생성형 AI 기반 RAG 개발자 과정

LLM 발전사
Rule-based 에서 ChatGPT 까지

Session 1 / 33 — Day 1

왜 지금 LLM 인가 — 50년 AI 역사를 한 줄기로

박수현 · (사)한국정보공학기술사회 · 2026

LLM 이론 · LangChain · ChromaDB CRUD · Agent — 4일 핸즈온

이번 시간을 마치면

🧭 AI 발전의 흐름을 이해한다

- 생성형 AI = **판별이 아니라 '생성'** 하는 모델
- **AI ⊃ ML ⊃ DL ⊃ 생성형 AI** 포함 관계
- **"규칙을 사람이" → "패턴을 데이터로"** 의 전환
- NLP 기본기 — **토큰화 · 임베딩 · NER**
- RNN·LSTM → **Transformer** 로의 진화

🚀 핵심 전환점을 설명한다

- 2017 **Transformer** 가 바꾼 것
- **GPT 계보** (2018 GPT-1 → 2025 GPT-5)
- 2022 **ChatGPT** 가 만든 변곡점

> 🎯 이 세션은 코드보다 **"왜"** 에 집중합니다. 이후 실습 전체의 바탕이 됩니다.

PART A

PART A
AI 패러다임의 발전사

> Rule-based → Machine Learning → Deep Learning — 약 12분

AI · ML · DL · 생성형 AI 의 포함 관계

> 💡 안쪽일수록 좁고 구체적 — 우리가 쓰는 **ChatGPT 는 가장 안쪽**, 딥러닝의 한 갈래인 생성형 AI, 그중 **언어 모델(LLM)** 입니다.

AI 발전을 관통하는 한 줄기 — "규칙을 사람이" → "패턴을 데이터로"

🖼️ 이미지 분야에서

- **예전**: "고양이는 귀가 뾰족하고 수염이 있다" — 사람이 **특징을 하나하나** 규칙으로 정의
- **지금**: 라벨 붙은 사진을 **왕창** 넣으면, 모델이 **특징을 스스로** 찾아냄

🗣️ 언어(NLP) 분야에서

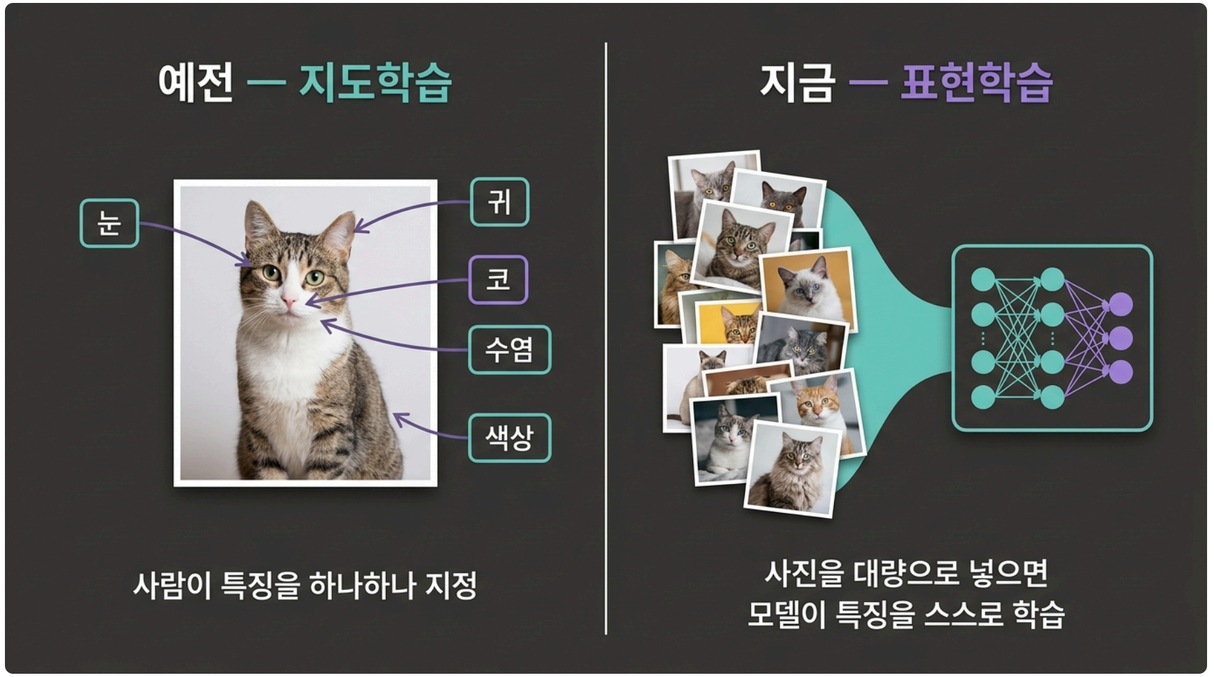
- **예전**: "이런 어미는 높임말" 식으로 **문법 규칙·사전**을 사람이 작성
- **지금**: 인터넷 규모의 텍스트로 **패턴을 스스로** 학습

🔁 같은 전환, 두 분야에서 반복

	사람이 규칙을	데이터로 학습을
누가 한다	전문가가 직접	모델이 스스로
한계	예외를 다 못 적음	데이터·연산 필요
결과물	깨지기 쉬움	일반화 잘됨

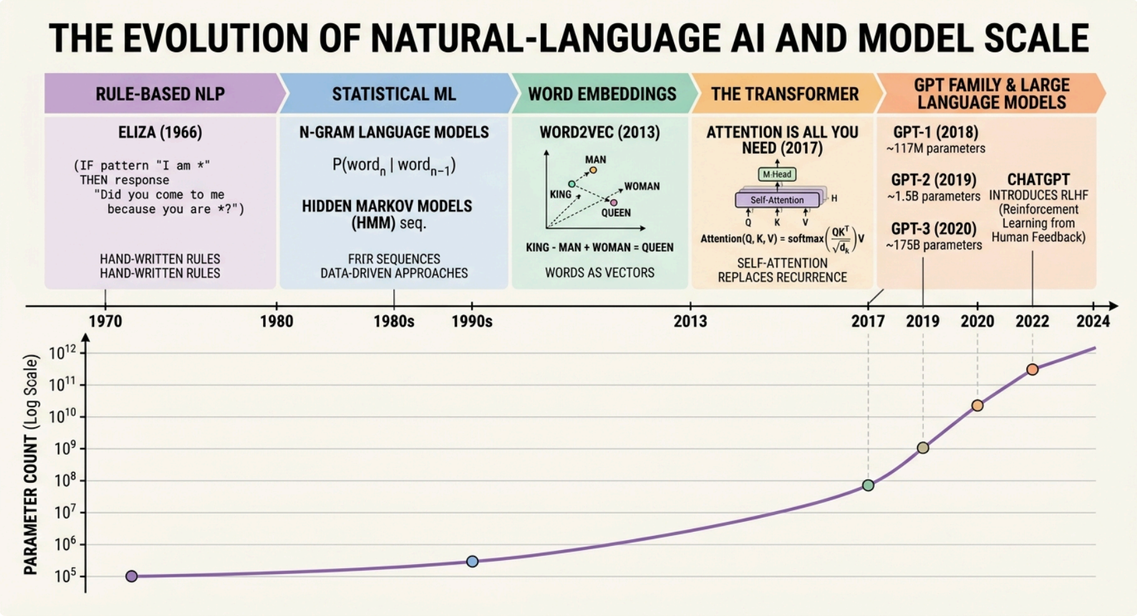
> 🔑 핵심: 이미지든 언어든 **"사람이 일일이 규칙을 적던" 방식이 "데이터에서 스스로 배우는" 방식으로** 넘어왔습니다.
> 이 한 줄기가 Rule-based → ML → DL → LLM 전체를 관통합니다.

고양이를 어떻게 가르치나 — 사람이 특징을 → 데이터로



> 💡 **예전**엔 사람이 '눈·귀·코·수염·색상' 같은 특징을 일일이 알려줬지만(지도학습), **지금**은 고양이 사진을 대량으로 넣으면 모델이 그 특징을 **스스로** 찾아낸다(표현학습). 이 전환이 딥러닝의 핵심이다.

규칙 기반에서 LLM 까지 — 패러다임 발전사



규칙기반 → 통계/ML → 단어 임베딩 → 트랜스포머(2017) → 대규모 LLM · 파라미터 규모의 폭발적 증가

단계	핵심 아이디어	한계
규칙 기반	사람이 규칙을 직접 작성	예외·변형을 모두 담을 수 없음
머신러닝	데이터에서 통계 패턴 학습	특징(feature) 설계를 사람이 해야 함
딥러닝	신경망이 특징까지 학습	긴 문맥·병렬화에 취약(RNN)
트랜스포머	어텐션으로 전체 관계 파악	큰 연산 자원 필요
LLM	대규모 사전학습 + 생성	환각·최신 정보 부재

💡 각 단계는 **앞 단계의 한계를 극복**하며 등장했습니다. 흐름은 "사람이 규칙을 짜던" 방식에서 **"모델이 데이터에서 스스로 배우는"** 방식으로 옮겨 갔습니다.
다음 슬라이드부터 이 5단계를 하나씩 살펴봅니다.

규칙 기반 AI — "사람이 규칙을 다 적는다"

어떻게 동작했나 (1950s~1980s)

- 전문가의 지식을 **if-then 규칙**으로 코딩
- 의료 진단(MYCIN), 세무·법률 **전문가 시스템**
- "감기 AND 38도↑ THEN 독감 의심"

강점

- 동작이 **투명**하고 설명 가능
- 규칙이 맞으면 **정확**

❌ 한계에 부딪힌 이유

- 규칙이 수천 개로 늘면 **유지 불가**
- "고양이 사진"을 if-then 으로 정의 불가능
- **예외**를 사람이 다 못 적음 → 현실 언어·이미지에서 붕괴

> 🔑 핵심 교훈: 세상은 규칙으로 다 적기엔 너무 복잡하다.
> → **"데이터로 규칙을 스스로 찾게 하자"** = 머신러닝의 출발

머신러닝 — "규칙 대신 데이터를 준다"

패러다임 전환

- 사람: **정답이 달린 데이터**를 제공
- 기계: 데이터에서 **패턴(규칙)** 을 스스로 학습
- 예: 스팸메일 1만 통 → 스팸 판별 규칙 자동 도출

대표 기법

- 결정트리 · SVM · 랜덤포레스트
- 통계 기반 자연어 처리(n-gram)

여전히 남은 한계

- **특징(feature)** 을 사람이 직접 설계해야 함
- "이 단어 포함?", "길이?", "대문자 비율?"
- 이미지·언어처럼 **특징이 복잡**하면 사람 설계가 한계

> 🔑 다음 질문: **특징 추출까지 기계가 하게 할 수 없을까?**
> → 딥러닝의 출발

딥러닝 — "특징도 기계가 알아서"

핵심 아이디어

- **여러 층의 신경망(neural network)** 이
- 낮은 층: 단순 특징(선·모서리) → 높은 층: 복잡 특징(얼굴·의미)
- **표현 학습(representation learning)** — feature 를 자동 추출

2010년대에 급부상한 배경

- **빅데이터** (인터넷 규모 데이터)
- **GPU** (대규모 병렬 연산)
- 알고리즘 개선 (ReLU, dropout 등)

성과

- 2012 ImageNet — 이미지 인식 인간 수준 근접
- 음성인식 · 번역 품질 급상승

> 🔑 그런데 **언어**는 여전히 어려웠다.
> "순서가 있는 긴 문장"을 다루는 게 핵심 난제였음 → 다음 슬라이드

PART B

PART B
NLP 의 진화 — 단어에서 문맥으로

기본기(토큰화·NER) → 단어 빈도 → 단어 의미 → 문맥 이해 — 약 16분

NLP — 컴퓨터에게 사람 말을 가르치는 일

두 가지 큰 과제

- **NLU (이해)** — 사람 문장을 컴퓨터가 **읽고 해석**
- 의도 파악 · 감성 분석 · 질의 이해
- **NLG (생성)** — 컴퓨터가 사람처럼 **문장을 작성**
- 번역 · 요약 · 대화 응답

왜 어려운가

- 같은 단어가 **문맥마다 다른 뜻** ("배" 가 과일/선박/신체)
- 어순·생략·중의성 — **규칙으로 다 못 적음**

그래서 먼저 해야 할 '기본기'

단계	하는 일	예
토큰화	문장을 처리 단위로 쪼갬	"안녕하세요" → 토큰들
정규화	표기 통일	대소문자·원형 복원
NER	고유명사 인식	"서울"=지명, "박수현"=인물
벡터화	토큰을 **숫자**로	모델은 숫자만 먹음

> 🔑 모델은 글자가 아니라 **숫자**만 다룹니다. 그래서 모든 NLP 는 **"문장을 토큰으로 쪼개 숫자로 바꾸는"** 전처리에서 출발합니다.

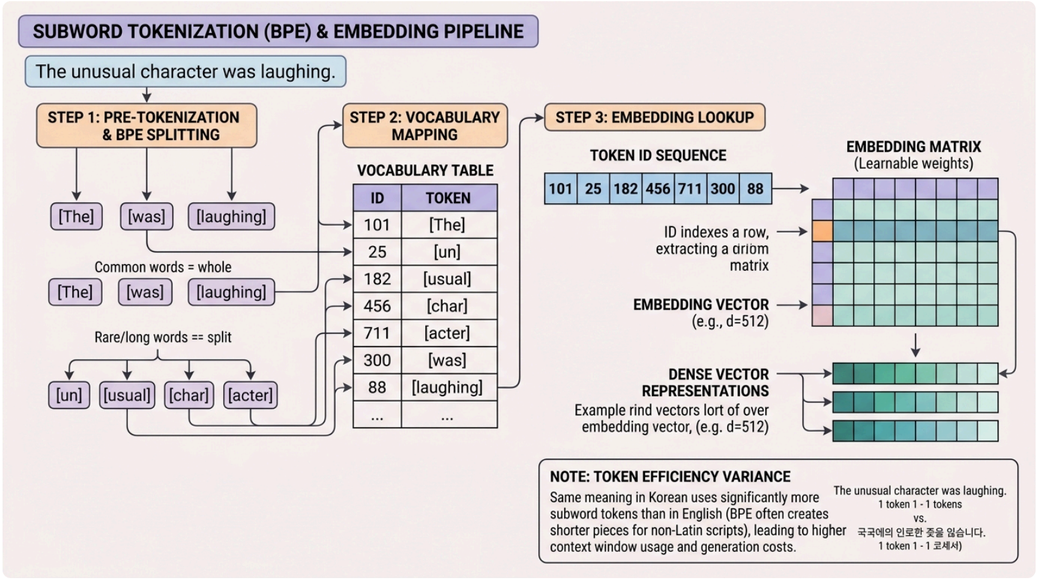
토큰화와 임베딩 — 글자를 숫자로 바꾸는 첫 관문

① 토큰화 (Tokenization)

- 문장을 **토큰** 단위로 분해 (단어보다 작을 수 있음)
- 현대 LLM 은 **서브워드** 방식 (BPE)
- "laughing" → "laugh" + "ing"
- 모르는 단어도 **조각으로 표현** → 어휘 폭발 방지

② 임베딩 (Embedding)

- 각 토큰을 **고정 길이 숫자 벡터**로 변환
- 비슷한 의미의 토큰 → **가까운 벡터**
- 이 벡터가 RAG **의미 검색**의 핵심 재료



문장 → 서브워드 토큰 → 어휘 ID → 임베딩 벡터

💡 "토큰 수" 가 곧 **API 비용·길이 제한**의 단위입니다. Day1 Session 7 에서 임베딩 벡터를 직접 계산하며 다시 만납니다.

토큰화·임베딩 — 한 문장으로 직접 따라가기

① 토큰화 (Tokenization)

질문을 토큰 단위로 분해합니다.

> **"이 물의 온도는 몇도야?"** → `[이] [물] [의] [온도] [는] [몇] [도] [야] [?]`

② 임베딩 (Embedding)

각 토큰을 **의미 벡터**로 바꿉니다. 예를 들어 `물` 토큰은:

> `물` → `[ 0.8, -0.4, 0.6, 0.0, -0.2, … ]`

- 실제로는 **1536차원** — 여기선 앞 5차원만 예시
- 각 차원은 **하나의 "의미축"** 으로 볼 수 있음

`물` 벡터의 각 차원이 말하는 것

차원	값	의미축	해석
1	**+0.8**	자연 (Nature)	강하게 **관련**
2	**−0.4**	고체 (Solid)	**반대** (물=액체)
3	**+0.6**	온도 (Temperature)	속성으로 **관련**
4	**0.0**	감정 (emotion)	**무관**
5	**−0.2**	건조함 (dryness)	**반대** 속성

> 🔑 임베딩 = 단어를 **여러 의미축 위의 좌표**로 놓는 일. 좌표가 가까우면 의미가 비슷 → 이것이 **RAG 의미 검색**의 토대입니다.

자연어 처리, 어떻게 똑똑해졌나

단계	한계	다음으로 넘어간 이유
BoW/TF-IDF	"왕-여왕" 관계 모름, 순서 무시	단어의 **의미**를 담고 싶다
Word2Vec	단어는 고정 — "배(과일/선박)" 구분 못함	**문맥**에 따라 달라져야
RNN/LSTM	순차 처리 → 느림, 장기 의존성 약함	**병렬화 + 긴 문맥**

"King – Man + Woman ≈ Queen"

단어를 숫자 벡터로 (2013)

- 비슷한 맥락에 등장하는 단어 → **비슷한 벡터**
- 단어 사이 **의미·관계가 산술**로 표현됨
- `King - Man + Woman ≈ Queen`

왜 중요한가

- 오늘 우리가 쓸 **임베딩(embedding)** 의 직계 조상
- **RAG 의 핵심**(의미 검색)이 바로 이 아이디어의 확장

여전한 한계: 고정 벡터

- "**배**가 아프다" vs "**배**를 타다"
- Word2Vec 은 둘 다 **같은 벡터** → 문맥 구분 불가

> 🔑 해결책: 같은 단어라도 **주변 문맥에 따라** 벡터가 달라져야 한다
> → Transformer 의 **문맥 임베딩**이 이걸 해냄 (Session 2)

단어 사이 관계가 벡터의 '방향'으로 나타난다

왕 (king) 여왕 (queen) 남자 (man) 여자 (woman)

성별(gender) 방향 → 왕족성(royalty) 방향 ↑

🔑 `왕 - 남자 + 여자 ≈ 여왕` — **'성별' 차이와 '왕족성' 차이가 일정한 방향(벡터)** 으로 나타나기 때문에 단어 관계가 **산술**로 풀립니다. 이 "의미의 방향" 직관이 곧 임베딩·RAG 의미 검색의 토대입니다.

순서는 처리했지만 — 느리고 잊어버린다

RNN → LSTM → GRU 의 기여

- 문장을 **앞에서 뒤로 순서대로** 읽음
- **LSTM·GRU** — 게이트로 장기 기억을 일부 보완
- 번역·음성인식에서 큰 도약 (Seq2Seq)
- **Attention** 보조 장치로 성능↑ (2014~)

❌ 두 가지 결정적 벽

1. **순차 처리** — 단어를 하나씩 → GPU 병렬화 불가 → 학습 느림
2. **장기 의존성** — 문장이 길면 앞부분을 **잊음** - "프랑스에서 자란 그녀는 … 유창하게 ___ 를 한다"

> 🔑 2017년, 누군가 이렇게 물었다:
> **"순서대로 읽지 말고, 문장 전체를 한 번에 보면 안 될까?"**

PART C

PART C
Transformer 혁명 (2017)

> Attention Is All You Need — 약 12분

모든 LLM 의 출발점이 된 논문 한 편

📄 "Attention Is All You Need" (2017)

- **저자**: Vaswani 외 8인 (Google Brain · Google Research)
- **학회**: NeurIPS 2017
- **한 문장**: 순환(RNN)·합성곱(CNN) 없이 **오직 Attention 만으로** 시퀀스를 처리하자

무엇을 제안했나

- **Self-Attention** — 문장 안 모든 단어가 서로를 **동시에** 참조
- 순서대로 읽지 않으니 **완전 병렬화** 가능
- 이 구조에 붙은 이름이 바로 **Transformer**

Attention Is All You Need

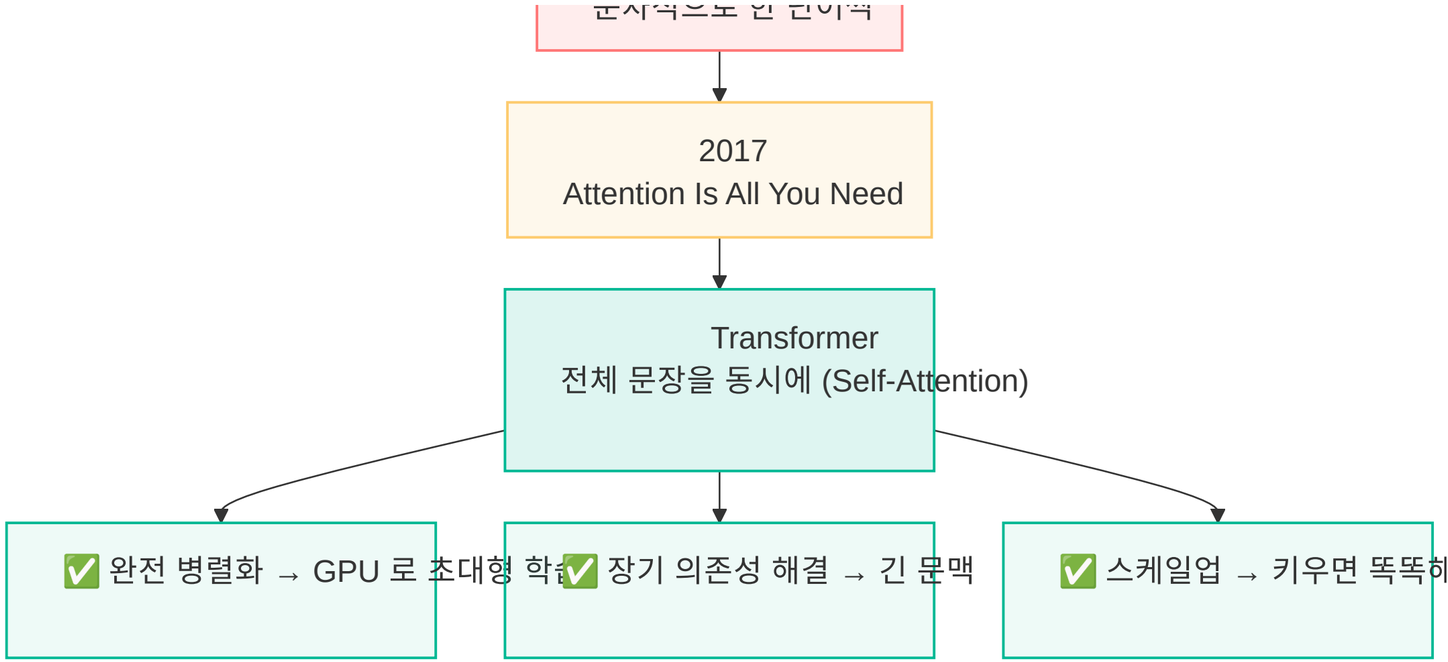
Vaswani 외 8인 · Google Brain / Google Research · NeurIPS 2017

Abstract. 기존의 우수한 시퀀스 변환 모델은 인코더·디코더 구조의 순환(RNN)·합성곱(CNN) 신경망에 기반한다. 본 논문은 순환·합성곱을 완전히 배제하고 **오직 어텐션 메커니즘만**에 기반한 새로운 단순 구조 **Transformer** 를 제안한다. 병렬화가 뛰어나 훨씬 적은 학습 시간으로 더 높은 번역 품질을 달성했다.

2017년 발표된 원 논문의 제목·저자·초록 (재구성)

> 💡 이 한 편이 **GPT · BERT · ChatGPT · Claude — 오늘의 모든 LLM** 의 공통 조상입니다.

"Attention Is All You Need"



> 💡 Google 연구팀의 이 논문 하나가 **GPT·BERT·ChatGPT·Claude — 오늘의 모든 LLM** 의 공통 조상입니다.
> 내부 구조(Self-Attention, Q·K·V)는 **Session 2** 에서 그림으로 분해합니다.

같은 뿌리, 두 갈래로 갈라지다

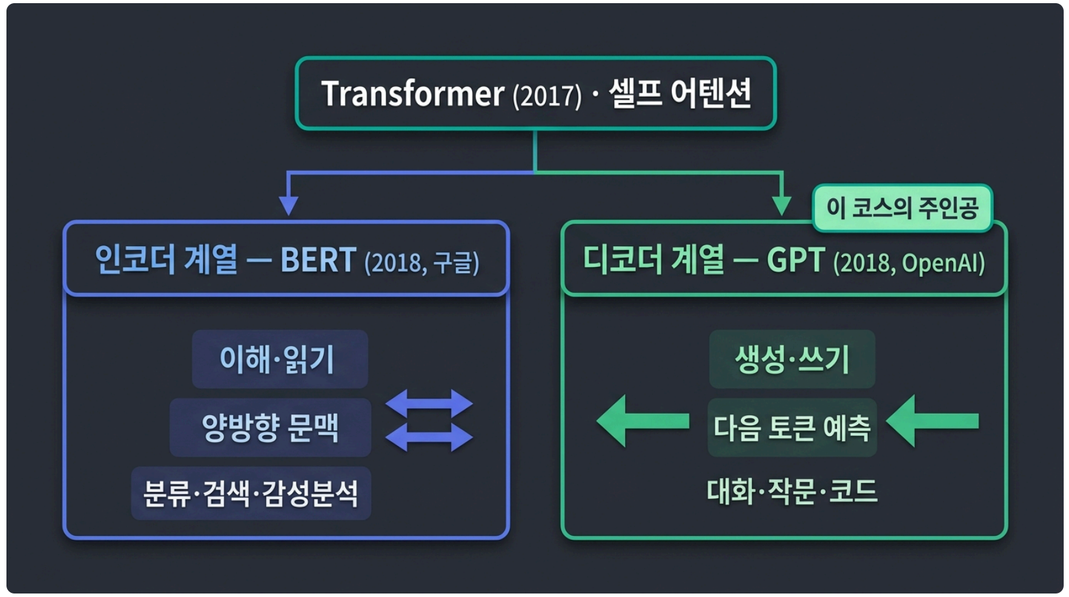
🔵 인코더 계열 — BERT (2018, Google)

- 문장을 **이해(읽기)** 하는 데 최적
- 양방향 문맥 (앞뒤 동시에 봄)
- 분류 · 검색 · 감성분석

🟢 디코더 계열 — GPT (2018, OpenAI)

- 다음 단어를 **생성(쓰기)** 하는 데 최적
- 왼→오 순서로 **다음 토큰 예측**
- 대화 · 작문 · 코드 생성

> 🔑 우리 코스의 주인공은 **GPT 계열(생성형)**.
> "다음 단어 맞히기"를 거대 규모로 했더니 **대화·추론**까지 창발(emergent)했다.



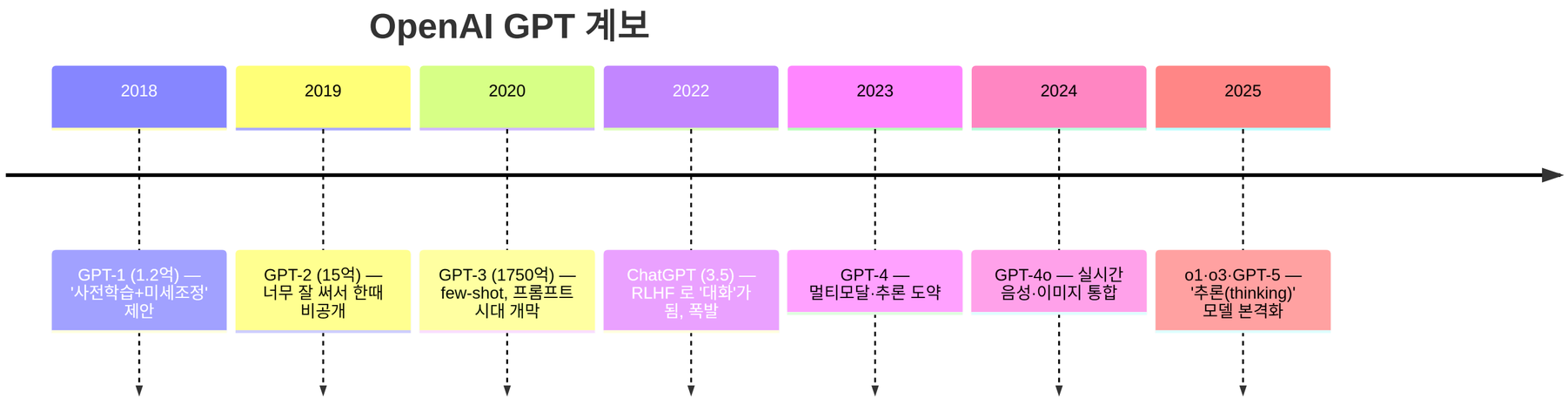
하나의 Transformer 뿌리 → 이해(BERT)·생성(GPT) 두 갈래

PART D

PART D
GPT 와 ChatGPT

계보 · 학습 패러다임 · 변곡점 — 약 12분

GPT-1 에서 GPT-5 까지



💡 핵심 추세 3가지: **① 규모 ↑ (파라미터·데이터)**, **② 정렬 ↑ (사람 의도에 맞춤)**, **③ 양식 ↑ (텍스트→음성·이미지·추론)**.

GPT 는 어떻게 글을 쓰는가 — 다음 토큰 예측

한 문장으로 요약

- 지금까지의 텍스트 다음에 올 **가장 그럴듯한 토큰**을 예측
- **토큰** = 단어보다 작을 수 있는 텍스트 조각
- 매 순간 **어휘 전체에 대한 확률 분포**를 계산 → 하나를 골라 이어붙임

예시

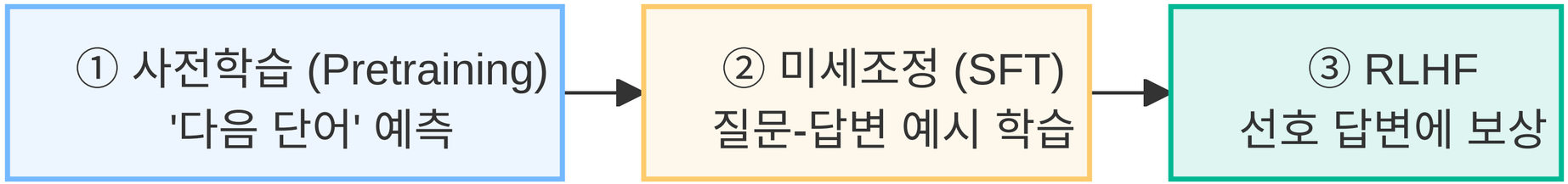
- 입력: `"대한민국의 수도는"`
- 다음 토큰 후보 → **"서울"** 에 높은 확률
- 생성한 토큰을 다시 입력 뒤에 붙여 **반복** → 문장 완성

그래서 생기는 일

- 방대한 텍스트로 이 예측을 반복 학습하면
- 문법·사실 지식·추론의 흔적까지 **파라미터 안에** 자리잡음
- 하지만 **'의미를 이해'** 하는 게 아니라 **통계적으로 그럴듯한** 토큰을 잇는 것

> 🔑 핵심: GPT 는 의미를 이해해서 답을 아는 게 아니라, **통계적으로 가장 그럴듯한 다음 토큰**을 잇습니다.
> 그래서 그럴듯하지만 **사실과 다른 문장**을 자신 있게 만들어 냄 → 이것이 **환각의 근본 원인**.

어떻게 "말귀 알아듣는" 모델이 되나



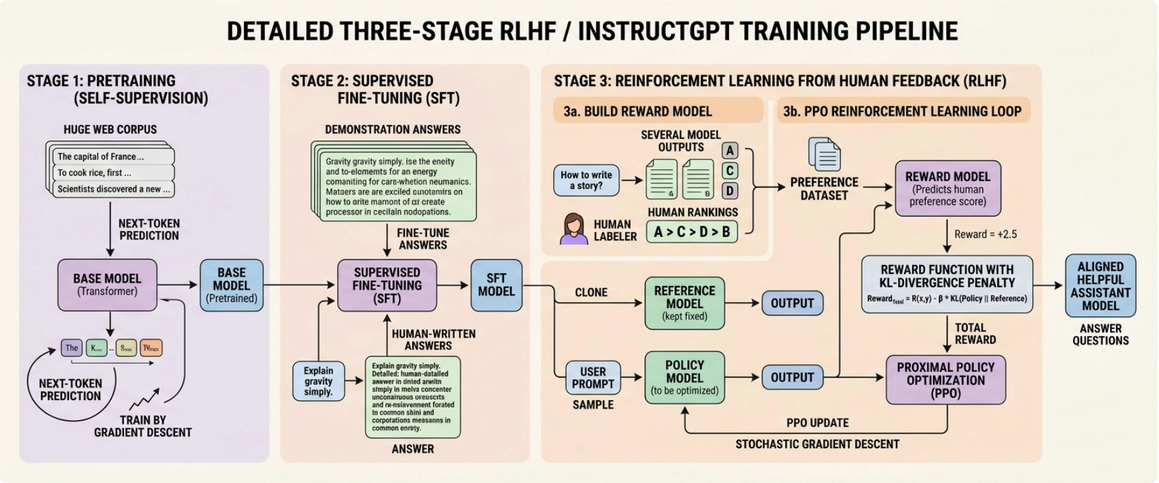
단계	결과물	비유
① **Pretraining**	박식하지만 산만한 모델	책 수억 권 읽은 신입
② **SFT**	지시를 따르는 모델	업무 매뉴얼로 OJT
③ **RLHF**	사람이 **선호**하는 모델	선배 피드백으로 다듬기

> 🔑 GPT-3 → ChatGPT 의 결정적 차이가 바로 ③ **RLHF**. (자세히는 Session 3)

③ RLHF — 사람의 선호로 모델을 다듬는 법

3단계로 쪼개 보면

1. **답변 후보 생성** — 한 질문에 모델이 **여러 답**을 만듦
2. **사람이 순위 매김** — 사람이 "A 가 B 보다 낫다" 식으로 **선호 비교**
3. **보상 모델 학습** — 이 선호 데이터로 *좋은 답에 높은 점수*를 주는 **보상 모델** 을 학습
4. **강화학습으로 정렬** — 모델이 **보상이 높아지는 방향**으로 답을 조정 (PPO)



Pretraining → SFT → RLHF(보상 모델 + 강화학습 루프)

> 💡 핵심 직관: 모델에게 **"정답"을 일일이 주는 게 아니라**, 사람이 **"이게 더 낫다"** 고 고르기만 하면 됩니다. 이 *선호 신호*가 무뚝뚝하던 GPT-3 를 **공손하고 도움 되는 ChatGPT** 로 바꿨습니다. (메커니즘 상세는 Session 3)

ChatGPT 가 바꾼 것

기술이 아니라 "경험"의 혁신

- 모델(GPT-3.5)은 새롭지 않았음
- **대화 인터페이스 + RLHF** 조합이 핵심
- 누구나 **자연어로** AI 를 쓰게 됨

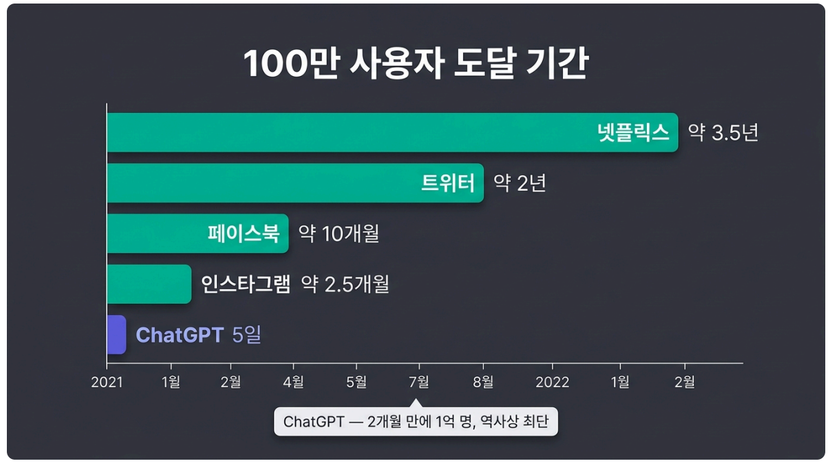
개발자에게 의미

- AI 가 **연구실 → 제품**으로 내려옴
- **API 한 줄**로 누구나 LLM 탑재
- 새로운 직군: **AI 앱 개발자 (= 여러분)**

> 🎯 이 코스의 존재 이유: ChatGPT 이후, **LLM 을 내 서비스에 붙이는 능력**이 핵심 역량이 됨.

📈 숫자로 본 충격

- 출시 **5일** 만에 100만 사용자
- **2개월** 만에 1억 — 역사상 최단



무엇이 근본적으로 달라졌나

항목	기존 NLP (작업별 모델)	LLM (범용 모델)
모델	작업마다 따로 학습	**하나로** 번역·요약·코딩·QA
데이터	작업별 라벨링 대량 필요	**프롬프트** 몇 줄로 지시
적용	ML 엔지니어가 모델 학습	**앱 개발자**가 API 호출
진입장벽	높음 (모델·GPU·데이터)	낮음 (**API 키 + 코드**)
우리 작업	모델 만들기	**모델 활용 + RAG + Agent**

> 💡 그래서 이 코스는 "모델 학습"이 아니라 **"모델 활용·연결·서비스화"** 에 집중합니다.

2026년, 누가 무엇을 만드나

🏢 폐쇄형 (API)

- **OpenAI** — GPT-5 · o 시리즈(추론)
- **Anthropic** — Claude (Opus·Sonnet·Haiku)
- **Google** — Gemini

🌐 오픈형 (가중치 공개)

- **Meta** — Llama
- **Mistral** · **Qwen** · 한국어 특화 모델
- → Day 4 에서 **Ollama 로컬 실행** 체험

본 코스에서 쓰는 모델

용도	모델
메인 실습	`gpt-4o-mini`
임베딩	`text-embedding-3-small`
Day4 비교	`claude-haiku`, 로컬 LLM

> 💡 API 만 바꾸면 모델 교체가 자유롭다 — **LangChain** 이 이 추상화를 담당 (Day 2).

그래서 '생성형 AI' 란 무엇인가

🔎 판별 모델 (Discriminative)

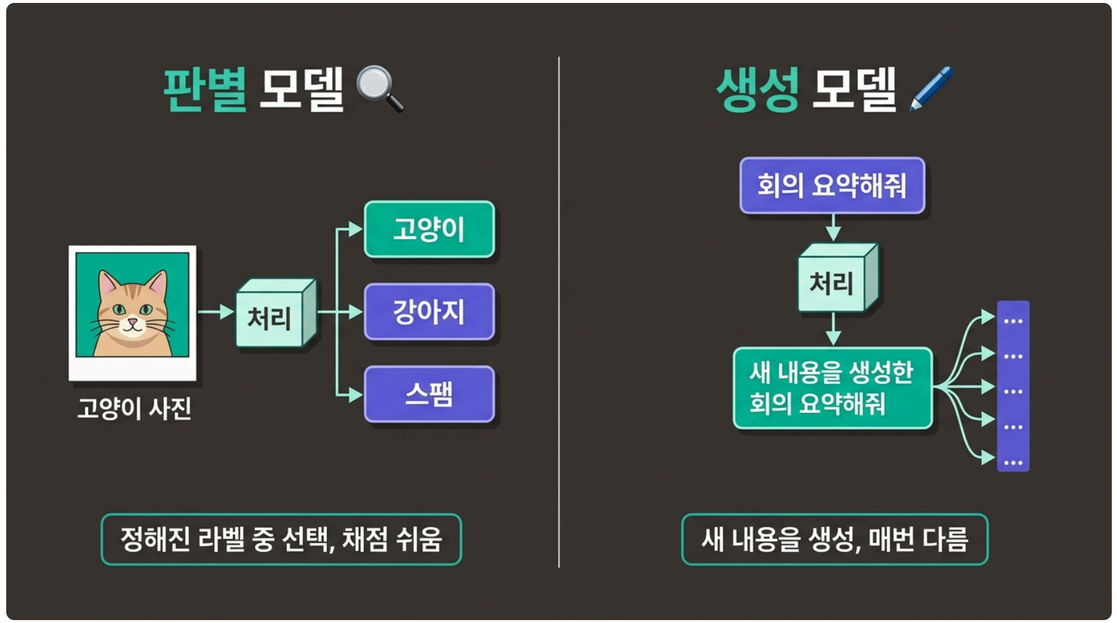
- 입력을 **미리 정의된 범주**로 분류
- 예: "스팸인가?" · "고양이인가?"
- 출력이 **정해져 있음** (라벨 집합)
- 정답과 비교 → **채점이 쉬움**

✍️ 생성 모델 (Generative)

- 학습한 패턴으로 **새 콘텐츠 생성**
- 예: "회의 요약해줘" · "코드 작성해줘"
- 출력이 **매우 다양** (열린 가능성)
- 같은 질문에도 **매번 다른 답**

> 🔑 핵심: 여기까지의 발전사가 도달한 지점 — 생성형 AI 는 '정답을 고르는' 게 아니라 **'그럴듯한 다음 내용을 만들어 내는'** 모델.
> 이 차이가 **환각(hallucination)** 이라는 약점과 **RAG** 라는 보완책으로 이어집니다 (Day1 Part2).

판별 vs 생성 — 한 그림으로



💡 **왼쪽(판별)** 은 고양이 사진을 받아 *정해진 라벨 중 하나*(고양이/강아지/스팸)를 고릅니다 — 답이 닫혀 있어 채점이 쉽습니다.
오른쪽(생성) 은 "회의 요약해줘" 같은 지시에 *매번 새로운 문장*을 만들어 냅니다 — 답이 열려 있어 같은 질문에도 결과가 달라집니다.

핵심 5가지

1. ✅ 생성형 AI = 판별이 아니라 **'그럴듯한 다음 내용을 생성'** — ChatGPT 는 **AI ⊃ ML ⊃ DL ⊃ 생성형 AI** 의 가장 안쪽
2. ✅ 발전사: Rule-based → ML → DL → Transformer → LLM, **"규칙을 데이터로, 특징까지 자동"** 으로 이동
3. ✅ NLP 는 **단어 빈도 → 의미(Word2Vec) → 문맥(Transformer)** 으로 진화
4. ✅ **2017 Transformer** — 병렬화·긴 문맥·스케일업 → 모든 LLM 의 조상
5. ✅ GPT = **다음 토큰 예측** — 의미 이해가 아니라 통계적으로 그럴듯한 토큰을 이음 → **환각의 근본 원인** → **RAG 로 보완**