GLOSSARY

용어사전 — 한 줄로 찾아보기

모듈을 읽다가, 혹은 회의에서 낯선 용어를 만나면 여기서 찾으세요. 각 항목의 [모듈 N]을 누르면 자세히 배우는 곳으로 이동합니다.

용어 검색 — 예: CUDA, 양자화, 골든셋…

컴퓨터 기초 (워밍업)

서버 / 클라이언트

요청을 받아 처리해주는 컴퓨터(서버)와 부탁하는 쪽(클라이언트, 우리 PC). 식당의 주방과 손님 관계. **[워밍업]**

API

프로그램끼리 정해진 형식으로 요청·응답을 주고받는 창구. 자판기처럼 정해진 버튼(형식)을 누르면 정해진 결과가 나옵니다. 사람용 창구가 화면이라면, 프로그램용 창구가 API. **[워밍업]**

터미널 (명령창)

마우스 클릭 대신 글자 명령을 타이핑해 컴퓨터를 조작하는 프로그램. 모듈의 `ollama run` 같은 명령은 전부 여기 입력합니다. Windows에선 PowerShell. **[워밍업]**

데이터베이스 (DB)

자료를 체계적으로 저장·검색하는 시스템 — 엑셀 표가 수백만 행으로 커지고 여러 사람이 동시에 쓸 수 있게 된 것. 쿼리는 DB에 던지는 질문. **[워밍업]**

정규식 (정규 표현식)

"010-####-#### 모양인가"처럼 글자 모양의 규칙을 기계적으로 검사하는 방법. 같은 입력엔 백 번이고 같은 판정 — 검증 게이트의 1순위 도구. **[워밍업]** **[모듈 5]**

비트 / 바이트 / GB / B(10억)

비트=0 또는 1 하나. 바이트=비트 8개(그래서 16비트=2바이트). GB≈10억 바이트(파일·메모리 용량 단위). 모델 이름의 "7B"는 Billion=파라미터 70억 개 — 용량의 B와 다른 B. **[워밍업]**

오픈소스

설계도(소스코드)를 공개하고 누구나 무료로 쓰고 고칠 수 있게 허락한 소프트웨어. llama.cpp와 오픈 모델들이 이 방식. **[워밍업]**

n8n

코드 없이 블록을 이어 업무 자동화 흐름(워크플로우)을 만드는 도구. 우리 실습에서 sLM API를 연결하는 조립대로 씁니다. **[모듈 5]**

하드웨어 · 인프라

CPU

소수의 강력한 코어(계산을 실제 수행하는 처리 단위 — 일꾼 한 명)로 복잡한 판단·순차 작업을 처리하는 중앙 처리 장치. 작은 모델(1~4B)은 CPU만으로도 느리게나마 구동됩니다. **[모듈 1]**

GPU

수천 개의 단순 코어로 같은 계산을 대량 병렬 처리하는 장치. LLM 추론(거대 행렬 곱셈)의 주력 하드웨어. **[모듈 1]**

NPU

AI 계산에 특화해 설계된 반도체의 총칭. 전력 효율이 강점이며 추론 시장을 중심으로 GPU 독주에 도전 중. 도입 판단 기준은 "내 모델·엔진이 실제로 도는가". **[모듈 7]**

RAM

CPU의 작업 공간(주기억장치). 모델이 VRAM을 넘치면 이곳으로 오프로딩됩니다. **[모듈 1]**

VRAM

그래픽카드에 붙은 GPU 전용 메모리. 용량이 곧 "올릴 수 있는 모델 크기의 상한". 스펙에서 가장 먼저 볼 숫자. **[모듈 1]**

메모리 대역폭

메모리를 초당 몇 GB 읽고 쓸 수 있는지. 토큰 생성 속도의 이론상 상한 ≈ 대역폭 ÷ 모델 크기. **[모듈 1]**

오프로딩

VRAM에 다 안 들어가는 모델의 일부를 RAM/CPU로 넘겨 실행하는 것. 돌아가긴 하지만 수 배~수십 배 느려지는 최후의 수단. **[모듈 1]**

CUDA

NVIDIA GPU에 일반 계산을 시키는 프로그래밍 플랫폼. 20년 치 도구·라이브러리·개발자 생태계가 NVIDIA 독주의 진짜 성벽. 드라이버 버전이 지원 CUDA 버전의 상한을 정합니다. **[모듈 1]** **[모듈 7]**

컴파일러 / 빌드

사람이 쓴 소스코드를 장비가 실행할 기계 명령으로 번역하는 프로그램 / 그 번역 과정. 같은 llama.cpp도 어떤 CPU·GPU용으로 빌드했느냐에 따라 도는 곳이 달라집니다. **[모듈 4]**

SDK

Software Development Kit — 특정 플랫폼용 소프트웨어를 만들 때 필요한 컴파일러·라이브러리·문서의 묶음. CUDA 툴킷이 대표적인 GPU용 SDK. 버전 궁합이 "안 도는 이유"의 단골. **[모듈 4]**

드라이버

운영체제와 GPU를 잇는 기본 소프트웨어. 오래된 드라이버는 최신 CUDA·도구를 막는 병목이 됩니다. 장비 조사 시 GPU 이름과 함께 확인. **[모듈 1]**

폐쇄망

인터넷과 분리(또는 제한 연결)된 내부 전용 네트워크. 클라우드 AI 사용이 막히는 대신, 데이터가 밖으로 나가지 않는 환경 — 우리 연구회의 전제 조건. **[모듈 0]**

Docker (도커)

프로그램과 실행 환경을 통째로 포장해 어디서든 같게 실행되게 하는 컨테이너 기술. 서버 서빙 배포의 사실상 표준. **[모듈 4]**

k8s (쿠버네티스)

컨테이너 수십~수백 개를 여러 서버에 자동 배치·복구하는 오케스트레이션 도구. 대규모용 — 서버 1대 규모에선 불필요. **[모듈 4]**

HBM

고대역폭 메모리. AI 가속기에 붙는 초고속 메모리로, GPU뿐 아니라 K-NPU 차세대 칩들도 채택. **[모듈 7]**

파운드리 / 팹리스

파운드리는 남이 설계한 반도체의 생산만 대신해주는 공장(삼성 파운드리, TSMC). 팹리스는 공장 없이 설계만 하는 회사(텐스토렌트, 리벨리온). **[모듈 7]**

명령어 규격 (RISC-V)

칩에게 "이렇게 계산하라"고 지시하는 공통 문법. RISC-V는 특정 회사 소유가 아니라 누구나 쓸 수 있게 공개된 규격 — 텐스토렌트가 개방 전략의 기반으로 삼는 것. **[모듈 7]**

TOPS

칩의 이론상 최대 연산 속도 단위(초당 조 단위 연산). 소프트웨어가 못 받쳐주면 이론값일 뿐 — 실측이 기준. **[모듈 7]**

모델 · 추론 원리

LLM / sLM

거대언어모델 / 소형언어모델. sLM은 수십억(1~30B) 파라미터급으로 로컬 장비에서 구동 가능한 크기. 좁은 태스크에서는 이미 실용 수준. **[모듈 0]**

파라미터

모델을 이루는 숫자(가중치)들. "7B"는 70억 개라는 뜻. 파라미터 수 × 저장 비트가 모델 파일 크기를 정합니다. **[모듈 3]**

토큰

모델이 글을 읽고 쓰는 최소 단위(글자 조각). 요금·속도·컨텍스트 한도가 전부 토큰 단위. 한국어는 영어보다 토큰이 더 드는 경향. **[모듈 2]**

학습 (training)

대량 데이터로 파라미터를 조정해 모델을 '만드는' 과정. 대규모 GPU·긴 시간이 들고, 끝나면 파라미터가 고정됩니다. 모델은 추론(사용) 중에 배우지 않습니다. **[모듈 0]**

테스트타임 컴퓨트 (test-time compute)

학습을 더 하는 대신 추론 시점에 더 생각하게(더 많은 추론 토큰·시도) 해서 성능을 끌어올리는 접근. thinking 모드가 대표 사례. **[모듈 7]**

추론 (inference)

학습이 끝난 모델을 '사용'하는 것 — 입력을 넣고 출력을 받는 과정. 학습(training)과 대비되며, AI 연산의 무게중심이 이쪽으로 이동 중. **[모듈 7]**

Transformer

현대 LLM의 표준 두뇌 구조. 어텐션 메커니즘으로 문맥 전체에서 중요한 부분을 골라 다음 토큰을 예측합니다. **[모듈 2]**

어텐션 (attention)

새 토큰을 만들 때 앞선 모든 토큰과의 관련도를 계산해 "어디를 볼지" 정하는 장치. KV 캐시가 필요한 이유. **[모듈 2]**

컨텍스트 윈도우

모델이 한 번에 기억하며 참고할 수 있는 토큰 수의 상한. 넘기면 오류가 나거나 앞부분을 잊습니다. 크게 잡을수록 KV 캐시 메모리를 더 먹음. **[모듈 2]**

프롬프트 / 시스템 프롬프트

모델에게 주는 지시문. 시스템 프롬프트는 역할·규칙을 고정하는 상위 지시. 출력이 마음에 안 들 때 가장 먼저, 비용 0으로 손댈 곳. **[모듈 5]**

Prefill / Decode

입력 전체를 읽어들이는 단계(prefill — 첫 반응까지의 침묵) / 토큰을 하나씩 생성하는 단계(decode — 초당 토큰 속도). **[모듈 2]**

KV 캐시

이미 읽은 토큰의 계산 결과를 GPU 메모리에 저장해 재계산을 피하는 장치. 컨텍스트가 길수록, 동시 사용자가 많을수록 VRAM을 차지 — "같이 쓰면 느려지는" 이유. **[모듈 2]**

thinking 모드

답하기 전에 긴 추론 토큰을 속으로 생성하는 방식. 어려운 문제 품질↑, 속도는 수십 배↓. 필요한 단계에만 선택적으로. **[모듈 2]**

MoE (혼합전문가)

모델 안의 분류기(라우터)가 입력을 보고 필요한 '전문가' 일부만 깨우는 구조. 총량은 크되 활성이 작아, 대역폭이 약한 장비에서 빠른 생성 속도를 얻는 열쇠. **[모듈 3]**

환각 (hallucination)

모델이 모르는 것을 그럴듯하게 지어내는 현상. 다음 토큰 예측기라는 본질에서 나옴. RAG(근거 제공)와 게이트(기계 검증)로 방어. **[모듈 6]**

양자화 · 모델 포맷

양자화 (quantization)

파라미터를 더 적은 비트로 저장해 모델을 줄이는 기술(정밀한 반올림). 크기↓ 속도↑, 품질은 소폭 타협 — 내 골든셋으로 측정해 판단. **[모듈 3]**

FP16 / INT8

숫자 저장 방식. FP16은 16비트 부동소수점(파라미터당 2바이트, 7B≈14GB), INT8은 8비트 정수(절반 크기). **[모듈 3]**

Q4_K_M / Q6_K / Q8_0

GGUF 양자화 등급 표기. 숫자가 비트 수(작을수록 가볍고 품질 타협). 실무 기본값은 Q4_K_M, 품질을 더 지키려면 Q6_K·Q8_0. **[모듈 3]**

GGUF

llama.cpp 계열(Ollama·LM Studio 포함)이 읽는 로컬 추론용 단일 파일 포맷. HuggingFace에서 "GGUF" 붙은 저장소를 받는 이유. **[모듈 3]**

GPTQ / AWQ

GPU 서빙(vLLM 등)용 양자화 방식. GGUF가 llama.cpp용이라면 이쪽은 신형 GPU 엔진용 — 포맷과 엔진은 짝이 있습니다. **[모듈 4]**

safetensors

학습 직후의 원본(비양자화) 모델을 배포하는 안전한 표준 포맷. 로컬 추론용으로는 보통 이걸 양자화해 GGUF 등으로 변환. **[모듈 4]**

파인튜닝 (fine-tuning)

이미 학습된 모델을 내 데이터로 추가 학습시켜 말투·형식을 고정하거나 좁은 태스크에 특화하는 것. 결정 순서의 마지막 수단(프롬프트→RAG→파인튜닝). **[모듈 5]**

서빙 · 운영

서빙 (serving)

모델 파일을 메모리에 올려 채팅·API 요청을 받아 처리하는 것. 파일이 악보라면 서빙 엔진은 연주자. **[모듈 4]**

llama.cpp

로컬 추론 엔진의 사실상 표준(GGUF의 본가). 구형 GPU·CPU·멀티 GPU까지 지원 폭이 가장 넓음. llama-server로 API 제공. **[모듈 4]**

Ollama

llama.cpp를 감싸 명령 두 개(pull/run)로 쓰게 만든 개인 실습 표준 도구. **[모듈 4]**

LM Studio

코드 없이 화면 클릭으로 모델 검색→다운로드→채팅하는 GUI 도구. 비개발자의 첫 로컬 LLM 경험용. **[모듈 4]**

vLLM

연속 배칭으로 대규모 동시 사용자를 감당하는 운영급 서빙 엔진. 단, 신형 GPU 전제 — 구형(2016년경의 파스칼 세대 등)은 미지원. **[모듈 4]**

TGI / SGLang / TensorRT-LLM

그 외 서빙 엔진들 — HuggingFace 통합 운영(TGI), 구조화 출력·에이전트 고속 처리(SGLang), NVIDIA 한계 성능(TensorRT-LLM). 전부 비교적 신형 GPU 전제. **[모듈 4]**

배칭 (batching)

여러 요청을 묶어 GPU가 놀지 않게 한꺼번에 처리하는 기법. 동시 사용자 처리량의 핵심. **[모듈 4]**

OpenAI 호환 API

ChatGPT API와 같은 요청·응답 형식. 데이터는 내 서버에만 가며, 이 규격 덕에 엔진을 바꿔도 도구(n8n 등)가 그대로 붙습니다. **[모듈 4]**

HuggingFace

누구나 자기가 만든 AI 모델을 올리고, 남이 받아갈 수 있는 세계 최대 모델 공유 사이트. 모델 카드에서 크기·양자화·라이선스를 확인합니다. **[모듈 4]**

TPS (tokens/sec)

초당 생성 토큰 수. 사람 읽기 속도(약 10~15 TPS)가 대화형 서비스의 기준점. 벤치의 1차 지표. **[모듈 2]**

RAG · 에이전트

RAG

검색 증강 생성. 질문과 관련된 내부 문서 조각을 검색해 프롬프트에 근거로 붙여, 모델이 모르는 사내 지식으로 답하게 만드는 방법(오픈북 시험). **[모듈 5]**

임베딩 (embedding)

문장을 의미 좌표(숫자 벡터)로 바꾸는 별도의 작은 모델. "비슷한 의미 = 가까운 좌표"라서 단어가 달라도 의미로 검색됩니다. **[모듈 5]**

벡터 DB

임베딩 좌표를 저장하고 "질문과 가까운 문서 조각"을 빠르게 찾아주는 데이터베이스. RAG의 검색 창고. **[모듈 5]**

재랭킹 (re-ranking)

1차 검색 결과를 더 정밀한 모델로 다시 순위 매겨 상위만 쓰는 것. RAG 정확도를 올리는 2차 필터. **[모듈 5]**

챗봇 / 워크플로우 / 에이전트

사람이 묻고 답하는 형태 / 정해진 순서의 부품으로 실행되는 형태(n8n) / 도구를 골라 쓰며 목표까지 스스로 반복하는 형태. 워크플로우로 충분하면 에이전트를 쓰지 않는다. **[모듈 5]**

도구 호출 (tool calling)

모델이 "이 함수를 이 인자로 실행해달라"고 요청하는 능력. DB 조회·검색·계산 같은 실제 행동의 토대. 모델 선정 시 지원 여부 확인 항목. **[모듈 5]**

하네싱 / 루프 엔지니어링

모델 출력을 기계적으로 검증하고 실패 시 재실행시키는 구조를 설계하는 일. "증거 없이는 통과되지 않는 구조"가 품질을 만든다. **[모듈 0]**

검증 게이트

모델의 "완료" 주장을 정규식(맨 위 '컴퓨터 기초' 카테고리 참고)·형식검사·테스트처럼 같은 입력에 늘 같은 판정이 나오는 검사로 확인하는 관문. FAIL이면 자동 재실행. **[모듈 5]**

평가

골든셋 (Golden Set)

입력과 정답 쌍으로 이뤄진 평가용 문제지. 50~100건이면 의미 있는 비교 시작. 모델 비교·양자화 확인·프롬프트 개선을 전부 같은 문제지로. **[모듈 6]**

합성 데이터

실데이터의 분포(유형·길이·난이도)를 분석해 LLM으로 생성한 가짜 데이터. 개인정보 0건으로 실전 난이도의 평가를 가능하게 함. **[모듈 6]**

정답률 / 근거성 / 지연 / 처리량

평가 4대 지표 — 정답 일치 비율(Accuracy) / 근거에만 기반했나(Groundedness) / 응답까지 시간(Latency) / 단위 시간당 처리 건수(Throughput). 태스크마다 주 지표가 다름. **[모듈 6]**

LLM-as-judge

더 강한 모델에게 채점 기준표를 주고 채점을 맡기는 방법. 기계 채점이 어려운 품질 판단용. 반출 허용 데이터만, 결과는 사람이 샘플링 재확인. **[모듈 6]**

← 돌아가기
전체 목차

다음 →
자가진단 · 6레벨 질문