MODULE 00

왜 폐쇄망 로컬 AI인가

본격적인 기술 내용에 들어가기 전에, 우리가 왜 이 공부를 하는지부터 정리합니다. 이 모듈만 읽어도 연구회의 방향이 보입니다.

01 ChatGPT가 있는데 왜 직접 모델을 돌리나요?

ChatGPT, Claude 같은 서비스는 **클라우드**에서 돌아갑니다. 내가 입력한 텍스트가 외부 회사의 서버(내 요청을 받아 처리해주는 원격 컴퓨터)로 전송되고, 답변이 돌아오는 구조입니다. 편리하지만 우리 업무 환경에서는 두 가지 벽에 부딪힙니다.

- **데이터가 밖으로 나간다** — 민원 원문, 내부 문서, 소스코드에는 개인정보와 내부 정보가 들어 있습니다. 외부 서버로 보내는 순간 규정 위반이 될 수 있습니다.
- **폐쇄망에서는 접속 자체가 안 된다** — 우리 업무망은 인터넷이 제한적으로만 열려 있습니다. 클라우드 AI에 상시 의존하는 업무 설계는 애초에 불가능합니다.

그래서 방향을 뒤집습니다. **모델을 우리 컴퓨터 안으로 가져오는 것** — 이것이 로컬 AI이고, 우리 연구회 AXIS(한국특허정보원 사내 학습동아리 CoP 가운데 폐쇄망 sLM·에이전트를 연구하는 그룹)의 출발점입니다.

> 💡 **비유**
>
> 번역이 필요할 때마다 외부 번역 회사에 문서를 보내는 것(클라우드)과, 번역사를 우리 사무실에 채용하는 것(로컬)의 차이입니다. 사무실 번역사는 외부 유출이 없고, 인터넷이 끊겨도 일하며, 건당 요금도 없습니다. 대신 우리가 책상(하드웨어)을 마련해줘야 하죠.

02 3분 기초 — 학습과 추론, AI의 두 국면

더 나아가기 전에, 가장 기초적인 구분 하나를 정확히 해둡시다. AI 이야기에는 **학습(training)**과 **추론(inference)**이라는 두 국면이 있습니다.

- **학습 = 모델을 만드는 과정.** 수조 개의 문장을 읽히며 모델 속 수십억 개의 숫자(파라미터)를 조금씩 조정합니다. 수천 장의 GPU(그래픽카드 — 모듈 1에서 자세히 다룹니다)로 몇 달씩 걸리는, 소수 기업만 할 수 있는 일이고 — 끝나면 파라미터가 고정된 '완성품'이 나옵니다.
- **추론 = 그 완성품을 쓰는 것.** 질문을 넣으면 답이 나오는 매 순간이 추론입니다. 이때 모델은 아무것도 배우지 않습니다 — 파라미터를 읽기만 하고, 바꾸지 않습니다.

> 💡 **비유**
>
> 요리학교에서 6년간 훈련받는 것이 학습, 졸업한 요리사가 주방에서 주문받아 요리하는 것이 추론입니다. 요리사는 주문마다 학교를 다시 다니지 않습니다. 손님이 "덜 짜게요"라고 하면 그 자리에서 조절할 뿐(프롬프트), 몸에 밴 기술(파라미터)이 바뀌는 것은 아닙니다. 이렇게 주문에 맞게 요청 문구를 다듬는 요령을 **프롬프트 엔지니어링**이라고 부릅니다.

여기서 챗봇을 써본 분들이 가장 많이 갖는 오해가 풀립니다 — **"모델은 나와 대화하면서 점점 배워가는 것 아닌가요?"** 아닙니다. 모델은 배우지 않습니다 — 대화 내용은 작업 기억(컨텍스트)에 잠시 담길 뿐이고, 새 대화를 열면 사라집니다. 모델 자체를 바꾸는 것은 별도의 학습 과정(파인튜닝, 모듈 5)뿐입니다.

그리고 이 구분이 산업 전체의 방향을 설명합니다. 모델을 만드는 일(학습)은 소수 기업이 가끔 하지만, 만들어진 모델을 쓰는 일(추론)은 전 세계가 매일 수십억 번 합니다. 그래서 **AI 연산의 무게중심이 학습에서 추론으로 이동**하고 있습니다 — 요리학교 정원은 그대로인데 전국에 새로 생기는 식당은 계속 늘어나는 것과 같습니다. 이 흐름 속에서 우리 연구회가 하는 모든 활동(서빙(모델을 장비에 올려 쓸 수 있는 상태로 돌리는 것 — 모듈 4의 주제), 하네싱, 평가)이 바로 이 추론의 영역입니다. 이 이동이 반도체 시장을 어떻게 흔드는지는 심화 모듈(모듈 7)에서 다룹니다.

03 sLM — 작은 모델이라는 선택지

GPT-4 같은 최상위 모델은 수천억 개의 파라미터를 가진 거대한 프로그램이라 개인 장비에서는 돌릴 수 없습니다. 하지만 최근 몇 년 사이 **sLM(small Language Model, 소형언어모델)**이 빠르게 발전했습니다. 파라미터 수십억 개(4B~30B급, B=Billion=10억) 규모로, **좋은 게이밍 노트북이나 사무실 GPU 서버에서 충분히 돌아갑니다.**

물론 sLM은 최상위 클라우드 모델보다 성능이 떨어집니다. 대략 수개월의 격차가 있다고들 말합니다(대형 모델이 몇 달 전에 보여주던 수준이라는 뜻입니다). 그런데 여기에 중요한 반전이 있습니다:

업무를 잘게 쪼개면, 작은 모델로도 충분한 일이 많습니다.
"민원 하나를 3줄로 요약해라", "이 문의를 12개 유형 중 하나로 분류해라", "이 텍스트에서 전화번호를 가려라" — 이런 좁고 명확한 태스크는 sLM이 이미 실용 수준입니다. 올해 상반기에 연구회 리더가 실제 민원 업무를 대상으로 미리 실증해서 확인한 사실이기도 합니다 — 그 과정과 결과는 앞으로 연구회 활동에서 함께 뜯어봅니다.

🔗 **우리 연구회에선**

우리 연구회의 전략이 정확히 이것입니다. 큰 인프라 예산을 기다리는 대신, **지금 있는 장비에서 도는 작은 모델 + 잘게 쪼갠 업무 단위**로 실제 자동화를 만들어보고, 그 실측 결과를 근거로 미래 인프라를 제안하는 것.

04 모델보다 구조 — 하네싱이라는 감각

이 연구회에서 여러분이 가져갈 가장 중요한 감각은 이것입니다: **AI의 성능은 모델 크기만으로 정해지지 않습니다. 모델을 감싸는 구조(하네스)가 결과를 바꿉니다.**

같은 모델이라도 —

- ✔ 그냥 "요약해줘"라고 시키면: 가끔 틀리고, 틀려도 모릅니다.
- ✔ "요약해줘 → 결과에 개인정보가 남았는지 **기계적으로 검사** → 남았으면 **다시 시켜**"라는 구조에 넣으면: 틀린 결과가 통과하지 못합니다.

모델을 바꾼 게 아닙니다. **증거 없이는 통과되지 않는 구조** 하나를 끼웠을 뿐입니다. 6월 발표에서 본 "완료를 의심하라" 데모가 바로 이것이고, 이런 구조를 설계하는 일을 **하네싱(harnessing)** 또는 루프 엔지니어링이라고 부릅니다.

> 💡 **비유**
>
> 신입 직원(모델)에게 일을 맡길 때, 유능한 신입을 뽑는 것도 중요하지만 **보고서 검토 절차(구조)**를 만드는 게 더 확실합니다. 검토 절차가 있으면 평범한 신입도 사고를 안 치고, 절차가 없으면 유능한

> 신입도 언젠가 사고를 칩니다.

이 감각은 작은 로컬 모델에서 익히면 **토큰 비용 없이 무한정 연습**(토큰=글자 조각 단위, 모듈 1에서 설명합니다)할 수 있고, 나중에 어떤 최신 모델을 만나도 그대로 통합니다. 모델은 계속 바뀌지만 구조를 설계하는 능력은 남기 때문입니다.

05 이 학습 세션의 지도

앞으로 7개 모듈에서 아래를 순서대로 배웁니다. 각 모듈 끝의 퀴즈까지 풀면 목차에 완료 표시가 붙습니다.

모듈	주제	이걸 배우면 답할 수 있는 질문
1	하드웨어 기초	내 장비에서 어떤 모델까지 돌릴 수 있나?
2	LLM 동작 원리	왜 길게 넣으면 느려지고, 동시에 쓰면 더 느려지나?
3	양자화	품질을 조금 양보하고 속도·메모리를 얻는 법은?
4	서빙 엔진	Ollama, llama.cpp, vLLM 중 뭘 언제 쓰나?
5	RAG와 에이전트	챗봇을 넘어 업무를 '실행'시키려면?
6	평가	어떤 모델이 "좋다"고 말할 근거는?
7 (심화)	AI 반도체 생태계	NVIDIA 독주는 왜, K-NPU는 어디까지 왔나?

↗ 심화 학습

LLM 전체 그림을 1시간에 잡고 싶다면: **Andrej Karpathy — Intro to Large Language Models** · 영어(자막 지원). OpenAI 창립 멤버가 비전공자 눈높이로 설명하는 무료 공개 영상 — 이 모듈의 "학습과 추론" 구분과 정확히 겹칩니다.

폐쇄망 로컬 AI의 극단 사례 — 인터넷 없이 위키백과+로컬 AI+지도를 통째로 담는 오픈소스(누구나 무료로 쓰고 고칠 수 있게 코드를 공개한 것): **Project NOMAD (GitHub)** · "오프라인에서도 AI가 돈다"는 우리 방향의 좋은 참고 구현입니다.

CHECK

이해했는지 확인해 봅시다

틀려도 괜찮습니다 — 오답을 고르면 왜 아닌지 설명이 나오고, 정답을 찾을 때까지 다시 고를 수 있습니다. 점수는 첫 시도 기준입니다.

문제 1 / 5

Q1. 우리 업무 환경에서 클라우드 AI를 상시 업무에 쓰기 어려운 '가장 근본적인' 이유는?

① 클라우드 모델은 사내 전문용어를 학습하지 못했기 때문

② 내부 데이터가 외부 서버로 전송되고, 폐쇄망에서는 접속 자체가 제한되기 때문

③ 클라우드 모델은 같은 질문에도 답이 매번 달라 업무에 쓸 수 없기 때문

④ 사용량이 늘면 API 요금이 자체 서버 구축비를 넘어서기 때문

← 이전
전체 목차

다음 →
모듈 1 · 하드웨어 기초