

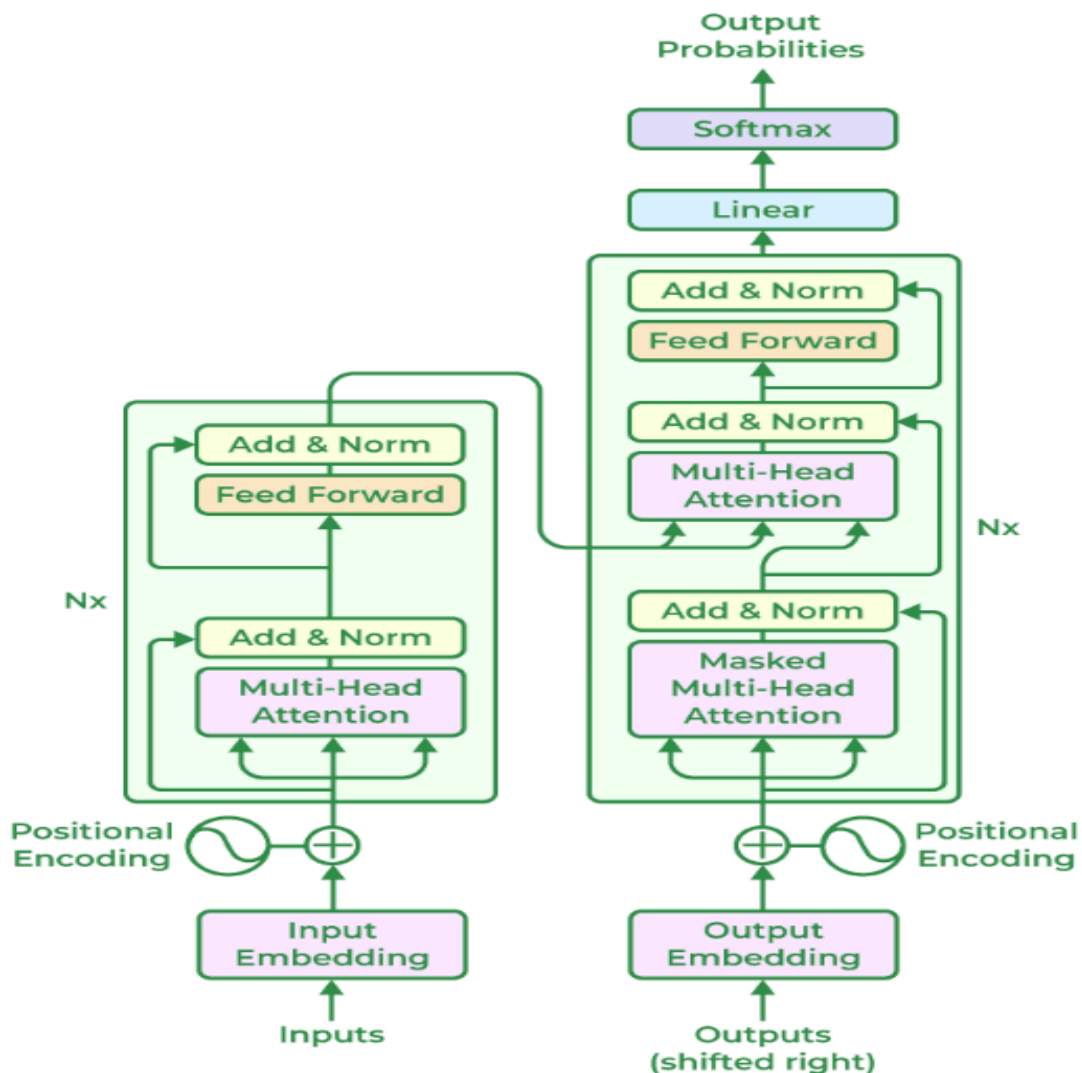
1. Large Language Model

서비스를 위한 LLM의 종류에 대한 설명

읽기 전에 알아보는 LLM 기초 상식

- Large Language Model(LLM)은 Language Model(LM)에서 발전한 것
- LM은 문장에서 특정 단어의 뒤에 어떤 단어가 나올 확률을 계산하는 것 ex) 오늘 점심은 [다음 나올 단어 ↗ 구내식당-90%, 외식-9.9%, 휴지-0.01% ...]
 - 확률에 의해 추출된 단어를 연결(concat)시켜 새로운 문장을 생성
 - 확률을 계산하기 위해 문장(=학습 데이터)을 학습시킴
 - 따라서, 학습 데이터인 문장에 의존적, 도메인 의존적인 모델이 구축됨
 - 데이터를 학습시킨다는 것은 신경망의 parameter를 update 한다는 뜻
- LLM은 LM보다 훨씬 많은 parameter를 가진 큰 신경망 (Transformer 구조는 동일)
 - LM보다 더 많은 데이터들을 학습할 수 있음 ↗ 다양한 질문에 대한 답변 가능(일반화)
 - LM보다 더 복잡한 연산을 통해 확률을 계산할 수 있음 ↗ 고품질 추론 가능(성능)
- Parameter가 많아질수록 다양한 질문에 대한 답변(일반화) 및 답변의 품질(성능)이 높아짐
- Parameter가 많아질수록 학습 및 연산(답변) 과정이 길어짐
 - 서비스에 이용하기 위해서는 GPU 필요
 - 연산을 위한 parameter 담길 공간인 메모리 또한 중요

Transformer Architecture



📌 LLM with GPU

- 🏠 가정집에서 LLM을 직접 돌려보려는 사람을 위한 GPU 선택 이야기

1.1 Foundation Model

👁️ 정의

방대한 양의 데이터를 학습하여 특정 Task를 지정하지 않아도 다양한 Task를 수행할 수 있는 대규모 언어 모델

- 빅테크에서 서비스 중인 LLM인 GPT, Gemini 등이 예시
- Task: 요약, 분류, 번역, 생성, QA 등

🔴 한계점

- 모델이 공개된 것이 아닌 API 사용이므로 사용하기는 편리하나 금융, 방산 등 기업 적용시 한계 ☞ Cloud Service
- 일반 기업에서는 opt out 방식으로 정보를 활용
- opt out: 정보 주체의 동의를 받지 않고 개인정보를 수집, 이용한 후 당사자가 거부 의사를 밝히면 개인정보 활용을 중지

1.2 Open Model

👁️ 정의

누구나 사용할 수 있도록 공개한 대규모 언어 모델

- 기업에서 컨트롤 가능하며, 오너십을 가지고 운영 가능
- 최초 모델 다운로드 후 오프라인에서 사용가능 ☞ On-prem 가능
- Hugging Face에서 제공하는 LLaMA, Mistral 등이 예시

🔴 한계점

- 영어 기반임으로 한국어에 대한 답변 품질이 좋지 못함

☀️ 대안

- 한국어 데이터 기반으로 학습한 모델들이 등장
- HuggingFace에서 ko로 검색하면 나오는 모델들
- Llama, Orion 등

1.3 sLLM

👁️ 정의

Large Language Model에 비해 Parameter 개수가 작은 모델

- GPT 3.0의 Parameter는 약 1700억 (모델 버전이 올라갈수록 높아짐)
- sLLM의 가장 작은 버전은 약 70억개

🔴 한계점

- 일반화 성능이 떨어지며 품질이 좋지 못할 수 있음

☀️ 대안

- 일반적인 상황이 아닌 특정 도메인(화학, 법률, 의료 등)에서 활용될 수 있도록 학습수행

📌 더 알아보기

- llama, mistral 등 다양한 open model 중 용량이 작은 7B, 14B 등의 모델을 sLLM이라고 부름
- 📖 [작지만 오히려 좋아! 소형 언어모델\(sLLM\)](#)

2. Using on Service

서비스를 위한 LLM의 사용 방법에 대한 설명

- 위 두 종류의 제공되는 LLM은 기업의 데이터를 알지 못함 → 기업 정보 질의에 대한 답변 불가능
- 따라서, 기업 서비스에 사용하려면 **프롬프트 엔지니어링** 또는 **학습** 이 필요하며 이들은 기업의 데이터에 기반함
- prompt eng: 언어모델에 사족을 다는 과정

2.1 Prompt Engineering

👁️ 정의

LLM에게 특정 작업을 수행하도록 지시하는 프롬프트 설계를 하는 과정

- **프롬프트 설계** 및 **프롬프트 최적화**를 하는 것으로 모델의 연산을 위한 parameter는 변화하지 않음
 - 프롬프트 설계: 모델이 주어진 작업을 수행하기 위해 필요한 정보를 프롬프트에 포함시키는 것
 - 프롬프트 최적화: 모델의 성능을 극대화 하기 위해 다양한 프롬프트를 시도하고 조정함
- 질의에 모델이 작업을 수행하는 방법에 대한 예시를 제공하는 **작업 예시** 포함 방법도 있음

🔴 한계점

- 학습되지 않은 정보에 대해서 답변하는데 한계가 존재함
- 질의시마다 관련된 예시를 생성해야 함
- 예시에 국한된(overfitting) 답변을 생성할 수 있음

☀️ 대안

- **RAG**: 관련된 정보를 DB 내에서 검색하여 예시를 풍부하게 증강함
- **Fine tuning**: 특정한 정보를 학습시켜 답변이 가능하도록 함

2.2 Few-shot Learning(FSL)

👁️ 정의

LLM에 소수의 예시를 사용하여 빠르게 학습하는 기술

- 데이터 수집이 어렵거나 비용이 많이 드는 상황에서 유리함
- 새로운 작업이나 도메인에 빠르게 적응할 수 있음 (전문용어, 새로운 도메인 등)

🔴 한계점

- 일반화 능력의 한계: Few-shot learning은 제공된 예제와 유사한 작업에는 잘 동작하나, 예제와 다른 상황에서는 일반화가 어려움

- 프롬프트 민감성: Few-shot learning의 성능은 제공된 프롬프트의 질과 형식에 크게 의존함

예시

Few shot learning 학습 데이터

Q: What is the capital of France?

A: Paris

Q: What is the capital of Italy?

A: Rome

Q: What is the capital of Japan?

A: Tokyo

더 알아보기

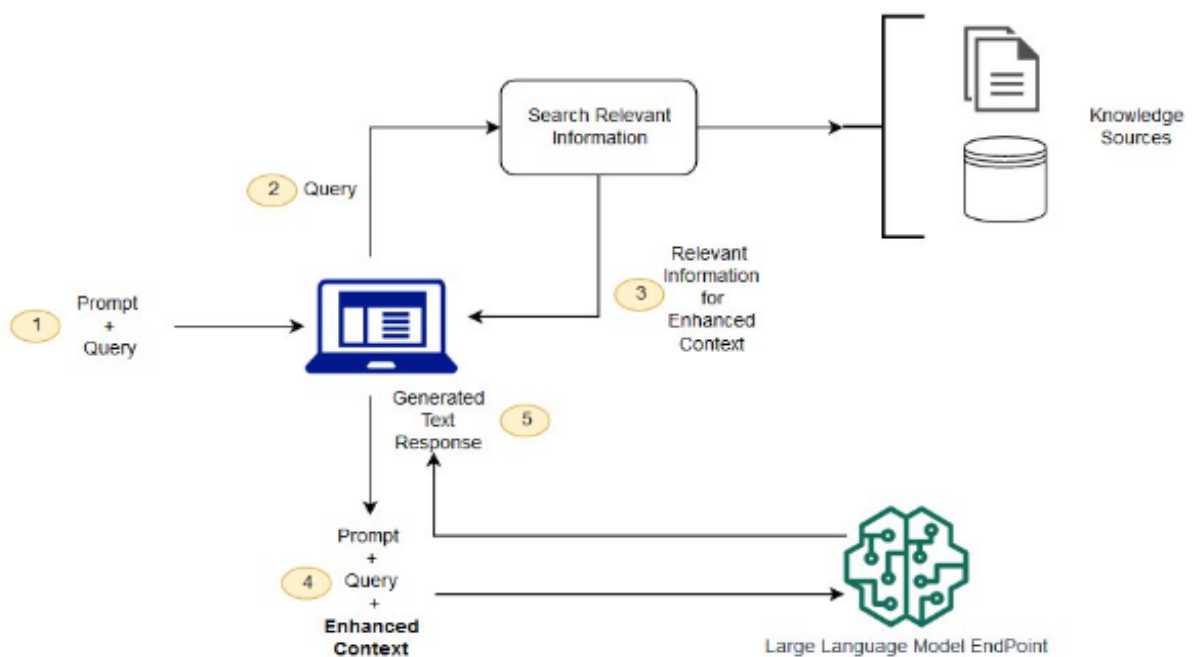
-  [프롬프트 엔지니어링 기초1-퓨샷러닝](#)

2.3 RAG(Retrieval-Augmented Generation)

프롬프트에 검색을 통한 정보를 포함하여 정보에 기반한 응답을 생성하는 기술


- 검색을 통해 질의와 유사한 정보를 추출하는 검색 엔진 필요
- 프롬프트에 정보를 포함하기 때문에 프롬프트 엔지니어링의 범주에 속한다고 볼 수 있음
- Prompt Engineering**과 동일하게 모델 내의 parameter는 수정되지 않음

Rag Architecture

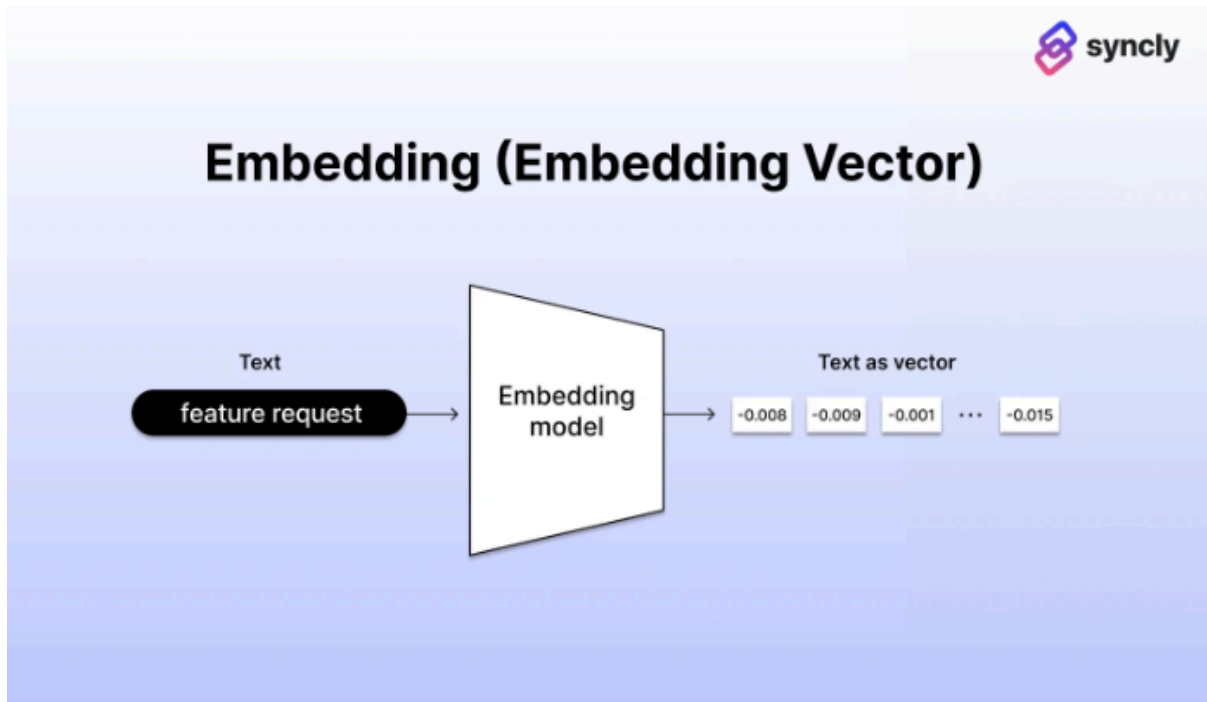


AWS의 Retrieval Augmented Generation (RAG) 아키텍처 설명

1. 프롬프트와 질문내용을 LLM 서비스에 전달하여 질의

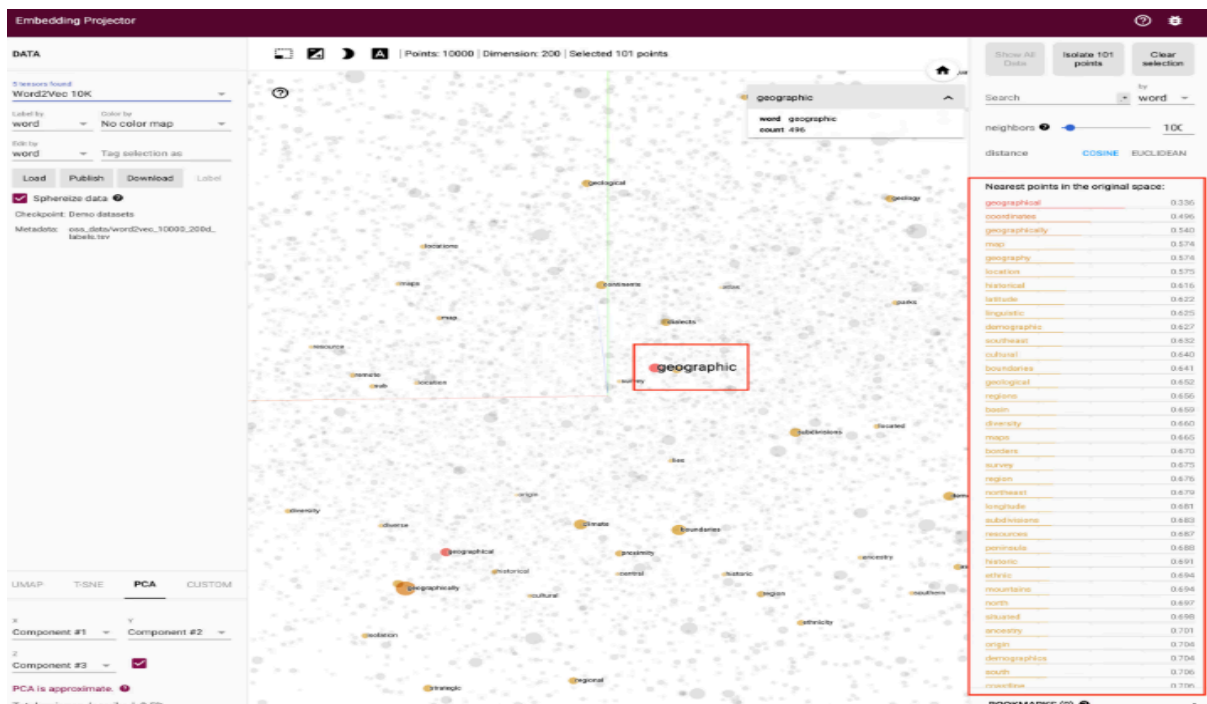
- 질문 내용과 유사한 문서를 **Vector Storage**에서 검색 (보유한 문서는 미리 저장되어 있음  아래 Vector Storage 참조)
- 검색한 문서를 LLM 서비스에 전달x
 - 프롬프트의 길이는 한정적이기 때문에 적절한 문서 수에 대한 설정이 필요
- 전달받은 문서를 활용해 향상된 프롬프트 질의 생성 및 LLM 전달
 - 2.1 Prompt Engineering**의 FSL참조
 - 프롬프트에 추가적으로 예시를 전달해주는 형식
- LLM의 수행 결과 전달

Vector & Embedding



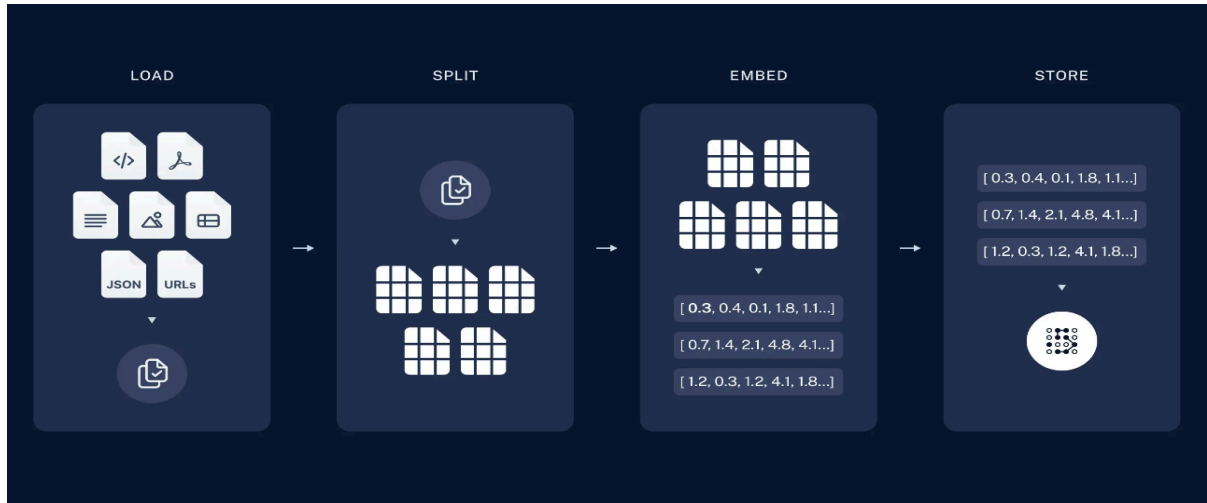
자연어의 임베딩 과정

- Vector: 실수의 집합
- Embedding: 텍스트를 실수로 표현한 결과



Vector Storage

LLM 질의에 활용할 보유 문서들은 임베딩(embedding)과정을 거쳐 Vector Storage(DB)에 저장됨



RAG 검색증강 생성 설명

- **LOAD:** 검색에 활용할 문서들을 불러옴
- **SPLIT:** 문서를 특정 기준에 따라 구문(chunk)으로 나누는 과정
 - 프롬프트의 길이는 한정적이기 때문에 문서 전체를 질의에 담을 수 없음
 - 정보가 잘 분산될 수 있는 단위로 나누는 과정이 필요
 - 나누는 방법은 다양하게 존재
- **EMBED:** chunk를 모델이 활용할 수 있는 숫자로 변환하는 과정
 - chunk는 작은 단위인 토큰(단어와 비슷함)으로 나누어짐
 - 토큰은 임베딩 모델을 통해 의미를 포함한 숫자로 변환됨
 - 임베딩 모델은 다양하게 존재(openAI, huggingface 등..)
 - 각 chunk에 포함된 토큰들이 변환된 숫자 집합(vector) 생성
- **STORE:** 변환된 숫자의 집합인 vector는 이를 저장할 수 있는 vector storage에 저장됨
 - Vector Storage는 다양한 종류가 있음

장점

- **정확성 향상:** 주어진 문서 내에서 답을 찾으려 하여 잘못된 답을 하는 할루시네이션 문제를 최소화
- **도메인 특화:** 어떠한 도메인의 기업이라도 보유한 데이터를 기반으로 별도 학습 없이 성능이 좋은 LLM(GPT, Gemini)를 통해 고품질 답변 생성 가능

한계점

- **데이터 의존성**
 - 높은 품질의 검색 데이터를 필요로 함
 - 문서에 기반한 답을 하기에 최신의 데이터가 아닐경우 잘못된 답변이 나올 수 있음
- **검색엔진의 한계**
 - 벡터 유사도를 통해 유사 문서를 찾기에 관련성이 낮거나 적절하지 않은 문서가 검색될 수 있음

Example

Query 📄 LLM

[프롬프트] (검색된 기존 데이터)

리뷰: 영화는 환상적이었고 매 순간을 즐겼습니다.

감성: 긍정적

리뷰: 줄거리가 지루하고 예측 가능했습니다.

감정: 부정적

리뷰: 연기는 훌륭했지만 뛰어나지는 않았습니다.

감성: 중립

[질문]

리뷰: 영화 촬영은 놀라웠지만 이야기의 깊이가 부족했습니다.

이 리뷰의 감성은 뭐야?

프롬프트에 대한 모델 수행 및 결과

부정

📌 더 알아보기

- 📖 RAG와 Few-shot learning의 상호작용

2.4 Fine tuning

👁 정의

사전 훈련(pretrained)된 대형 언어 모델을 특정 작업이나 도메인에 맞추어 추가로 훈련시키는 과정

- 사전 훈련된 모델이란 제공자가 대형 데이터 셋으로 미리 학습시킨 모델
- 사전 훈련되지 않은 모델은 모델의 아키텍처만 가지고 있는 것은 parameter가 초기 상태
- GPT, Llama 등 앞서 소개한 Large Language Model 들이 모두 사전 훈련 모델로 볼 수 있음
- 사전 훈련된 LLM은 언어에 대한 일반적인 이해도를 갖추고 있기 때문에 대부분의 작업 수행 가능
- Fine tuning은 특정 작업을 수행하도록 전문화된 모델을 얻기 위해 사용
- 도메인에 적합한 데이터 셋을 통해 사전훈련된 LLM의 parameter를 update

🟢 장점

- LLM이 일반적인 지식 외에 추가 정보 없이 특정 도메인에 대한 대답도 가능하게 함

🔴 한계점

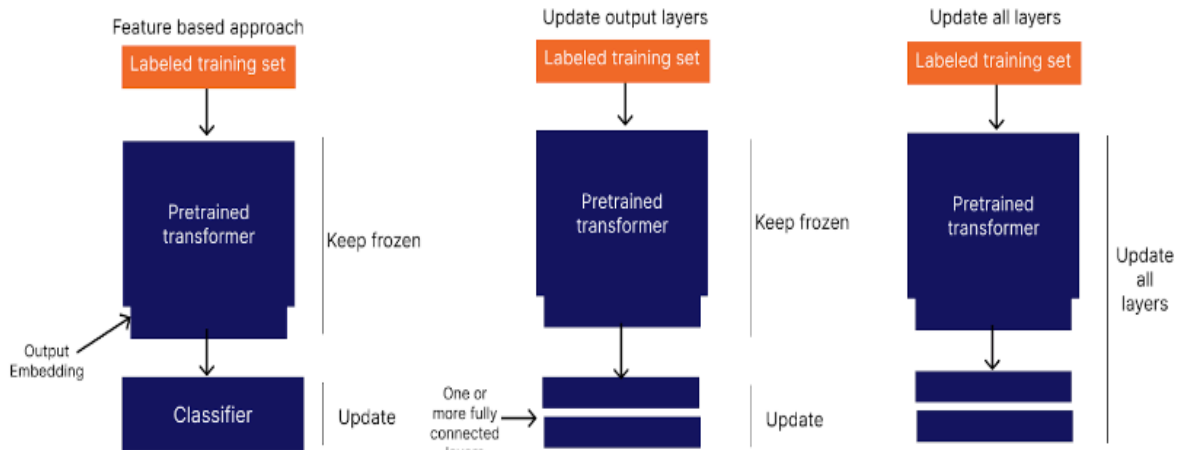
- 학습을 위한 구체적인 데이터 세트가 필요함
- 학습을 위한 인프라가 필요함 (GPU-모델에 따른 memory)

📌 Transfer Learning

👁 정의

Pre-trained Model(사전 학습 모델)에 새로운 지식을 학습시켜 다른 작업에 활용되게 하는 기술

- fine tuning은 transfer learning의 한 방법



Transfer Learning의 방법들

- Feature-based approach: pre-trained transformer의 parameter는 고정하고 classification layer를 update
- Update the output layers: pre-trained transformer의 parameter는 고정하고 output layer(fc)를 update
- update all layers: 전체 모델의 parameter를 update
- 오른쪽으로 갈 수록 fine tuning의 비용과 성능 향상 효과가 증가함
- 학습 비용을 줄이기 위해 Parameter Efficient Fine Tuning(PEFT) 가 고안되었으며 예시로 Low-Rank Adaption(LoRA) 등이 있음
- 오른쪽으로 갈 수록 미세 조정의 비용과 성능 향상 효과가 증가함

📌 Prompt Engineering vs Fine Tuning

	Prompt Engineering	Fine Tuning
목적	학습 없이 더 나은 답변을 얻기 위함	특정 도메인에서 향상된 성능을 얻기 위함
방식	많은 정보가 포함된 입력을 수행	특정 도메인에 대한 데이터셋으로 학습
장점	학습을 위한 리소스가 필요 없음	특정 도메인에 대해 깊이 있는 답변을 얻을 수 있음
단점	좋은 prompt를 생성하기 위해 작업이 필요	학습을 위한 리소스가 필요함 (GPU)

📌 더 알아보기

- 📖 Prompt tuning의 방법 soft prompt

2.5 🧡 PEFT 🧡

👁️ 정의

LLM의 전체 parameter를 update하는 것이 아닌 소수의 (추가) 모델 파라미터만 fine tuning하여 계산 및 저장 비용을 줄이는 기법

- LLM의 전체 parameter를 update하는 것은 많은 비용 발생
- PEFT는 사전 학습된 대규모 모델을 다양한 애플리케이션에 효율적으로 적용하기 위한 라이브러리
- 성능 또한 모든 parameter를 fine tuning 한 것과 비슷함

🟢 장점

- 비용이 절약됨 ☞ 가성비 모델 생성

🔴 한계점

- 완벽한 성능을 보장하진 않음 (만능이 아님)
- 전문적이며 정확한 지식이 필요한 곳에는 **fully fine tuning**을 권장
- 새로운 데이터에 대한 지속적인 **fine tuning** 필요
- 과거의 데이터를 잊지 못하며 원문 위치, 첨부파일 제공 어려움

📌 LoRA

👁 정의

대규모 LLM을 학습하는 대신 저차원의 model을 학습하여 update 해야할 parameter의 수를 줄이는 기법

- Transformer 구조의 각 layer에 학습 가능한 **rank decomposition matrices**를 삽입
- **Downstream task**에서의 학습가능한 파라미터 수를 크게 줄임

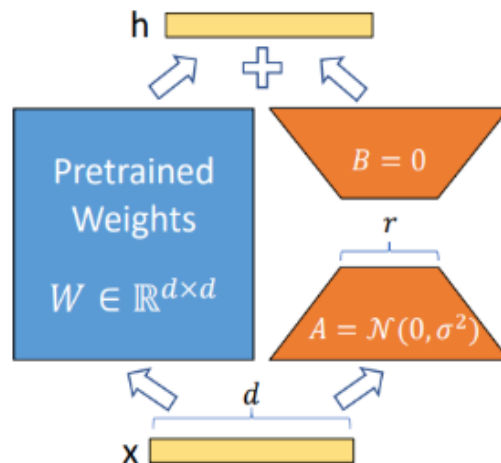
🖋 알고리즘

1. 가정

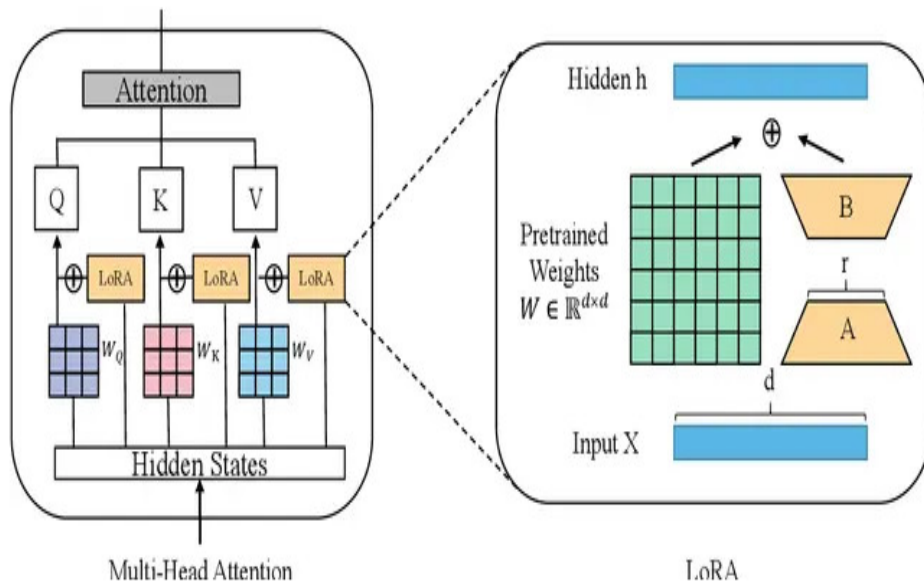
1. 고도화된 LLM이 가지고 있는 parameter가 내가 하고싶은 일에 비해 과도하게 많다
2. 내가 필요한 parameter는 일부분이라고 생각함
3. intrinsic dimension, Low Rank(deep learning)

2. 방법론

$$h = w_0x + \Delta x = W_0x + BAx$$



LoRA 아키텍처



LoRA 적용

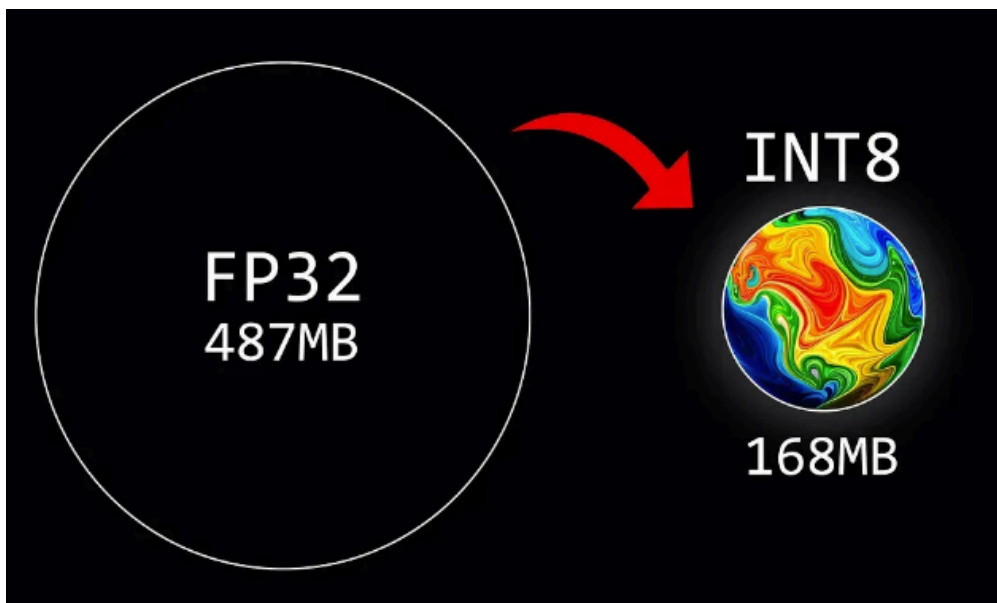
- Pretrained Weights는 freeze
- A, B 저차원의 weight matrix를 학습

📌 Quantization

👁 정의

LLM의 parameter를 실수형에서 정수형으로 바꾸어 비트수를 줄이는 과정

- 32bit 부동 소수점 형태를 8bit로 변환 ⇔ 모델 size 4배 축소
- 크기가 줄어들고 계산 효율성이 향상
- 비트수를 N배 줄이면 곱셈의 복잡도는 $N \times N$ 으로 감소
- 추론 속도와 메모리 사용량도 두배에서 네배까지 효율적



양자화 설명