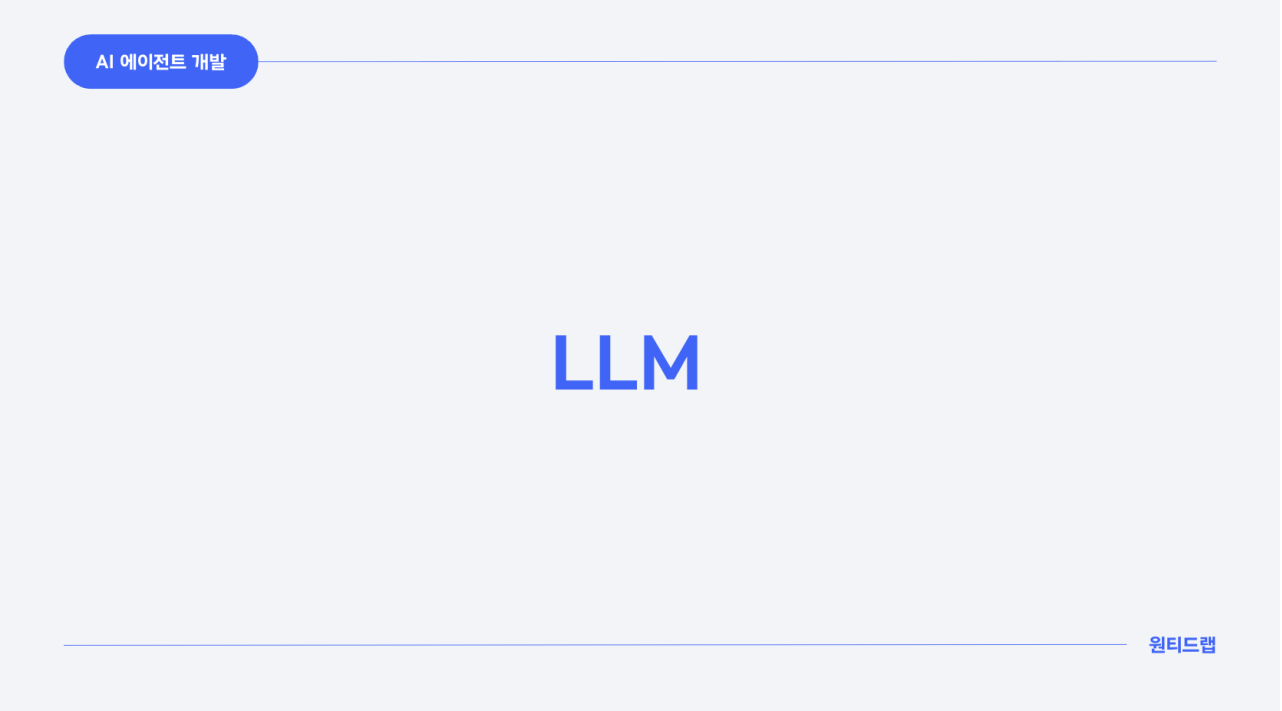
04 LLM



1. LLM의 등장

1) Transformer

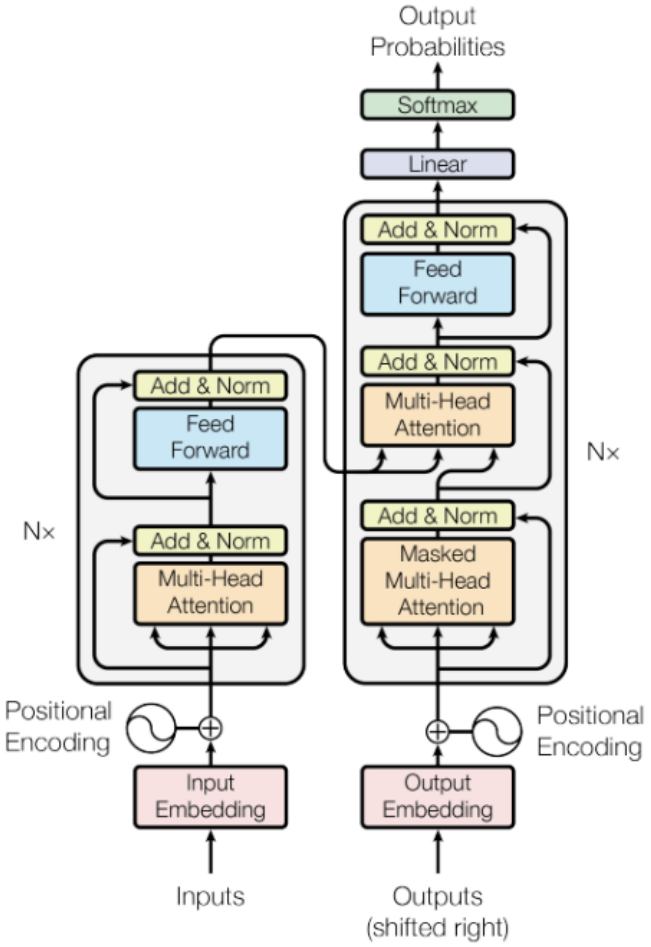
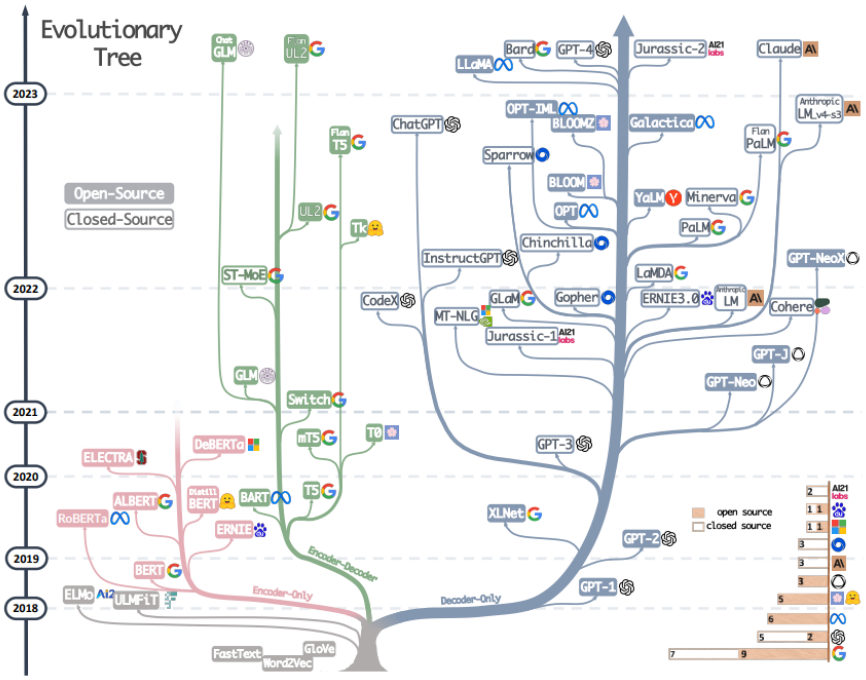


Figure 1: The Transformer - model architecture.

2) LLM 역사와 발전



- 2018~2020: BERT, GPT-2 → 언어 이해/생성 초기 단계
- 2020~2021: GPT-3, T5, mT5, Codex → 본격적 대규모 모델
- 2022~현재: GPT-4, LLaMA, Mistral, Qwen, DeepSeek, Claude 등
- **가지가 갈라지는 기준**
  - 개발 주체(기업/연구기관)
  - 오픈소스 vs 클로즈드소스
  - 모델 크기(파라미터 수)
  - 학습 데이터의 범위(언어/도메인 특화 여부)

3) Foundation 모델

- 대규모 범용 데이터로 학습 → 다양한 태스크에 활용 가능한 기반 모델
- 대표: GPT-4, LLaMA, Claude, Mistral, Qwen, DeepSeek 등

4) 우리나라의 Foundation LLM

- 네이버 HyperCLOVA X
- 카카오 KoGPT
- LG EXAONE
- 업스테이지 Solar (HuggingFace 리더보드 상위권 기록)

> 🖊 **왜 한국어 특화 모델이 필요할까?**
>
> - 영어 중심 모델은 한국어 어휘/문법/문화 맥락에 약함
> - 공공·교육·고객센터 등 한국어 비중이 큰 서비스에서 성능 차이 발생
> - 따라서 한국어 데이터로 학습한 **한국어 특화 Foundation 모델** 필요

2. LLM 관련 용어 살피기

1) 모델 이름 속 숫자 (7B, 70B)

- **B = Billion(10억)** → 파라미터 개수
- 7B = 70억 개 파라미터, 70B = 700억 개 파라미터
- 비유: 뇌세포 수 (많을수록 똑똑하지만 메모리 소모 ↑)
- 실행 기준 (예시):
  - 7B → 8–16GB VRAM
  - 13B → 24GB 이상 VRAM 권장

2) 내 컴퓨터 사양 확인하기

- **RAM**: 전체 작업 공간
- **GPU VRAM**: 모델이 올라가는 공간

사양 확인 방법

- Windows: `작업 관리자 → 성능 탭`
- Mac: `  → 이 Mac에 관하여`
- GPU: 터미널에서 `nvidia-smi`

`nvidia-smi`

```
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 560.94                 Driver Version: 560.94         CUDA Version: 12.6     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                  Driver-Model | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA GeForce RTX 4070 ...  WDDM  |   00000000:01:00.0 Off |                  N/A |
| N/A   43C    P8              1W /   28W |     647MiB /   8188MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|    0   N/A  N/A      5324    C+G   ...inaries\Win64\EpicGamesLauncher.exe      N/A      |
|    0   N/A  N/A     21012      C   ...al\Discord\app-1.0.9207\Discord.exe      N/A      |
|    0   N/A  N/A     23240      C   ...al\Programs\LM Studio\LM Studio.exe      N/A      |
|    0   N/A  N/A     26828      C   ...al\Programs\LM Studio\LM Studio.exe      N/A      |
|    0   N/A  N/A     28504      C   ...al\Programs\LM Studio\LM Studio.exe      N/A      |
+-----------------------------------------------------------------------------------------+
```

영역	의미	예시
Driver Version / CUDA Version	현재 설치된 NVIDIA 드라이버 버전과 해당 드라이버가 지원하는 최대 CUDA 버전	Driver: 560.94 / CUDA: 12.6
GPU Name	GPU 모델 이름	NVIDIA GeForce RTX 4070
Fan / Temp / Perf	팬 속도, GPU 온도, 성능 상태(P0~P8)	43C, P8 (절전 상태)
Pwr: Usage/Cap	GPU 전력 사용량과 최대 전력 용량	1W / 28W
Memory-Usage	현재 GPU 메모리 사용량 / 전체 용량	647MiB / 8188MiB
GPU-Util	GPU 연산 점유율(%)	0%
Processes	현재 GPU를 사용 중인 프로세스 목록 (PID, 실행 파일)	Discord.exe, LM Studio.exe 등

☆ 중요하게 봐야 할 부분

- **Memory-Usage**: 내가 실행할 모델이 들어갈 여유 공간이 있는가?
- **GPU-Util**: 현재 GPU가 얼마나 바쁘게 일하고 있는가?
- **Processes**: 누가 GPU를 사용 중인지 확인 (PID와 프로그램 이름으로 구분)

3) 파인튜닝과 관련 용어

파인튜닝이란 이미 학습된 모델을 특정 목적에 맞게 다시 학습하는 것을 말한다.

구분	기법	주요 특징	자원 요구사항	적합한 상황
Full Fine-Tuning	전체 파인튜닝	모델의 모든 파라미터를 새 데이터로 학습	높은 GPU 메모리, 긴 학습 시간, 대규모 데이터	대규모 인프라 보유, 높은 성능 필요
SFT (Supervised Fine-Tuning)	지도학습 기반 파인튜닝	정답이 있는 데이터로 일부/전체를 재학습	데이터 라벨링 필요, GPU 자원 소모	특정 태스크(예: 고객센터 답변) 최적화
PEFT (Parameter-Efficient FT)	LoRA (Low-Rank Adaptation)	저차원 행렬을 추가, 핵심 파라미터만 학습	상대적으로 적은 GPU 메모리	제한된 컴퓨팅 자원 환경
	QLoRA (Quantized LoRA)	LoRA + 양자화 → 더 작은 자원으로 학습	소규모 서버/개인 PC 가능	최소 자원, 온디바이스 학습
	어댑터 (Adapters)	각 층 사이에 작은 신경망 추가	다중 어댑터 전환 가능	여러 태스크를 동시에 다룸
Instruction Tuning	지시어 튜닝	다양한 명령어 형식을 학습 (“요약해줘”, “설명해줘”)	데이터 다양성 필요	모델을 사용자 친화적으로
최적화 관련	양자화 (Quantization)	숫자 정밀도 줄여 모델 크기 축소 (32bit→4bit)	GPU 메모리 절약, 속도↑	개인 PC 실행, 배포 환경
	GGUF	로컬 실행 최적화된 모델 파일 포맷	CPU/GPU 모두 가능	Ollama, LM Studio 등에서 활용

ⓘ 양자화(Quantization)

- **왜 필요한가?**
- 원래 모델은 파라미터를 32비트(float32) 숫자로 저장
- → 너무 크고 무거움 (수십 GB ~ 수백 GB)
- 개인 PC/서버에서 실행하기 어려움
- **어떻게 하는가?**
- 숫자의 정밀도를 줄여 저장
- 예시: 32비트 → 16비트 → 8비트 → 4비트
- "100.123456" → "100.12"처럼 필요한 자리수만 남기고 단순화

4) 성능 평가

- **방법**: 시험 문제처럼 다양한 질문을 던져 점수화
- **대표 벤치마크**
  - MMLU: 대학 시험 문제 모음
  - HellaSwag: 상식 추론 문제
  - TruthfulQA: 거짓 없는 답변 평가
- **참고**:
  - HuggingFace Open LLM Leaderboard
  - LLM 한국어 리더보드

3. 로컬 LLM 다루기

1) 왜 로컬 LLM인가?

- 클라우드 LLM (ChatGPT 등)
  - 장점: 설치 필요 없음, 최신 모델 사용 가능
  - 단점: 비용 발생, 데이터 보안 우려, 인터넷 의존
- 로컬 LLM
  - 장점: 내 PC/서버에서 실행 → 데이터 보안↑, 비용↓, 오프라인 사용 가능
  - 단점: PC 사양에 따라 속도와 가능 모델 제한

2) 주요 툴 비교

툴	특징	장점	단점
vLLM	대규모 서비스 배포용, 빠른 추론·메모리 효율	서버 환경에서 대량 요청 처리 최적화	설치와 설정이 비교적 복잡
LM Studio	GUI 기반, 사용자 친화적	클릭 몇 번으로 모델 실행 가능, 개발 지식 불필요	세밀한 커스터마이즈 어려움
Ollama	Mac/Windows 간단 설치, CLI & API 제공	명령어 한 줄로 실행, Python 등과 쉽게 연동	고급 설정에 한계 있음

3) Ollama 기본 명령어

- 모델 설치: `ollama pull <model_name>`
- 모델 실행: `ollama run <model_name>`
- 설치된 모델 목록: `ollama list`
- 실행 중 모델 확인: `ollama ps`
- 모델 삭제: `ollama rm <model_name>`