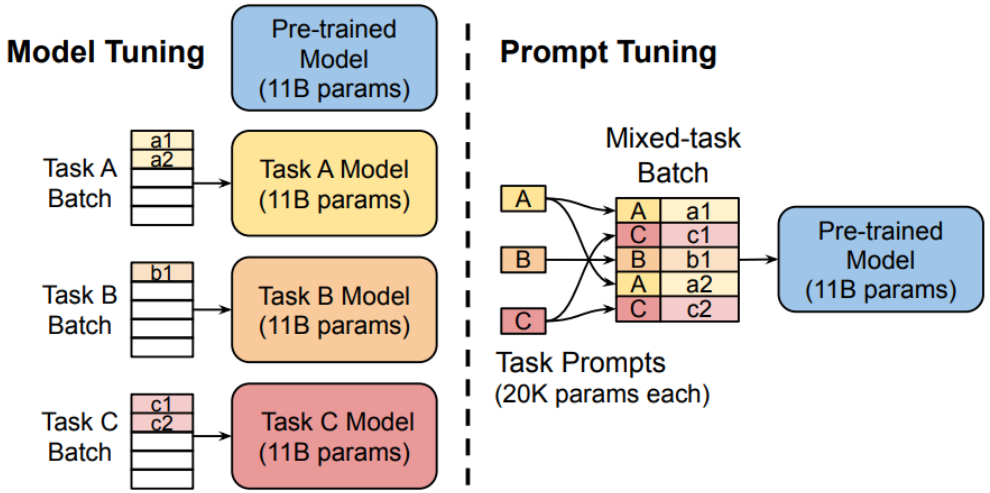
- BERT, RoBERTa 같은 작은 모델을 사용할 때는 Full Fine-tuning 진행 (step 2a)
- 하지만 LLaMA같은 큰 모델이 나오면서 full fine-tuning의 cost가 막대해짐에 따라,
- 최근에는 사전 훈련된 LLM에 **소수의 새로운 파라미터**를 추가하고, 추가된 파라미터만 파인튜닝하는 **PEFT(Parameter Efficient Fine-Tuning)** 방법 주로 사용 (step 2b)

LLM Fine-tuning: PEFT

- **PEFT(parameter-efficient fine-tuning)**
  - **Adapter method**: 사전 학습된 모델의 원래 파라미터는 고정하고, 각 레이어 사이에 작은 모듈(adapter)을 추가하여 이들만 학습
  - **Prompt tuning**: 모델의 입력 부분을 조정하여 특정 작업에 적응시키는 방법
  - **Low-Rank Adaptation(LoRA):** 가중치 행렬을 저차원 행렬로 분해하여 사전 학습된 모델의 원래 가중치는 고정하고, 저차원 행렬들만 학습



Adapter method



Prompt Tuning

LLM Fine-tuning: LoRA

- **"LoRA (Low-Rank Adaptation of Large Language Models)"**

  Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen

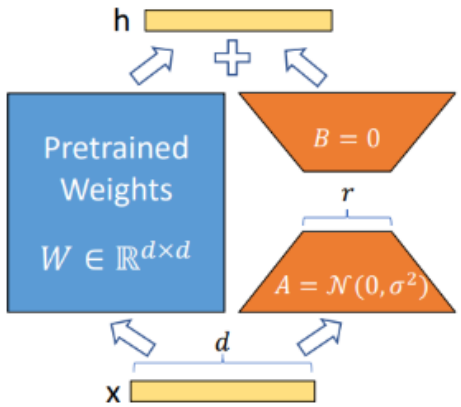
  17 Jun 2021

- 저자들은 모델을 새로운 데이터나 작업에 맞게 조정(adaptation)할 때, **가중치의 변화가 사실상 저차원 공간에서 일어난다고** 생각했다. 즉, 많은 가중치가 변하는 것처럼 보여도 실제로는 몇 가지 주요 방향으로만 변화한다는 것이다.
- LoRA를 사용할 때, 모델의 기존 **사전 학습된 가중치(*W*)는 변경되지 않으며**, 대신에, 저차원 공간에서의 변화를 나타내는 행렬들(rank decomposition matrices)을 학습한다. 이는 원래 고차원 가중치 행렬을 **저차원 행렬 두 개(*A*, *B*)로 분해**하는 것이다.

$$W_0 + \Delta W = W^0 + BA$$

$$\text{where} \quad B \in \mathbb{R}^{d \times r}, \quad A \in \mathbb{R}^{r \times k}, \quad r \ll \min(d, k)$$

$$h = w_0 x + \Delta x = W_0 x + BAx$$



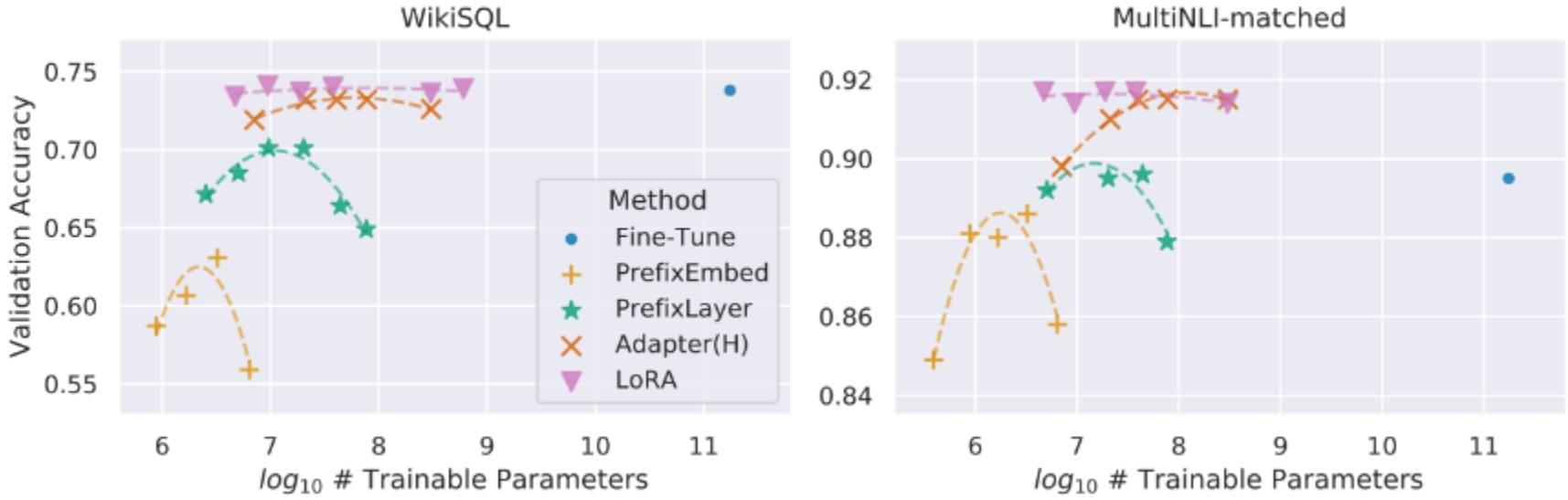
- 원래 파라미터 수: d x k
- LoRA 파라미터 수: d x r + r x k
  (여기서 r은 rank)
- 이 때 d=100, n=100, r=5라고 하면, 10,000 > 1,000

LLM Fine-tuning: LoRA

- **"LoRA (Low-Rank Adaptation of Large Language Models)"**

  Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen

  17 Jun 2021



LLM Fine-tuning: QLoRA

- 최근에는 LoRA를 양자화(quantization)(ex. float32→int8)하여 메모리를 효율적으로 처리하는 **QLoRA** 사용
- Ex. "도메인 특화 LLM: Mistral 7B를 활용한 금융 업무분야 파인튜닝 및 활용 방법", 정천수(2024)

```python
df = data['train'].to_pandas()
df.head(6)
```

1 to 6 of 6 entries Filter

Index	QA_text	Category
0	##Question: 골프보험 알려줘? ##Answer: 골프장에서의 사고를 대상으로, 자동차 보험과는 다르게 골프에 관련된 위험을 포괄적으로 담보하는 보험으로서, 보상하는 손해는 골프의 연습 또는 경기, 지도 중에 타인의 신체에 장해를 입히거나 타인의 재물을 손괴함으로써 생긴 제3자에 대한 손해배상책임, 자기신체손해, 골프용품의 도난 파손 등의 용품손해, 홀인원/알바트로스를 달성한 경우에 관행상 불가피하게 지출해야 하는 축하회 비용 같은 홀인원 비용등 입니다.	장기보험-물건,사물
1	##Question: 고지의무 알려줘 ##Answer: 보험계약의 체결에 있어서 보험계약자 또는 피보험자가 보험자에 대하여 중요한 사실을 고하지 않거나 중요한 사항에 대하여 부실(不實)한 고지를 하여서는 안 된다는 의무입니다.	보험용어
2	##Question: 손해보험과 생명보험의 차이 알려줘 ##Answer: 손해보험은 재산상의 손해를 보험의 목적으로 하며 실손 방식으로 보상함. 생명보험은 사람의 생존과 사망을 보험의 목적으로 하며 정액 방식으로 보상함.제3보험은 상해, 질병, 간병과 관련된 상품을 말하며, 생·손보사에서 모두 판매 가능함.	보험용어
3	##Question: 골프클럽하우스의 가입업종 알려줘? ##Answer: 골프클럽하우스의 가입업종은 골프클럽하우스를 적용합니다. 골프클럽하우스 건물 및 골프장 구내에 있는 모든 시설에 대하여 적용합니다.	장기보험-물건,사물

```python
model_name = "mistralai/Mistral-7B-v0.1"  # Mistral-7B model

bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,              # 4비트 정밀도로 모델 로드
    bnb_4bit_quant_type="nf4",      # pre-trained model은 4비트 NF 형식으로 양자화 되어야 함
    bnb_4bit_use_double_quant=True, # QLoRA 에서 언급한 이중 양자화 사용
    bnb_4bit_compute_dtype=torch.bfloat16, # Pre-trained model은 BF16 형식으로 로드되어야 함
)

model = AutoModelForCausalLM.from_pretrained(
    model_name,
    quantization_config=bnb_config, # bitsandbytes config 사용
    device_map="auto",  # "auto" -> HF Accelerate가 모델의 각 레이어에 배치할 GPU를 결정
    trust_remote_code=True, # Mistral-7B 모델을 사용을 위한 True값 설정
)
```

```python
#레이어에 어댑터 추가
model = prepare_model_for_kbit_training(model)

lora_alpha = 32 # 가중치 matrices를 위한 scaling factor
lora_dropout = 0.05 # LoRA 레이어의 드롭아웃
lora_rank = 32 # Low-Rank matrices dimension

peft_config = LoraConfig(
    lora_alpha=lora_alpha,
    lora_dropout=lora_dropout,
    r=lora_rank,
    bias="none",  # 편향 대신 가중치 매개변수만 훈련하는 경우 'none'값 설정
    task_type="CAUSAL_LM",
    target_modules=["q_proj", "k_proj", "v_proj", "o_proj","gate_proj"]
)
peft_model = get_peft_model(model, peft_config)
```

```python
output_dir = "/content/gdrive/MyDrive/LLM/Mistral-7B-Finetuning-Insurance"
per_device_train_batch_size = 2 # 메모리 부족 오류가 발생하면 배치 크기를 2배로 줄
gradient_accumulation_steps = 2  # 배치 크기가 줄어들면 기울기 누적 단계가 2배 증가
optim = "paged_adamw_32bit" # 더 나은 메모리 관리를 위해 페이징을 활성화
save_strategy="steps" # 학습 중에 채택할 체크포인트 save strategy
save_steps = 10 # 두 개의 체크포인트가 저장되기 전의 업데이트 단계 수
logging_steps = 10  # 두 로그 사이의 업데이트 단계 수
learning_rate = 2e-4  # AdamW 최적화 프로그램의 학습률
max_grad_norm = 0.3 # 최대 그라데이션 표준(gradient clipping)
max_steps = 60        # 60단계 동안 학습
warmup_ratio = 0.03 # 0에서 learning_rate까지 선형 준비에 사용되는 단계 수
lr_scheduler_type = "cosine"  # 학습률 스케줄러
```

LLM Fine-tuning

- 하지만, 도메인 특화 LLM을 만드는데 있어서 파인튜닝(fine-tuning)이 가지는 **문제점**들이 있음
  - 해당 도메인의 **풍부한 양질의 학습 데이터**가 필요
  - 특정 작업에 최적화된 모델은 다른 작업에 대한 **일반화 능력이 떨어질** 수 있음, over-fitting
  - **큰 메모리**와 계산 능력 필요(단, PEFT는 비교적 낮음)
  - 새로운 데이터, 변경된 정보에 대해 **모델 업데이트**, **유지 보수가 어려움**

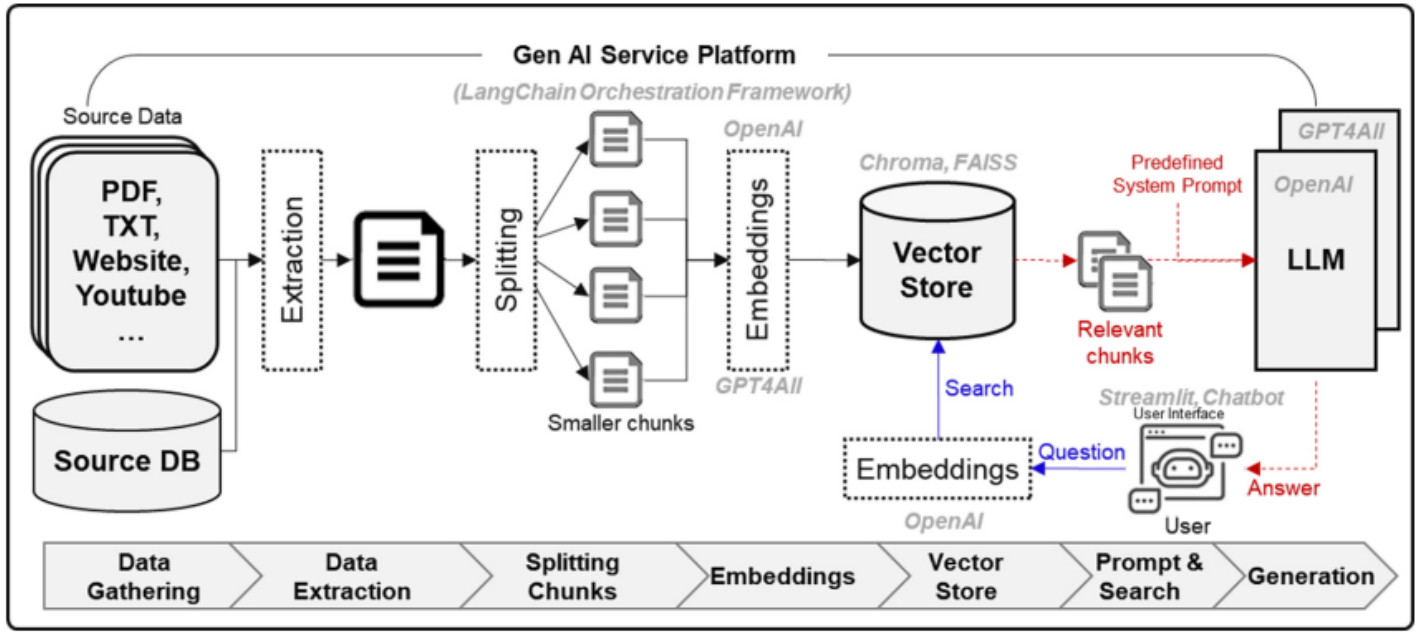
대안 →

1. **프롬프트 엔지니어링**
   - one-shot, few-shot과 같이 프롬프트 질문 시 답변 예시를 제공하여 더 나은 답변 유도
   - 그러나 모든 정보를 프롬프트에 넣어주는 것은 현실적으로 어려움
2. **RAG(검색 증강 생성)**
   - 정보를 데이터베이스에 저장하고, 필요한 정보를 검색하여 LLM에 전달하는 방식으로 구현

방법	컴퓨팅 비용	필요 GPU 개수	예시
Train from Scratch	매우 높음	수백 개 ~ 수천 개	GPT-3 학습, 수천 개의 GPU, 수 주간 학습
Fully Fine-Tuning	높음	수십 개 ~ 수백 개	BERT 모델 fine-tuning, 수십 개의 GPU, 수 일 ~ 수 주간 학습
PEFT (Parameter-Efficient Fine-Tuning)	중간 ~ 낮음	몇 개	LoRA, Adapter, 몇 개의 GPU, 수 시간 ~ 수 일간 학습
RAG (Retrieval-Augmented Generation)	중간 ~ 높음 (초기 설정), 이후 낮음	초기 설정 시 수십 개, 이후 몇 개	검색 시스템과 생성 모델 결합, 초기 설정 시 수십 개의 GPU, 이후 유지보수 시 몇 개의 GPU
Prompt Engineering	매우 낮음	없음 (또는 최소한의 GPU)	기존 모델에 특정 작업을 위한 프롬프트 설계, 추가 학습 불필요

RAG(Retrieval-Augmented Generation)

- RAG 모델은 텍스트 생성 작업을 수행하는 모델로 **주어진 소스 데이터로부터 정보를 검색**하고, 해당 정보를 활용하여 원하는 **텍스트를 생성**하는 과정을 수행



✓ **데이터 처리:** 원본 소스 데이터를 **청크(Chunk)단위**의 작은 조각으로 나눔(splitting) → 텍스트를 숫자로 전환하는 **임베딩(Embedding)** → **벡터 저장소(Vector store)**에 저장

✓ **검색** 및 **생성**: 쿼리가 주어지면, 의미기반 검색 사용 **쿼리 임베딩**을 통해 벡터 DB에서 **청크 검색** → 디코딩하여 생성 과정에 활용 → LLM으로 **텍스트 생성**

RAG(Retrieval-Augmented Generation)

- **“Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks”**

  Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, Douwe Kiela

  22,May,2020

- 사전 훈련된 모델의 한계점
  - 새로운 지식을 업데이트하거나, 기존 지식을 수정하기 어려움
  - hallucination(사실이 아닌 것을 사실처럼 말하는 경우) 문제가 있음
- parametric memory와 non-parametric (retrieval-based) memory를 결합한 하이브리드 모델들
  - (당시 기준) REALM, ORQA는 encoder-only 구조로 이루어졌기 때문에 open-domain extractive QA에 대한 가능성만 탐구
  - 이 논문에서는 이런 hybrid 접근법에 seq-seq 구조를 도입해 좀 더 넓은 가능성을 탐구
- 논문에서 제안하는 모델
  - parametric memory: BART
  - non-parametric memory(document index): dense vector of Wikipedia
  - retriever: DPR(Dense Passage Retriever)

RAG(Retrieval-Augmented Generation)

- Notations
  - x: input sequence
  - z: text document to retrieve
  - y: target sequence to generate
  - x가 들어왔을 때, y를 생성하기 위해 z를 참고

- Components
  - $p_\eta(z|x)$: retriever
    - η: parameter
    - 쿼리 x가 주어졌을 때 passage들에 대한 distribution을 리턴
    - DPR 사용
  - $p_\theta(y_i|x, z, y_{1:i-1})$: generator
    - θ: parameter
    - 인풋 x와 검색된 passage z, 이전의 $i-1$개의 토큰을 기반으로 다음 토큰 생성
    - BART 사용 (단, 어떠한 encoder-decoder든 사용가능)
  - 이 retriever와 generator는 훈련 과정에서 동시에 학습
  - output을 산출하기 위해 retrieve 된 document에 대해 marginalize 하게 되는데, 저자들은 이 marginalize* 방식을 다르게 한 두 가지 모델인 RAG-Sequence와 RAG-Token을 제안

    *marginalize는 output y에 대한 확률만을 구하기 위해 각 document z에 대해 구해진 곱사건의 확률을 모두 더하는 것

RAG(Retrieval-Augmented Generation)

- Models
  - RAG-sequence model

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

  - RAG-token model

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

  - RAG-Sequence는 document에 대한 값을 sequence 단위로 고려한 다음 marginalize
  - RAG-Token은 document에 대한 값을 token단위로 고려한 다음 marginalize 하고, 다음 token을 생성하면서 sequence를 생성
- DPR (retriever)

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

  - $d(z)$는 document encoder를 통해 산출되는 dense representation, $q(x)$는 query encoder를 통해 산출되는 query representation
  - x에 대한 z의 분포는 $d(z)$와 $q(x)$의 내적 연산을 기반으로 하여 산출
  - 내적 값이 높은 순서대로 top-k document를 골라 검색(retrieve)

RAG(Retrieval-Augmented Generation)

- **"Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks"**

  Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, Douwe Kiela

  22,May,2020

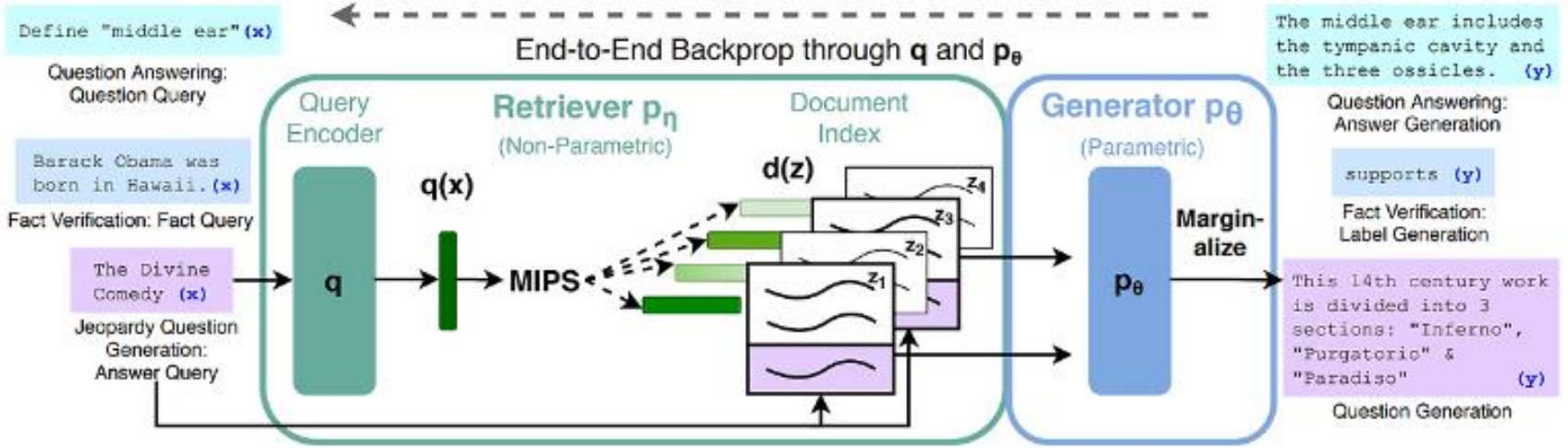


Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder* + *Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

RAG(Retrieval-Augmented Generation)

- 벡터 데이터베이스(Vector Database)란?
  - 벡터 데이터베이스는 LLM의 장기 기억 부족 문제를 해결하기 위해 개발된 새로운 유형의 데이터베이스
  - 고차원 실수 벡터 인덱스를 효율적으로 저장하고 관리하는데 특화되어 있으며 벡터 데이터베이스는 전통적인 데이터베이스와 다르게 쿼리를 실수 벡터(Embedding) 형태로 표현
  - Ex. Chroma, FAISS, Pinecone, Weaviat 등

- 서비스 구현을 위한 프레임워크: ex. Langchain
  - LangChain은 생성형 AI의 언어 모델을 활용하여 애플리케이션을 개발할 수 있도록 지원하는 오픈 소스 프레임워크
  - OpenAI, Hugging Face 등 다양한 LLM 모델을 지원하며, 사용자는 자신의 요구에 맞는 모델을 선택하여 사용
  - LangChain의 핵심적인 개념은 LLM 프롬프트의 실행과 외부 소스의 실행을 엮어 Chaining하는 것

RAG(Retrieval-Augmented Generation)

- RAG 예시 ("LLM 애플리케이션 아키텍처를 활용한 생성형 AI 서비스 구현: RAG모델과 LangChain 프레임워크 기반", 2023, 정천수)

〈표 4〉 소스 데이터 타입

Doc Type	Python Code
DOC	from LangChain.document_loaders.word_document import UnstructuredWordDocumentLoader loader = UnstructuredWordDocumentLoader("근무 복장 기준.docx")
TXT	from LangChain.document_loaders import TextLoader loader = TextLoader("지급보험금계산.txt")
PDF	from LangChain.document_loaders import PyPDFLoader loader = PyPDFLoader("휴직 및 복직 관리기준.pdf")
CSV	from LangChain.document_loaders.csv_loader import CSVLoader loader = CSVLoader("회사생활가이드(QA형).csv")

```
!pip install -q pypdf  # Library to extract text from pdf document
from langchain.document_loaders import PyPDFLoader
loader = PyPDFLoader("휴직 및 복직 관리기준.pdf")
pages = loader.load_and_split()
data = loader.load()
print(f"{len(data)} documents, contain {len(data[0].page_content)} words")
```

```
1 documents, contain 1021 words
```

〈그림 10〉 소스 데이터 수집 및 추출

```
### Splitting Chunks
from langchain.text_splitter import RecursiveCharacterTextSplitter
text_splitter = RecursiveCharacterTextSplitter(chunk_size = 300,
                                               chunk_overlap = 0)
all_splits = text_splitter.split_documents(data)

### Embeddings & Vector stores
!pip install chromadb # 벡터스토어
!pip install tiktoken # 토큰 계산용
import tiktoken   #tiktoken is a fast BPE tokeniser for use with OpenAI's models.

from langchain.llms import OpenAI
from langchain.embeddings import OpenAIEmbeddings
from langchain.vectorstores import Chroma

vectorstore = Chroma.from_documents(documents=all_splits, embedding=OpenAIEmbeddings())
```

〈그림 11〉 청크 분할 및 임베딩

RAG(Retrieval-Augmented Generation)

- RAG 예시 (“LLM 애플리케이션 아키텍처를 활용한 생성형 AI 서비스 구현: RAG모델과 LangChain 프레임워크 기반”, 2023, 정천수)

```
question = "휴직 기간에 대하여 알려줘"

docs = vectorstore.similarity_search(question)
len(docs)
print(f"{len(data)}개의 문서에 {len(data[0].page_content)}개의 단어를 가지고 있습니다")
```

```
1개의 문서에 1021개의 단어를 가지고 있습니다
```

```
from langchain.chat_models import ChatOpenAI
from langchain.chains import RetrievalQAWithSourcesChain

retriever = vectorstore.as_retriever(search_kwargs={"k": 1})
llm = ChatOpenAI(model_name="gpt-3.5-turbo", temperature=0)
chain = RetrievalQAWithSourcesChain.from_chain_type(
    llm=llm,chain_type="stuff",retriever = retriever,return_source_documents=True)

result = chain(question)
result
```

```
{'question': '휴직 기간에 대하여 알려줘',
 'answer': '휴직기간은 휴직 사유에 따라 다양한 기간 내에서 필요에 따라 부여될 수 있습니다. 육아휴직의 경우 최대 2년이며, 진학의 경우 국내진학은 최대 2년, 해외진학은 학위 기간 + 출입국 기간을 포함하여 최대 2.5년입니다. 어학의 경우 최대 1년입니다.₩n',
 'sources': '휴직 및 복직 관리기준.pdf',
 'source_documents': [Document(page_content='가족돌봄이 필요하여 1개월 이상 정상적인 직무를 ₩n 수행할 수 없다고 인정될 경우 ₩n사사육아 - 육아로 인해1개월 이상 정상적인 직무를 수행할 수 ₩n 없다고 인정되는 경우 ₩n기타 - 가족돌봄을 제외한 기타 개인사정 (건강관리 , 임신기 , ₩n 육아等 )으로1개월 이상 정상적인 직무를 수행할 수 ₩n 없다고 인정될 경우 ₩n3. 휴직기간 ₩n휴직기간은 휴직 사유별로 다음과 같은 기간 내에서 필요에 따라 부여할 수 있음 ₩n(1) 휴직 기간 ₩n₩uf06c 육아휴직 : 최대2年, 1회 분할 사용가능 (상세내용은 육아휴직 기준 참조) ₩n₩uf06c 진학: 국내진학은 학위 기간 내 최대2년, 해외진학은 학위 기간 + 출입국 ₩n기간(0.5년)내 출입국 기간을 포함하여 최대2.5년 ₩n₩uf06c 어학: 최대1년(국내 최소3개월 이상, 해외 최소6개월 이상)', metadata={'page': 0, 'source': '휴직 및 복직 관리기준.pdf'})]}
```

〈그림 12〉 질문(User Prompt)과 검색(Search)결과

```
from langchain.prompts.chat import (ChatPromptTemplate,SystemMessagePromptTemplate,HumanMessagePromptTemplate,)
system_template="""다음 내용을 참조하여 사용자의 질문에 간단히 답변할것.
문서와 질문에 대한 요약(summaries)이 주어지면 참조("SOURCES")를 사용하여 최종 답변을 작성할것.
답을 모른다면 "모른다"고 말하고 답을 만들어 내려고 하지 말것.
----------
{summaries}

한국어와 마크다운 형식으로 답변할것:"""

messages = [
    SystemMessagePromptTemplate.from_template(system_template),
    HumanMessagePromptTemplate.from_template("{question}") ]
prompt = ChatPromptTemplate.from_messages(messages)
```

```
from langchain.chains import RetrievalQAWithSourcesChain
chain_type_kwargs = {"prompt": prompt}
chain = RetrievalQAWithSourcesChain.from_chain_type(
    llm=llm, chain_type="stuff", retriever = retriever, return_source_documents=True,chain_type_kwargs=chain_type_kwargs)
```

```
question = "휴직 기간에 대하여 알려줘"
result = chain(question)
result['answer']
```

```
'휴직기간은 휴직 사유에 따라 다음과 같은 기간 내에서 필요에 따라 부여됩니다.₩n₩n1. 육아휴직: 최대 2년, 1회 분할 사용 가능합니다. 상세 내용은 육아휴직 기준을 참조하시면 됩니다.₩n₩n2. 진학: 국내진학은 학위 기간 내 최대 2년, 해외진학은 학위 기간 + 출입국 기간(0.5년)을 포함하여 최대 2.5년까지 가능합니다.₩n₩n3. 어학: 최대 1년까지 가능하며, 국내는 최소 3개월 이상, 해외는 최소 6개월 이상이 필요합니다.₩n₩n이상입니다. 추가로 궁금한 사항이 있으시면 말씀해주세요.'
```

```
question = "육아 휴직 기간 알려줘!"
result = chain(question)
result['answer']
```

```
'육아휴직은 최대 2년까지 가능하며, 1회 분할 사용이 가능합니다. 자세한 내용은 "육아휴직 기준"을 참조해주세요.'
```

```
question = "근무복장 규정 알려줘!"
result = chain(question)
result['answer']
```

```
'근무복장 규정에 대한 내용은 제공되지 않았습니다. 죄송합니다.'
```

〈그림 13〉 LLM을 활용한 답변 생성

References

- Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
- Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." arXiv preprint arXiv:2106.09685 (2021).
- Lewis, Patrick, et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks." Advances in Neural Information Processing Systems 33 (2020): 9459-9474.
- Li, Yinheng, et al. "Large language models in finance: A survey." Proceedings of the Fourth ACM International Conference on AI in Finance. 2023.
- 정천수. "LLM 애플리케이션 아키텍처를 활용한 생성형 AI 서비스 구현: RAG모델과 LangChain 프레임워크 기반." 지능정보연구 29.4 (2023): 129-164.
- 정천수. "도메인 특화 LLM: Mistral 7B를 활용한 금융 업무분야 파인튜닝 및 활용 방법." 지능정보연구 30.1 (2024): 93-120.