DAY 01–10 · 1부 AI가 뭔지

1부 총정리 — AI가 뭔지

용어를 정의하고 그대로 쓰는 판 · 원리와 배경까지

시험 없음 — 읽고 넘어가는 복습 자료

한 조각씩 고르고, 붙이고, 또 고른다

1. AI는 문장을 통째로 지어내지 않아. 다음에 올 조각 하나만 고르고, 그걸 붙인 다음 또 고른다.

2. 이 고르기를 수백 번 반복한 결과가 우리가 읽는 긴 답이야.

3. 그래서 AI는 어딘가에 있는 답을 찾아오는 게 아니라, 매번 새로 만들어 내는 거야.

왜 하필 한 조각씩일까

1. 학습할 때 시킨 일이 딱 그거였기 때문이야 — 문장 일부를 가리고 다음에 올 조각을 맞히기.

2. 잘 맞히는 법을 배운 기계는, 답할 때도 배운 그 방식으로 움직인다.

3. 답이 한 글자씩 흘러나오는 것처럼 보이는 것도 이 때문이야. 실제로 그렇게 만들어지고 있으니까.

이 방식의 이름과 약점

1. 이런 생성 방식을 자기회귀(autoregressive)라고 불러 — 자기가 방금 내놓은 조각을 다시 입력에 넣고 다음 조각을 고른다는 뜻이야.

2. 약점은 되돌아가지 않는다는 점이야. 앞에서 한 번 이상한 조각을 고르면 그 위에 계속 쌓는다.

3. 오늘날 언어 모델 대부분이 이 구조 위에 서 있어. 바뀌어 온 건 조각을 고르는 방법이지, 고른다는 사실 자체가 아니야.

그 조각의 이름, 토큰

1. AI가 고르는 그 조각에 이름이 있어. 토큰이라고 불러.

2. 토큰은 글자도 단어도 아니야 — 자주 붙어 다니는 덩어리는 통째로, 낯선 덩어리는 잘게 자른 조각이야.

3. '오늘 급식 뭐야?'가 사람 눈엔 세 어절이지만, AI 눈엔 다섯 조각쯤 된다.

토큰 수는 곧 시간이고 돈이다

1. 자르는 규칙을 토크나이저라고 해. 모델마다 규칙이 달라서 같은 문장도 토큰 수가 다르게 세어져.

2. 토큰이 많아지면 계산할 것도 그만큼 늘어나 — 느려지고 비싸진다.

3. 한국어는 영어보다 같은 뜻을 담는 데 토큰이 더 많이 드는 경우가 흔해.

자르는 규칙은 어떻게 만들까

1. 널리 쓰는 방식이 BPE(Byte Pair Encoding)야 — 자주 같이 나오는 두 조각을 하나로 합치는 걸 반복해 사전을 만든다.

2. 그래서 처음 보는 단어도 잘게 쪼개면 표현할 수 있어. 사전에 없는 말이라고 막히지 않는 이유야.

3. 뒤에 나올 '컨텍스트 길이'도 글자 수가 아니라 토큰 수로 센다 — 이 조각이 모든 계산의 단위야.

확률표와 온도 손잡이

1. 다음 조각을 고를 때 AI는 후보마다 점수를 매기고, 그 점수대로 제비를 뽑아.

2. 온도는 그 제비뽑기를 얼마나 모험적으로 할지 정하는 손잡이야.

3. 낮추면 1등만 계속 뽑히고, 올리면 아래 후보도 자주 튀어나온다.

언제 낮추고 언제 올릴까

1. 같은 질문에 매번 다른 답이 나오는 건 기분이 달라서가 아니라 제비뽑기라서야.

2. 정답이 하나인 일(계산·요약·코드)에는 낮추는 게 낫고, 아이디어를 여러 개 받고 싶을 땐 올리는 게 나아.

3. 온도를 올린다고 똑똑해지지 않아 — 다양해질 뿐이고, 엉뚱해질 위험도 같이 커진다.

온도는 계산의 어디에 끼어드나

1. 점수를 확률로 바꾸는 계산을 소프트맥스라 하고, 온도는 그 직전에 점수를 나누는 숫자야. 나누는 값이 크면 점수 차가 줄어 확률표가 평평해진다.

2. 온도를 0에 가깝게 두면 사실상 1등만 고르는 방식이 돼.

3. 실제로는 상위 몇 개만 남기는 top-k, 확률 합이 일정 수준이 될 때까지만 남기는 top-p 같은 손잡이를 함께 쓴다.

학습은 조금씩 고치기의 반복

1. AI의 학습은 깨달음이 아니라 '조금씩 고치기'의 반복이야.
2. 문장 일부를 가리고 다음 조각을 맞혀 보게 한 다음, 틀린 만큼 속의 숫자를 살짝 고쳐.
3. 이걸 어마어마하게 반복하면 문법도 상식도 그 숫자들 안에 녹아든다.

숙제 채점에 가까운 일

1. 틀린 문제마다 '이만큼 틀렸다'는 점수를 내고, 그 방향으로 조금 움직이는 것에 가까워.

2. 한 번에 크게 고치지 않아. 조금씩, 대신 아주 여러 번.

3. 그래서 학습에는 엄청난 시간과 전기가 들고, 한 번 끝나면 그 뒤로 숫자는 고정된다.

손실과 역전파, 그리고 고정된 숫자

1. 틀린 정도를 재는 값을 손실(loss)이라 하고, 그 값을 줄이는 방향을 거꾸로 계산해 숫자를 고치는 방법이 역전파(backpropagation)야 — 1986년에 정리된 방법이지.

2. 학습이 끝난 뒤에는 우리와 대화한다고 모델이 배우지 않아. 그 숫자는 그대로다.

3. '어제 알려준 걸 기억해?'가 안 통하는 이유가 여기 있어. 기억처럼 보이는 건 7일차에 본 다른 장치야.

고쳐지는 숫자의 개수, 파라미터

1. 학습으로 고쳐지는 그 숫자들의 개수가 파라미터야.

2. 많을수록 담을 수 있는 패턴의 그릇이 커진다.

3. 요즘 큰 모델은 수백억에서 1조 개 단위야 — 세어 볼 수 있는 숫자가 아니야.

이 숫자가 얼마나 큰가

1. 1초에 하나씩 센다면 1천억 개를 세는 데 3천 년이 넘게 걸려.

2. 사람 뇌의 시냅스가 약 100조 개니까, 아직 거기엔 못 미친다.

3. 그런데 숫자가 크다고 다 똑똑한 건 아니야 — 무엇을 얼마나 학습했는지가 같이 따라와야 해.

크기 경쟁에서 균형 이야기로

1. 2020년 GPT-3 논문이 1,750억 개라는 숫자로 이 경쟁의 기준점을 만들었어.

2. 2022년 친칠라 연구는 모델만 키우기보다 학습 데이터 양과 균형을 맞추는 쪽이 낫다는 걸 보여줬지.

3. 그래서 지금은 '몇 개냐'보다 '어떻게 학습했느냐'를 더 많이 이야기한다.

자신 있게 지어내는 일 — 환각

1. AI가 없는 사실을 태연하게 지어내는 걸 환각이라고 불러.

2. 거짓말하려는 게 아니야. 그럴듯한 다음 조각을 고르는 구조에서 당연히 따라오는 일이야.

3. AI는 '나는 이걸 모른다'를 표시하는 장치를 따로 갖고 있지 않아.

특히 잘 틀리는 자리

1. 학습 데이터에 드물었던 것, 아주 최근 일, 그리고 숫자·날짜·이름 같은 구체적인 사실에서 잘 틀려.

2. 말투는 아는 것과 모르는 것이 똑같아. 자신 있어 보인다고 사실인 게 아니야.

3. 출처를 물어보고 그 출처를 직접 열어보는 습관이 가장 싸고 확실한 대비야.

왜 못 없애고 줄이기만 하나

1. 2023년 정리 논문은 환각을 '입력과 어긋나는 것'과 '세상 사실과 어긋나는 것'으로 나눠 본다.

2. 완전히 없애는 방법은 아직 없어. 검색을 붙이거나, 출처를 강제하거나, 확신도를 따로 재는 식으로 줄이는 중이야.

3. 그러니 확인할 수 있는 형태로 묻는 것 — 그게 쓰는 사람 쪽의 기술이야.

기억이 아니라 다시 읽기

1. AI는 대화를 저장해 두지 않아. 매번 지금까지의 대화를 통째로 다시 읽고 답해.

2. 한 번에 읽을 수 있는 분량에 한계가 있는데, 이걸 컨텍스트 길이라고 불러.

3. 대화가 길어지면 앞부분이 창밖으로 밀려난다.

길어도 문제, 가운데도 문제

1. '아까 말했잖아'가 안 통할 때가 있어. 그 말이 이미 창밖으로 나갔을 수 있거든.

2. 길이가 남아 있어도 문제야 — 가운데 있는 내용이 앞뒤보다 덜 반영되는 경향이 확인됐어.

3. 정말 중요한 정보는 대화 뒷부분에서 한 번 더 짚어주는 게 가장 확실해.

창의 단위는 토큰이다

1. 이 창의 단위는 글자가 아니라 2일차에 본 토큰이야.

2. 2023년 'Lost in the Middle' 연구가 긴 입력의 중간 정보가 덜 쓰인다는 걸 보여줬어.

3. 창이 아무리 길어져도 읽는 값은 공짜가 아니야 — 길수록 느리고 비싸진다.

질문을 바꾸면 확률표가 바뀐다

1. 프롬프트는 AI에게 건네는 입력 문장 전체야.

2. 문장이 달라지면 다음 조각의 확률표 자체가 달라져 — 그래서 답이 확 바뀐다.

3. AI를 바꾸는 게 아니라 질문을 바꾸는 거야.

잘 통하는 세 가지

1. 역할을 주고, 조건을 적고, 원하는 형식을 말하기 — 이 셋이 가장 잘 통해.

2. '좋은 글 써줘'보다 '중학생이 읽을 300자 안내문, 예시 한 개 포함'이 훨씬 쓸모 있는 답을 끌어낸다.

3. 속이는 기술이 아니야. 내가 원하는 걸 더 또렷하게 말하는 것뿐이야.

프롬프트가 할 수 있는 일과 없는 일

1. 2021년 조사 논문은 프롬프트를 '모델을 다시 학습시키지 않고 원하는 동작을 끌어내는 인터페이스'로 정리했어.

2. 단계를 나눠 풀게 하는 것처럼, 질문 방식만 바꿔도 정확도가 달라지는 경우가 있어.

3. 다만 프롬프트로 6일차의 환각을 없앨 수는 없어 — 줄일 뿐이야.

크기는 파라미터 수다

1. 모델이 크다는 건 5일차에 본 파라미터가 많다는 뜻이야.

2. 큰 모델은 아는 게 많고 어려운 문제를 버티지만, 느리고 비싸고 서버에서만 돈다.

3. 작은 모델은 내 기기 안에 들어가서 빠르고, 인터넷이 없어도 돌아.

어느 일을 어느 쪽에 맡길까

1. 급하고 자주 하는 가벼운 일은 작은 쪽, 오래 생각하고 많이 읽어야 하는 일은 큰 쪽.

2. 큰 모델이 만든 답을 교재 삼아 작은 모델을 가르치는 방법도 있어.

3. 그러면 작은 모델이 제 덩치보다 잘하게 되지만, 큰 모델을 그대로 대신하진 못해.

증류 — 배운 걸 옮겨 담기

1. 이 방법을 증류(distillation)라고 부르고, 2015년 힌턴 팀이 정식화했어.

2. 지금 폰 안에서 도는 AI 상당수가 이 길을 거쳐 왔다.

3. 우열이 아니라 배치의 문제야 — 좋은 크기는 쓸 자리가 정한다.

알맹이가 어디에 있느냐

1. 닫힌 모델은 알맹이가 회사 서버 안에만 있어. 우리는 창구로 말만 걸 수 있어.

2. 열린 모델은 그 알맹이, 곧 가중치를 통째로 내려받아 내 컴퓨터에서 돌릴 수 있어.

3. 묻는 자리는 같아도, 모델이 사는 자리가 다른 거야.

'열렸다'를 네 조각으로

1. '열렸다'는 한 덩어리가 아니야. 가중치·돌리는 코드·학습 데이터·고쳐 쓸 권리, 네 조각으로 쪼개 봐야 해.

2. 가중치와 코드는 열어도 학습 데이터는 안 여는 쪽이 대부분이야.

3. 그래서 열렸다는 말만 듣지 말고, 어느 조각이 열렸는지를 따져 물어야 한다.

라이선스가 가르는 선

1. 2023년 라마 2는 상업 이용까지 허용했지만 회사가 만든 자체 라이선스를 붙였고, 그래서 '이게 오픈소스냐'는 논쟁이 시작됐어.

2. 같은 해 미스트랄 7B는 널리 쓰는 아파치 2.0을 그대로 붙여 제약을 거의 없앴지.

3. 열렸다는 건 공짜라는 뜻이 아니야 — 돌릴 기계도, 업데이트도 받은 사람 몫이다.

열흘은 하나의 이야기였다.
다음 조각을 고르는 기계.



2부에서는 이 물건이 어디서 왔는지 — 1958년, 스스로 배우는 인공 뉴런부터 거슬러 올라간다.