

LLM-Wiki: □□□□□□□□□□ □□□□□□□□□□□□□□ AI □□□□□□□ □□□□□□□ □□□□□□□□□□□□

Contents

LLM-Wiki: 지식이 축적되는 AI 작업 환경 만들기	12
들어가며	12
이 책의 구성	12
읽는 방법	12
장별 구성	13
이 책이 약속하지 않는 것과 사람의 몫	14
서문	14
1장. LLM-Wiki란 무엇인가	15
대화가 남지 않는 이유	16
LLM-Wiki의 한 문장 정의	16
LLM-Wiki를 시작하게 된 계기	16
일반적인 메모, 검색, 채팅과의 차이	17
LLM-Wiki를 이루는 다섯 층	18
사람이 말을 일과 AI가 말을 일	18
사람마다 달라지는 쓰임새	18
계속 갱신되는 순환	18
2장. AI 에이전트와 작업 폴더	19
챗봇에서 에이전트로	19
Markdown이라는 형식	20
폴더가 작업 공간이라는 뜻	20
웹 채팅과 폴더 기반 도구의 차이	21
AI 에이전트 도구 고르기	21
Codex, Claude Code, ChatGPT가 속한 카테고리	22
같은 회사의 두 도구를 고르는 기준	22

여러 대의 컴퓨터에서 같은 위키를 쓰는 방법	22
폴더를 정하고 새 세션을 여는 법	23
Codex에서 첫 세션 만들기	23
Claude에서 첫 세션 만들기	23
3장. Karpathy의 제안에서 나의 연구 위키까지	26
매번 다시 계산하는 검색	26
Karpathy 원형의 세 층	27
연구 논문용 네 층 확장	28
연구자용 LLM-Wiki의 아홉 가지 운영 원칙	28
답의 근거를 지키는 네 가지 원칙	28
위키의 품질을 지키는 다섯 가지 원칙	29
설치 전에 정할 질문과 카테고리	30
Gist로 폴더를 세우는 과정	31
따라해보기	31
설치 직후에 확인할 것	33
4장. 논문 한 편을 LLM-Wiki에 넣는 법	33
한 편의 ingest가 만드는 것	33
Step 0. PDF의 정체성과 완전성 확인	33
Step 1. 원문 보존과 텍스트 추출	34
Step 2. Source note 작성	35
Step 3. 카테고리 분류	36
Step 4. Paper wiki page 작성	36
Step 5. Synthesis layer에 연결	37
Step 6. 검증, 로그, 색인	38
5장. 논문 PDF를 ingest 하는 방법	39
원본 PDF를 고르는 기준	39
추출이 어려운 이유	39
추출기를 고르는 두 갈래	41
따라해보기	41
조용히 성공하는 실패	42
추출 결과를 다듬는 범위	42
추출 도구 기록	42

PDF에서 정보를 다시 뽑아야 하는 시점	43
6장. 논문 PDF 여러 개를 ingest하기	43
한 편과 여러 편 사이에서 달라지는 것	45
이미 가지고 있는 논문에서 시작하기	45
배치를 열기 전에 확정하는 목록	46
따라해보기	46
중복과 제외 대상을 거르는 두 겹의 검사	47
조용히 실패하는 추출	47
배치에서도 건너뛰지 않는 종합 연결	47
분모와 분자를 함께 적는 완료 보고	48
배치를 달고 색인을 다시 만드는 순서	48
배치가 위키의 구조로 굳는 현상	48
따라해보기	49
배치가 남기는 부채	49
7장. 논문 PDF를 확보하는 방법	50
서지 정보와 원문을 나누어 생각하기	50
OpenAlex로 후보를 모으기	50
따라해보기	50
PubMed와 프리프린트 서버	52
관심 저널을 정하고 예약으로 돌리기	52
기준을 정해 거르기	52
인용 목록에서 거슬러 올라가기	52
브라우저로 내려받기	53
Paywall 논문의 처리	53
확보 기록	54
8장. Supplementary Table과 Document를 ingest 하는 방법	54
지식 층과 자원 층의 구분	54
표가 논문의 본체인 경우	55
보충 자료를 자동으로 모으는 방법	55
따라해보기	55
파일 이름이 알려 주지 않는 것과 manifest	55

Reporting Summary에서 살릴 것과 버릴 것	56
보관하지 않는 문서	56
압축 파일 안에 들어 있는 것	56
남의 파일과 내가 만든 파일	57
보충 자료가 있는지 미리 아는 법	57
표를 분석 입력으로 쓰기	57
9장. 질문이 지식으로 축적되는 법	58
질문을 받으면 LLM-Wiki 안에서 일어나는 일	58
LLM-Wiki에 질문했을 때 위키 파일을 검색하는 방법	58
완전한 질문으로 검색하는 이유	59
검색 후보에서 근거로	60
여러 논문을 연결하는 Deep Synthesis	61
답을 위키로 되돌리기	62
Index를 최신으로 유지하는 법	63
10장. 논문의 구조를 해부하기	64
결과 요약과 구조 분석의 차이	64
인용이 맡는 여덟 가지 기능	65
원문과 인용의 연결	65
인용문 읽기와 원문 읽기의 차이	66
Section별 인용 지도	66
Intellectual core와 provenance tail의 분리	67
인용 관계의 도식화	68
위키 coverage의 교차 점검	68
같은 축으로 다섯 편을 재면	69
11장. 위키에서 연구 산출물로	69
위키에서 만들 수 있는 산출물	70
질문에서 연구 아젠다로	70
위키에서 원고 구조로	72
인터랙티브 만들기	72
산출물과 정보의 관계	73

세 가지 미니 프로젝트	73
산출물에서 위키로 되돌아오는 것	74
12장. Obsidian으로 보기와 채팅으로 묻기	74
폴더를 여는 도구	75
그래프가 보여 주는 것	75
화면에서 눈에 걸리는 것들	75
그래프가 보여 주지 못하는 것	76
다른 도구를 고를 때	76
채팅 쪽을 주로 쓰는 이유	76
Obsidian과 채팅 나누기	77
화면을 여는 일이 목적이 될 때	77
13장. 에이전트에게 주는 권한의 경계	77
권한을 물어보는 세 가지 상황	78
두 도구의 권한 모드	78
Claude Code의 권한 모드	79
Codex의 승인 정책과 샌드박스	79
따라해보기	80
확인 창에서 볼 것	80
도구가 막는 것과 규칙이 막는 것	80
되돌릴 수 있는가로 나누는 경계	81
도구 단위 허용 목록	81
자료에 섞여 들어온 지시문	81
설정이 흔들리는 원인	81
처음 시작할 때 권한하는 조합	82
14장. 말로 지시하고 맥락을 넘기는 법	82
타이핑하지 않게 되는 맥락	82
긴 수다 세션	83
음성으로 넣는 실제 방식	84
내가 Codex를 권한하는 또 하나의 이유	84
따라해보기	84
인터뷰로 바꾸기	85

되돌아온 정리 확인	85
수다가 이미 있는 문서를 만날 때	85
어느 작업 공간으로 갈지 정하기	85
작업 공간마다 다른 위키	86
음성 입력에서 남길 것	86
15장. 논문이란 무엇인가 — 분야의 대화에 들어가는 법	87
논문이라는 이야기의 구조	87
영어보다 먼저 필요한 맥락	87
시리즈 안에서 한 편이 갖는 의미	88
과학적 리더십의 형성	88
Review-first 온보딩과 첫 지도의 선택	89
첫 review를 LLM-Wiki에 넣고 대화하기	89
Review에서 원저 논문으로 내려가는 반복 순환	90
온보딩의 산출물인 한 장짜리 field map	91
16장. 상용 AI에 넣으면 안 되는 자료	92
전송이 곧 반출인 경우	92
식별 가능한 자료의 범위	93
위키를 둘로 나누는 경계	93
폐쇄망에서 도는 모델	93
로컬 모델로 충분한 일과 아쉬운 일	94
따라해보기	94
경계에서 새는 경로	94
회의 녹음과 전사	95
공개 데이터를 쓸 때도 남는 조건	95
학생과 정하는 자료 경계	95
나가도 되는 것을 미리 정하기	95
17장. 데이터 분석을 위한 폴더 구성	96
방법을 다루는 세 종류의 논문	96
방법을 꼬집어내는 열한 가지 질문	97
Concept page와 method page의 구분	98

실험 설계의 큰 그림	98
분석 설계와 실험 설계의 동형성	99
따라해보기	100
계획서가 만들어지는 순환	101
실험과 계산 분석의 갈림	101
계획에서 미팅 자료로의 전환	102
18장. 공동 연구 미팅을 준비하는 법	102
미팅의 기능	103
미팅 전에 정하는 결정 항목	103
LLM-Wiki로 pre-read를 만드는 법	104
같은 페이지의 표준 구조	104
질문별로 답할 수 있음과 답할 수 없음을 나누기	105
미팅 자료에 담기는 연구자의 판단	105
미팅 중 사실과 해석과 결정의 분리	106
회의 기록을 전사에서 노트로 바꾸는 경로	107
따라해보기	108
회의 노트를 공유 문서로 옮기기	108
미팅 후의 후속 행동 목록	108
세 가지 사례	109
19장. 논문 원고 폴더 구성하기	110
주장 체계로서의 논문	110
중심 주장에서 필요한 근거를 역설계하기	110
서론의 네 단계	111
결과의 배열 순서	112
그림 서사를 사람 손으로 검수하기	112
LLM-Wiki로 주장의 문헌 근거를 검증하기	113
고찰의 네 부분	113
논리 단위로서의 문단	114
스스로 원고를 반박해 보기	114
논문 작성의 전체 순환	115
따라해보기	116

20장. 새로운 주제 온보딩 하기	116
고른 논문을 건네는 이유	116
묵음에 함께 담는 것	117
학생마다 달라지는 묵음	117
따라해보기	117
여러 학생에게 반복해서 만들 때	118
잘라 낼 때 끊어지는 링크	118
밖으로 나가면 안 되는 것	118
묵음을 받은 학생이 첫 주에 하는 일	118
학생의 문서가 연구실 위키로 돌아오는 경로	119
묵음이 낡는 속도	119
21장. 에이전트가 대신해 주지 않는 것, 당신의 과학	120
요약을 읽는 것과 아는 것	120
줄지 않고 옮겨 가는 시간	120
자기 말로 설명하기	120
따라해보기	121
사람을 만나는 훈련	121
발표가 만드는 것	121
하루 30분의 글쓰기	122
학습 시간의 배분	122
도구를 쓰는 것과 의존하는 것	122
위키가 학습에 기여하는 조건	122
자기가 만든 지식에 대한 책임	123
자동화되는 쪽과 남는 쪽	123
22장. MCP 기본 개념	123
규약이 연결을 도구에서 떼어 놓는 방식	123
연결하기 전에 정하는 경계	125
메일과 일정	125
팀 대화	126
프로젝트 문서	126

HWP 양식을 다루는 연결	126
따라해보기	126
확인하는 순서를 지침 파일에 적기	127
연결을 고르고 검증하는 일	127
연결이 늘어날 때 생기는 문제	127
23장. 도반(道伴), 당신의 공동연구자를 지도하는 법	128
지도교수는 토요일 아침이 두렵다	128
학생별 기록의 여섯 층	130
개인별 기록에 남기는 것	131
관찰과 해석과 지도 행동의 분리	131
미팅 전 현재 상태의 복원	132
미팅 후에 남길 것과 남기지 않을 것	133
따라해보기	134
원고를 함께 쓰는 순환	134
학생마다 다른 순환의 설계	135
합류에서 독립 연구자로	135
팀의 기억과 개인의 존엄	136
여러 작업 공간의 연결	136
24장. Meetings 폴더로 PI의 외부 활동 관리하기	137
Meetings 폴더가 다루는 활동	137
외부 활동의 진행 상태	138
정보마다 정본을 맡는 작업 공간	138
외부 요청 intake의 구조화	139
이메일 스레드에서 복원하는 요청과 결정	139
약속과 준비라는 일정의 두 층	140
미팅 전의 같은 출발점	140
미팅 중 관찰과 해석과 결정의 분리	141
미팅 후 결과를 원래 작업 공간에 반영하기	141
새 분석 공동연구의 역할 배정	143
분석 담당자가 바뀔 때	143
계산 결과를 실험으로 검증하는 협업	143

Meetings 폴더와 위키의 역할	144
따라해보기	144
25장. Notion MCP로 연구실 프로젝트 관리하기	146
유료 Notion을 운영 비용에 포함하기	147
과제 하나에 데이터베이스 하나	147
프로젝트 보드에 만드는 카드	147
공동 랩노트와 진도 관리	148
카드 속성을 통일하기	148
LLM 위키에서 프로젝트 Notion으로 논문 공유하기	149
따라해보기	149
미팅 기록을 세 종류로 나누기	150
Zoom MCP에서 Notion 미팅 노트까지	151
설명회 영상에서 Notion 회의 노트까지	151
따라해보기	151
연구실 전체가 함께 보는 공통 Notion	153
실제 운영 순서	154
26장. LLM-Wiki와 에이전트를 이용하여 연구과제 관리하기	154
생애주기로 보는 그랜트	154
따라해보기	156
과제의 현재 상태를 통제하는 최소 문서	156
먼저 확정하는 공고 적합성	157
위키의 근거에서 Aim으로의 전환	157
역할과 접점으로 설계하는 참여 기관	158
실행 항목과 집중 시간으로 바꾸는 마감	158
심사 국면의 version 보존	159
합성 사례의 첫 국면인 제안서 심사	159
합성 사례의 두 번째 국면인 공동 집필 제안서	159
선정 이후 달라지는 관리의 종류	160
종료할 때 다음 과제로 넘기는 것	160
과제별 통제층이 위키가 되는 방식	161
27장. AI 비서와 예약 자동화 만들기	162
정해진 시각에 무언가를 돌리는 장치	162

예약 Agent라는 말이 가리키는 것	162
모듈로 나누는 반복 업무	163
생산자와 마지막 집행 단계의 분리	163
AI 에이전트를 Slack에 연결한 방법	164
산출물과 handoff와 log의 분리	164
최신성을 적지 않은 handoff의 위험	165
표로 정하는 자동화 권한의 경계	166
Transaction으로 다루는 일정 변경	166
비서가 일부러 하지 않는 일	167
나만의 비서를 만드는 순서	167
실패가 만든 운영 원칙	168
위기가 소유하는 자동화의 운영 계약	168
28장. 브리핑과 채널로 결과를 받는 법	169
아침 briefing의 항목과 순서	169
학생들과 함께 보는 연구 브리핑	170
자동화의 산출물이 도착하는 채널	171
따라해보기	172
읽히고 있는지 확인하는 법	172
29장. 행정 서류를 안전하게 처리하는 법	172
행정 서류를 소유한 업무	173
서류 작업의 라우터인 문서 규칙 묶음	173
양식을 다루는 처리 계층이 맡는 일	174
공식 원본 확인이라는 첫 관문	174
읽기와 빈칸 목록	176
값의 출처 결정	176
채우기와 고치기와 새로 만들기의 선택	177
별도 단계로서의 서명과 날인	178
구조 검증과 시각 검증의 병행	178
개인정보와 제출 package	179
첫 번째 사례인 외부 평가와 자문 서류	179

두 번째 사례인 과제 수행 중 인력 등록	179
세 번째 사례인 강의계획서와 교과 행정	179
따라해보기	180
처리를 멈춰야 하는 경우	180
30장. 원격으로 에이전트를 관리하기	181
24시간 돌아가는 컴퓨터 한 대	181
처음 한 번 연결하기	182
음성으로 하는 원격 작업	182
Codex 원격 접속으로 가능한 사례	182
원격 작업의 지식 기반인 LLM-Wiki	183
Codex Remote와 Claude Dispatch를 써 본 인상	183
원격 작업의 승인 규칙	183

LLM-Wiki: 지식이 축적되는 AI 작업 환경 만들기

저자: Joon An (안준용), 고려대학교

Last Update: 2026년 8월 5일

들어가며

대화형 AI로 좋은 답을 받아 실제로 써먹고도 반년 뒤에는 그 답이 어디 있는지 알 수 없는 경우가 흔하다. 당시에는 분명 도움이 되었지만 다시 찾으려면 어느 대화에 있었는지부터 헤매게 된다. 대화는 시간순으로 쌓이고 지식은 주제와 관계로 조직되므로 쌓이는 것과 축적되는 것은 서로 다른 일이 된다. 이 책은 그 사이를 메우는 작업 방식을 다룬다. 주제는 대화가 끝난 뒤에 무엇을 남길 것인가다. LLM-Wiki는 사람이 방향을 정하고 AI 에이전트가 읽고 갱신할 수 있도록 폴더와 Markdown 문서로 구성된 지속적인 작업 기억이다. 폴더 하나, 텍스트 파일 몇 개, 그리고 그 안에서 무엇을 어떻게 다룰지 정한 규칙이면 시작할 수 있다.

이 책의 구성

책은 세 부분으로 나뉘고 세 부분은 같은 순환을 서로 다른 책임의 크기로 다룬다. 파트 1은 코딩 경험이 없어도 읽을 수 있고 대상은 연구자를 넘어 학생, 교사, 기획자, 행정 담당자, 창작자처럼 파일과 지식을 다루는 사람 전부다. 열네 장을 마치면 도구를 고르고 권한을 정하고 자신의 관심사에 맞는 위키를 만들고 자료 한 건을 끝까지 넣고 이미 가지고 있는 논문을 한 번에 들이고 보충 자료의 표를 분석에 쓰고 질문에 근거로 답하고 그 답을 다시 위키에 되돌릴 수 있다. 파트 2는 연구를 시작한 학생을 위한 과정이다. 이 도구가 무엇을 대신해 주지 않는지를 먼저 짚고 논문 한 편을 분야의 역사와 현재 논쟁 속에서 읽는 법에서 출발해, 남의 방법을 자기 설계로 바꾸고 다른 사람과 판단을 맞추고 자신의 근거로 과학적 주장을 쓰는 데까지 간다. 파트 3은 여러 사람과 프로젝트를 동시에 책임지는 연구책임자를 위한 것으로, 학생 지도, 공동연구, 그랜트 생애주기, 예약 자동화, 행정 서류 처리를 하나의 운영 기억으로 잇는다.

읽는 방법

파트 1은 순서대로 읽는다. 각 장이 앞 장에서 만든 것을 쓰기 때문이다. 파트 2와 3은 필요한 장부터 읽어도 되지만 파트 1의 3장에서 9장까지를 건너뛰면 나머지가 공중에 뜬다. 각 장에는 20분에서 30분이면 끝나는 실습이 하나씩 있다. 실습을 건너뛰고 읽으면 이 책은 개념 설명서로 남고 실습을 하면 마지막 장에 도달했을 때 자기 위키가 하나 생긴다. 뒤쪽을 권한다.

장별 구성

장	제목	다루는 것
	파트 1. 누구나 시작하는 LLM-Wiki	
1	LLM-Wiki란 무엇인가	대화가 남지 않는 이유와 다섯 층의 구조
2	AI 에이전트와 작업 폴더	챗봇과 에이전트의 차이, 도구 고르기와 설치, 지침 파일
3	Karpathy의 제안에서 나의 연구 위키까지	세 층 원형과 네 층 확장, 설치와 카테고리 설계
4	논문 한 편을 LLM-Wiki에 넣는 법	ingest 여섯 단계와 종합 층 연결
5	논문 PDF를 ingest 하는 방법	추출기 선택과 조용히 발생하는 실패
6	논문 PDF 여러 개를 ingest하기	배치 절차, 제외 검사, 분모와 분자 보고
7	논문 PDF를 확보하는 방법	서지 정보 수집, 예약 후보 목록, paywall 앞의 절차
8	Supplementary Table과 Document를 ingest 하는 방법	자원 층, manifest, 표를 분석 입력으로 쓰기
9	질문이 지식으로 축적되는 법	검색과 종합, 답을 위키로 되돌리기
10	논문의 구조를 해부하기	인용의 기능 분류와 신뢰 전략 비교
11	위키에서 연구 산출물로	아젠다, 발표, 원고로 나가는 경로
12	Obsidian으로 보기와 채팅으로 묻기	그래프가 보여 주는 것과 대화로 지식을 내재화하기
13	에이전트에게 주는 권한의 경계	Codex와 Claude Code의 권한 모드, 되돌릴 수 있는가라는 기준
14	말로 지시하고 맥락을 넘기는 법	긴 수다 세션, 작업 공간별 라우팅
	파트 2. 연구를 시작하는 학생의 LLM-Wiki	
15	논문이란 무엇인가	분야의 대화에 들어가기, field map
16	상용 AI에 넣으면 안 되는 자료	자료 경계, 폐쇄망 위키, 로컬 모델
17	데이터 분석을 위한 폴더 구성	분석 프로젝트 폴더, method page와 연구 설계
18	공동 연구 미팅을 준비하는 법	사전 자료, 결정 기록, 회의 전사와 노트
19	논문 원고 폴더 구성하기	논문 폴더, claim-evidence 지도와 원고 구성
20	새로운 주제 온보딩 하기	고른 논문 건네기, 잘라 낼 때의 링크, 돌아오는 경로
21	에이전트가 대신해 주지 않는 것, 당신의 과학	요약과 얇은 거리, 말하기와 글쓰기 훈련, 과학적 책임
	파트 3. 연구책임자의 LLM-Wiki	
22	MCP 기본 개념	메일, 달력, 팀 대화, 팀 문서, HWP 양식
23	도반, 당신의 공동연구자를 지도하는 법	사람별 기록과 개입의 승격
24	Meetings 폴더로 PI의 외부 활동 관리하기	세미나, 자문, 미팅, 출장과 반복 업무

	장 제목	다루는 것
25	Notion MCP로 연구실 프로젝트 관리하기	앵커 규약, 공유 계층과 지식 저장소
26	LLM-Wiki와 에이전트를 이용하여 연구과제 관리하기	여섯 국면과 Aim별 근거 표
27	AI 비서와 예약 자동화 만들기	스케줄러의 종류, 모듈 분리, 권한 경계와 운영 계약
28	브리핑과 채널로 결과를 받는 법	아침 브리핑과 연구 브리핑, 채널을 나누는 기준
29	행정 서류를 안전하게 처리하는 법	HWP 양식, 서명 배치, 멈추는 조건
30	원격으로 에이전트를 관리하기	24시간 호스트, Codex Remote, 음성 지시와 원격 승인

이 책이 약속하지 않는 것과 사람의 몫

이 방식이 주는 것은 추적 가능성이다. 틀린 정리를 넣으면 틀린 정리가 축적되고 검증하지 않은 요약물 정본으로 삼으면 그 오류가 이후의 모든 종합에 들어간다. 어떤 주장이 어느 근거에서 나왔는지 되짚을 수 있다는 것이 이익이고 그 위에서 정확성을 만드는 일은 사람의 몫으로 남는다. LLM-Wiki와 에이전트로 많은 논문을 읽고 정리하는 일은 자동화할 수 있다. 이 책은 그 자동화를 만드는 방법에 상당한 지면을 쓴다. 그 내용을 이해하고 근거에 입각해 주장하는 주체는 사람이다. 두 논문의 결과가 어긋날 때 그것이 진짜 모순인지 조건의 차이인지 판단하는 일, 어느 근거가 결론을 떠받칠 만큼 강한지 재는 일, 그리고 그 판단이 틀렸을 때 책임을 지는 일이 여기에 속한다. 에이전트는 후보를 모아 오고 형식을 맞추고 빠진 것을 알려 주지만 무엇을 믿을지는 정해 주지 않는다. 이 구분을 흐리면 이 방식이 오히려 위험해진다. 정리된 문서가 늘어날수록 그 안의 문장이 검증을 거친 것처럼 보이기 때문이다. 위키가 잘 만들어져 있을수록 읽는 사람은 그것을 더 쉽게 믿고 그래서 잘못된 정리 하나가 더 멀리 퍼진다. 축적된 것을 사람이 검증해야 신뢰가 생긴다. 이 책이 절차와 검증에 많은 지면을 쓰는 이유가 여기에 있다.

서문

25년 하반기에 AI 에이전트를 처음 접했다. 당시에 간단한 싱글셀 분석을 할게 있었는데 데이터의 위치를 알려주고 공동연구자에게 받은 배경을 설명해주었다. 30분이 채 지나지 않아서 분석결과를 들고왔다. 분석 프로그램 설치부터, 분석, 그리고 논문에 들어갈 그림까지 굉장히 빠른 시간에 거의 정확하게 분석을 해냈다. 그때 기분은 마치 자다가 외계인 우주선에 납치되어 화성에 떨어진 느낌이었다. 그때부터 겨울 내내 에이전트를 다루면서 많은 작업을 했다. 인생에서 이보다 짜릿한 순간이 있었을까 싶을 정도로. 당시에 에이전트를 구축하거나 바이브 코딩을 하는 사람들이 남긴 SNS의 글을 보면서 여러가지 방식들을 시도해봤다. 그때, OpenAI의 창립 멤버인 안드레이 카파시 (Andrej Karpathy)의 글을 많이 보게 되었다. 카파시의 글과 인터뷰는 마치 나를 새로운 세계로 인도하는 길 같았다.

2026년 4월에 카파시가 그의 깃헙 gist에 짧은 제안 하나를 공개했다. 바로 LLM-Wiki라는 개념이다. 이름이 다소 단순하고 아저씨 같지만 내용을 살펴보면 더 단순하다. 질문이 들어올 때마다 원자료 조각을 다시 찾아 답을 합성하는 대신, LLM이 지속적인 Markdown 위키를 미리 만들고 갱신하게 하자는 것이었다. 원자료는 고치지 않는 층으로 두고 위키는 LLM이 소유하며 구조와 규칙은 사람과 LLM이 함께 발전시킨다. 이것을 굉장히 간단한 규칙문으로 gist에 올려두었고 이러한 규칙을 바탕으로 작업하면서 유저는 지식과 근거의 원천으로 사용한다. 에이전트를 다루는 카파시의 방식이 매우 매력적으로 보였다.

가만히 생각해보니, 이 방식을 연구에 활용해보면 좋겠다는 생각이 들었다. 대학원에서 학위를 시작할 때를 생각해보자. 논문을 읽고 또 논문을 읽는다. 석사 첫 학기에 대략 30편 정도의 논문을 읽었던 것 같은데 그때 읽은 논문들이 서로 연결되어 학위 주제를 만드는 방향을 제시한다. 나의 석사 학위는 오래전이지만 그때 수집되고 정제된 지식이 계보를 이루어 지금의 나라는 연구자를 규정한다. 연구자의 모든 연구 활동은 날마다의 기록이 그 사람의 연구적 방향성과 품질을 결정한다. 그래서 카파시 박사님이 말한 LLM-Wiki를 사용하면, 이 과정을 보다 효율적으로 할 수 있겠다 생각이 들었다. 이 책은 그 제안을 내 연구에 적용해 몇 개월간 구축한 방법을 다루고 있다. 초기의 시행착오와 방법 구축이 있었고 지금 내 LLM-Wiki는 약 5번 정도의 진화를 거쳐, 내 삶의 가장 핵심적인 부분이 되었다. 여러가지 운영 원칙에 대한 노하우를 공개한다.

첫째는 내가 논문을 읽는 방식이다. 한 학기를 쓰고 나니 사용 빈도와 만족도가 계속 늘었다. 논문을 들일 때는 두 가지 방식을 나눠 쓰는데 하나는 읽고 정리해 제자리에 넣는 일반적인 처리이고 다른 하나는 기존 위키와 합쳐 개념을 잇고 새 질문을 뽑아내는 데까지 가는 처리다. 뒤의 것은 시간이 더 들지만 그 논문 한 편이 위키 전체를 조금씩 고친다. 이렇게 쌓이고 나서야 되는 일들이 생겼다. 공동연구를 함께하는 분들의 논문을 모두 찾아 넣고 그분들이 다루는 실험과 기술을 파악해 실험 계획을 짜 보는 일, 오래전 그 분야를 만든 논문들을 모아 서론을 어떻게 열고 고찰을 어떻게 닫았는지 전개 방식을 정리하는 일, 같은 주제를 두고 관점이 갈리는 연구 그룹들의 논문을 나란히 놓아 보는 일이 그렇다. 석사 첫 학기에 서른 편을 읽으며 머릿속에서 하던 일을 지금은 풀더가 대신 붙들고 있어 준다는 느낌에 가깝다. 다만 비용이 실제 문제여서 같은 논문을 연구실의 여러 사람이 각자 처리하면 같은 작업이 반복된다. 그래서 누군가 한 편을 읽으면 정리 결과와 원문을 함께 공유하는 자리를 따로 두었다.

둘째는 학생을 가르치는 방식이다. 지금 학생들이 연구실에 입학해서 하는 연구 주제는 대부분 여러 분야에 걸쳐 있다. 그래서 학생들은 넓은 범위의 지식을 배우고 그것들을 서로 이어야 한다. 내가 하는 유전체 연구는 지난 10년 사이 데이터가 쏟아지면서 요구하는 지식의 폭이 넓어졌다. 전산학과 통계학의 방법으로 데이터를 다루는 동시에, 그 관찰로 밝혀진 생물학을 깊은 수준까지 알아야 한다. 그래서 처음 연구를 배우는 학생에게는 각 연구가 어떤 흐름에서 발전해 나왔고 그 결과가 무엇을 뜻하는지를 함께 가르쳐야 한다. 연구자가 되려면 자기 주제를 스스로 밀고 나가야 하는데 그 주제를 자기 말과 글로 꺼내 놓는 과정이 빨라지니 자신감이 빨리 붙는다. 훈련 과정에서 그것이 틀렸다고 하더라도 자기 언어로 풀어내 보는 일은 중요하고 거기서 다음 질문을 이어 붙이는 일은 더 중요하다. 온보딩에서 특히 그렇다. 새로 온 학생은 좋은 논문을 골라내는 일부터 어려워하므로 내가 먼저 중요한 논문을 읽고 주제의 흐름을 파악한 다음 그 결과를 건네는 대신 골라낸 논문들을 건넨다. 정리는 학생이 자기 위키에서 직접 하게 한다. 정리된 문서를 받아 읽는 것과 자기가 정리해 보는 것은 남는 것이 다르고 지금까지 학생을 훈련시킨 방식 가운데 이것이 가장 효과가 좋았다. 이 방식을 처음 접한 분들은 학생이 논문을 대충 읽고 내용을 파악하지 못하면 어떻게 하느냐고 걱정하시는데 실제로 써 보면 그 걱정은 크지 않았다.

셋째는 연구책임자로 연구실을 운영하는 방식이다. 사람이 늘고 과제가 늘면 지금 무엇이 어디까지 왔는지를 묻는 일 자체가 일이 된다. 답이 한 곳에서 나오지 않고 메일과 달력과 공유 문서와 메시지에 흩어져 있기 때문이다. 그래서 각 도구가 무엇을 소유하는지 정하고 위키는 그것을 가리키게 두었다. 논문을 읽는 위키 옆으로 원고를 쓰는 풀더와 분석을 하는 풀더가 갈라져 나온 것도 같은 이유다. 각각에 풀더 배치와 작업 규칙을 정해 두고 무엇을 왜 바꿨는지를 작업 로그로 남기면, 세 종류의 작업이 각각 자기 규칙과 자기 기록을 소유한다. 사람에 관한 기록도 같은 원칙으로 쓴다. 날짜가 붙은 사실만 쌓고 점수나 등급은 만들지 않으며 오래된 관찰은 현재 상태로 읽히지 않도록 표시가 내려간다. 매일 아침 판단할 것을 모아 주는 자리를 하나 두었더니 흩어진 것을 찾아다니는 시간이 줄었고 그랜트와 행정 서류처럼 형식이 정해진 일에서는 자동화가 어디서 멈춰야 하는지를 정하는 일이 더 중요했다.

이러한 목적을 갖고 사용하다보니, LLM-Wiki는 내가 하는 모든 활동에 대하여, 다시 말해 읽고 정리하고 지식을 체득하는 과정을 계속 이어 갈 구조를 만들어 주었다. LLM-Wiki 사용법에 대한 강의를 하다보면, 수강생들이 자주 하는 질문 (or 걱정?)이 하나있다. AI로 논문을 읽는 것은 클릭 몇 번이고 그렇게 해서 지식은 남지 않는다는 것이다. 일부는 맞는 말인데, 꼭 그렇진 않다. 여기에는 조건이 하나 붙는다. AI 에이전트로 처리하는 과정에서 나오는 내용을 연구자는 적어도 한번은 읽는다. 왜냐하면, AI와 다르게 인간은 그 내면에 깊숙이 패인 어떤 질문에 대한 호기심이 있기 때문이다. 그래서 AI 에이전트와 하는 모든 과정을 함께 한다. AI는 앞으로 더 발전할 거고 더 정확한 수행을 하는 기계가 되겠지만, 인간 사회에서 일어나는 다양한 일을 과학적 방법으로 풀고자 하는 동기는 인간의 고유한 내면에서만 나온다. 그래서 AI 에이전트나 LLM-Wiki를 사용한 뒤에도, 어떤 질문이 생겨 위키에 물어보는 중에도, 발표를 준비하려고 그 문서를 다시 열어 보는 순간에도, 인간 과학자의 한쪽에는 지식이 반드시 남는다.

이 책은 지난 반년 동안 쓰면서 얻은 경험과 방법을 우리 연구실 학생들을 위해, 커뮤니티의 연구자들을 위해 공유한다. 그리고 갖은 잡무와 국가 발전을 위해 없어져야 할 여러가지 행정문서 올림픽으로 오늘도 고생하시는 교수와 연구책임자들을 위해 바친다. 우리의 삶이 보다 본질적인 질문과 과학을 하기 위해, 그리고 AI 에이전트와 LLM-Wiki가 그런 변화에 씨돌이 되기를 바라며.

1장. LLM-Wiki란 무엇인가

어떤 사람이 반년 동안 AI와 매일 대화했다고 하자. 논문을 요약시키고 코드를 고치고 발표 자료의 흐름을 상의하고 학생에게 보낼 메일의 어조를 다듬었다. 대화 하나 하나는 만족스러웠다. 그런데 반년이 지난 뒤 “지금까지 정리한 내용 중에 이 주제와 관련된 것을 모아 달라”고 요청하면 아무것도 나오지 않는다. 대화는 서비스 안에 시간순으로 쌓여 있지만 그 안에서 무엇이 결론이었고 무엇이 폐기된 아이디어였는지는 구분하기 어렵다. LLM-Wiki는 반년 동안 쌓인 판단과 결과물을 풀더 하나와 Markdown 문서 몇 개에 기록하고 서로 연결한다. 문서가 사용자의 파일 시스템에 남으므로 다음 대화에서도 다시 읽을 수 있다. 오늘 바로 시작할 수 있는 크기다.

대화가 남지 않는 이유

대화형 AI를 쓰는 사람들이 반복해서 겪는 상황이 세 가지 있다. 첫째는 같은 배경 설명과 결과 형식을 매번 다시 적는 일이다. 웹에서 ChatGPT를 쓸 때 나는 새 대화를 열 때마다 무엇을 연구하는지, 이 논문을 왜 읽는지, 어느 측면을 중점적으로 볼지 설명해야 한다. 필요한 정보를 표로 받을지 문단으로 받을지, 열 이름과 요약 길이는 어떻게 할지도 다시 정한다. 대화가 바뀔 때마다 연구 맥락과 출력 형식을 복원하는 데 시간이 들고 설명이 조금씩 달라지면 결과의 형식도 흔들린다. 둘째는 이전에 내린 결정을 잊는 일이다. 두 달 전에 어떤 방법을 쓰지 않기로 정했는데 그 이유가 기억나지 않아 같은 논의를 다시 한다. 셋째는 좋았던 답을 다시 찾지 못하는 일이다. 분명 잘 정리된 설명을 받았는데 어느 대화에 있었는지 알 수 없어 결국 다시 물어본다. 세 상황의 원인은 같다. 채팅 기록은 시간순으로 쌓이지만 지식은 시간순으로 조직되지 않는다. 어제 나는 대화와 석 달 전에 나는 대화가 같은 주제를 다루었다면 그 둘은 서로 옆에 있어야 하는데 채팅 기록에서 둘 사이의 거리는 석 달이다.

1. 최근 한 달 동안 반복해서 찾았거나 남에게 설명한 주제를 세 가지 적는다.
2. 각 주제에 대해 그때 만들어진 결과물이 지금 어디에 있는지 적는다. 찾을 수 없다면 찾을 수 없다고 적는다.
3. 각 주제에 대해 무엇이 사라졌는지 한 줄로 적는다. 사라진 것이 사실인지, 판단인지, 근거인지, 아니면 그 셋을 이었던 맥락인지 구분한다.
4. 세 주제를 놓고 “내가 만들 LLM-Wiki는 무엇을 기억해야 하는가”에 대한 다섯 문장짜리 선언문을 쓴다.
5. 선언문을 readme 또는 이라는 이름의 Markdown 문서로 저장해 앞으로 만들 위키의 첫 문서로 삼고 세 주제와 각각에서 사라진 것을 표로 정리해 같은 문서 아래에 붙인다.

성공 기준은 세 주제 중 최소 하나에서 사라진 것이 판단이나 맥락이었다는 점을 스스로 확인하는 것이다. 사실은 다시 찾을 수 있지만 판단과 맥락은 다시 만들어야 하고 그것이 이 책 전체가 다루는 문제다. 2장에서 폴더를 만들 때 이 표가 무엇을 어디에 둘지 정하는 근거가 된다. 지식은 주제와 관계를 축으로 조직되어야 다시 꺼낼 수 있고 시간축은 그 조직을 만들어 주지 않는다. 검색 기능이 있어도 사정은 크게 달라지지 않는다. 검색은 내가 그때 어떤 단어를 썼는지를 기억해야 작동하고 무엇보다 서로 다른 대화에서 내린 판단들이 서로 모순되는지 알려주지 않는다. 대화의 편리함과 기억의 지속성은 서로 다른 문제다. 대화형 인터페이스가 편리한 이유는 준비 없이 시작할 수 있기 때문이다. 폴더를 정하지 않아도 되고 파일 형식을 고르지 않아도 되고 어디에 저장할지 미리 결정하지 않아도 된다. 그 편리함의 대가가 바로 아무것도 정해지지 않았다는 사실이다. 저장할 위치를 정하지 않았으므로 저장되지 않고 조직할 축을 정하지 않았으므로 조직되지 않는다.

LLM-Wiki의 한 문장 정의

LLM-Wiki는 폴더 안의 Markdown 문서를 사람이 소유하고 AI 에이전트가 함께 읽고 고치는 지식 작업 환경이다. 정의에 들어 있는 요소를 하나씩 풀어 보면 뜻이 분명해진다. 폴더 안에 있다는 것은 문서가 사용자의 파일 시스템 위에 놓인다는 뜻이다. Markdown 문서라는 것은 사람이 직접 열어 읽고 고칠 수 있는 일반 텍스트라는 뜻이다. 사람이 소유한다는 것은 무엇을 남기고 무엇을 버릴지 사람이 정한다는 뜻이고 AI가 함께 읽고 고친다는 것은 AI가 자료를 찾고 초안을 쓰고 형식을 맞추는 일을 실제로 수행한다는 뜻이다. 이 환경에서 일은 하나의 순환을 이룬다. 원자료가 들어오고 읽어서 정리하고 기존 지식과 연결하고 여러 문서를 묶어 종합하고 그 종합에서 다음 행동이 나온다. 이 순환의 각 단계가 서로 다른 문서로 남고 문서끼리 링크로 이어진다. 다음에 비슷한 질문이 들어오면 이미 만들어진 정리와 종합에서 출발한다. 같은 일을 두 번 하지 않는다는 것이 이 구조의 이익이다. Wiki라는 이름을 쓰는 이유는 문서 사이의 연결과 지속적인 개정에 있다. 새 문서가 들어오면 기존 문서를 고치고 고친 흔적이 남고 문서끼리 서로를 가리킨다. 위키를 만든다는 말은 계속 고쳐지는 문서 체계를 유지한다는 뜻이다. 이 책에서 위키라는 말은 언제나 이 의미로 쓴다.

LLM-Wiki를 시작하게 된 계기

챗지피티가 등장하고 AI를 이용해서 논문을 빠르게 요약하고 읽는 일이 많아졌다. 읽어야 할 논문은 늘 밀려 있고 전문을 처음부터 끝까지 눈으로 훑을 시간은 없으니, 흥미로운 논문을 발견할 때마다 PDF를 대화창에 올려 빠르게 파악하는 방식을 썼다. 이 방식은 한 편을 읽는 동안에는 잘 작동한다. 문제는 며칠 뒤에 온다. 이삼일 지나 또 흥미로운 논문을 발견하고 같은 방식으로 읽다 보면, 그 논문이 며칠 전에 읽은 논문과 이어져 있다는 것을 알아채게 된다. 연구자가 읽는 논문은 무작위로 흩어져 있지 않다. 자기 관심사를 따라 비슷한 주제를 계속 찾아 읽으므로 새로 발견한 논문의 내용이 앞서 읽은 것과 맞물리는 일이 오히려 흔하다. 그래서 다음 단계는 자연스럽게 정해졌다. 두 편의 PDF를 한 대화에 함께 올려 두 논문을 비교해 달라고 시키는 것이다.

두 편일 때는 이 방식이 잘 통한다. 문제는 편수가 늘어날 때 생긴다. 같은 주제를 따라 읽다 보면 그런 논문이 열 편이 되고 스무 편이 되고 쉰 편까지 늘어난다. 그때 이 논문들을 어떻게 서로 비교하면서 읽을 것인가라는 질문이 남는다. 웹

대화창에는 한 번에 올릴 수 있는 파일 수가 정해져 있고 대화에 남는 기록에도 한계가 있어 앞서 읽은 내용을 언제까지나 기억하지는 못한다. 열 편짜리를 읽을 때 첫 편에서 무엇을 확인했는지가 이미 대화 밖으로 밀려나 있다는 뜻이다. 논문이 늘어날수록 비교의 가치는 커지는데 비교할 수 있는 능력은 줄어드는 구조였다. 그 무렵 Google의 NotebookLM이 나왔다. 자료 여러 편을 한 노트북에 넣어 두고 그 안에서 묻고 요약하게 하는 방식이라, 한 편씩 올려 가며 읽던 웹 대화창보다 훨씬 나았다. 무료 등급 기준으로 노트북 하나에 자료를 50개까지 넣을 수 있었고 한 주제의 논문을 모아 놓고 읽는 일이 처음으로 가능해졌다. 그러나 두 가지 한계가 남았다. 하나는 그 편수를 넘기기 어렵다는 것이고 다른 하나가 더 중요했다. 한 노트북에서 쉰 편을 읽고 다른 노트북을 열어 또 쉰 편을 읽을 수는 있지만 두 노트북에 나뉘어 들어간 논문들 사이의 연결은 볼 수 없다. 연구자가 읽는 논문은 노트북 경계를 따라 나뉘지 않는다. 앞의 쉰 편과 뒤의 쉰 편은 같은 관심사에서 나온 것이므로 서로 이어져 있는데 그 연결을 확인할 방법이 없었다. 결국 같은 문제가 규모만 키워 돌아온 셈이다.

조금 더 넓게 보면 문제가 하나 더 있었다. 논문을 읽고 머릿속에 요약이 남는 것으로 일이 끝나지 않는다. 거기서부터 질문이 갈라져 나오고 그 질문마다 다른 작업이 붙는다. 논문을 쓰고 발표 자료를 만들고 미팅을 준비하고 공동연구를 설계한다. 네 작업 모두 같은 논문에서 출발하는데 각각 다른 곳에서 처음부터 다시 시작하고 있었다. 읽기와 그 뒤에 오는 일을 같은 폴더의 문서로 이어 갈 구조가 필요했다. 그러던 차에 안드로이드 카파시가 제안한 위키 방식을 보고 이것이라면 연구에 본격적으로 쓸 수 있겠다고 판단했다. 이 책은 그 판단에서 시작해 실제로 운영해 본 결과다.

논문을 읽고 요약하는 일은 어렵지 않다. AI가 등장하기 전에도 어렵지 않았고 지금은 더 쉬워졌다. 어려운 것은 이미 읽은 논문과 지금 읽는 논문이 어떤 관계인지 기억하는 일이다. 같은 현상을 다른 방법으로 측정한 두 논문이 서로 다른 결론을 냈을 때, 그것이 진짜 모순인지 아니면 측정 대상이 달랐던 것인지는 두 논문을 나란히 놓아야 판단할 수 있다. 논문 한 편을 요약한 문서가 백 개 있어도 이 판단은 만들어지지 않는다. 요약은 논문 안에서 끝나고 관계는 논문 사이에 있기 때문이다. 같은 지식이 서로 다른 작업에서 반복해서 필요해진다는 문제도 있다. 어떤 방법의 한계를 설명하는 일은 연구 질문을 세울 때도 필요하고 강의에서 설명할 때도 필요하고 학생의 분석 계획을 검토할 때도 필요하고 논문의 고찰을 쓸 때도 필요하다. 네 번 모두 처음부터 다시 정리한다면 네 번 모두 조금씩 다른 설명이 만들어지고 어느 것이 현재의 판단인지 알 수 없게 된다. 한 번 정리한 내용을 네 작업이 함께 참조하도록 연결해야 한다. AI를 쓸수록 산출물은 늘어난다. 요약, 초안, 표, 코드, 슬라이드가 계속 만들어진다. 그런데 만들어진 것을 다시 사용할 구조가 없으면 늘어나는 것은 일회성 산출물뿐이다. 파일이 많아지는 것과 지식이 축적되는 것은 다른 일이다. 지식이 축적되었다는 말은 새 질문에 답할 때 예전에 만든 것에서 출발할 수 있다는 뜻이고 그러려면 산출물이 서로를 가리키고 있어야 한다.

일반적인 메모, 검색, 채팅과의 차이

방식	잘하는 것	쉽게 잃는 것
메모 앱	빠른 기록과 개인 정리	출처와 판단 규칙, AI가 작업을 이어 갈 구조
검색	이미 존재하는 정보 찾기	내가 이전에 내린 결정과 관심사의 누적
웹 채팅	즉각적인 설명과 아이디어 생성	폴더 전체의 맥락, 지속적으로 관리되는 정보
문서형 RAG LLM-Wiki	저장 문서에서 관련 내용 검색 자료와 해석, 연결, 규칙, 작업 이력의 누적	지식 구조의 능동적 개정과 실행 규칙 자동으로 정확해지지 않으며 사람의 검증이 필요함

표의 마지막 줄에 적은 대로 이 방식은 자동으로 정확해지지 않는다. 틀린 정리를 넣으면 틀린 정리가 축적되고 검증하지 않은 요약을 정본으로 삼으면 그 오류가 이후의 모든 총합에 들어간다. 이 구조가 주는 것은 추적 가능성이다. 어떤 주장이 어느 원자료에서 나왔는지, 언제 누가 고쳤는지, 지금 이 판단이 무엇에 근거하는지를 되짚을 수 있다는 것이 이 방식의 이익이고 그 위에서 정확성을 만드는 일은 사람의 몫이다. 메모 앱은 여전히 빠르고 검색은 여전히 필요하고 웹 채팅은 여전히 생각을 열어 보기에 좋다. LLM-Wiki는 이 도구들의 결과를 어디에 놓을지 정하는 상위 작업 방식이다. 웹 채팅에서 좋은 설명을 얻었다면 그것을 위키의 어느 문서에 반영할지 정하고 검색으로 찾은 자료라면 원자료로 남길지 아니면 참고만 하고 넘어갈지 정한다. 도구를 여러 개 쓰면서도 그 결과가 흩어지지 않게 붙잡아 두는 일이다.

LLM-Wiki를 이루는 다섯 층

이 환경은 성격이 다른 다섯 종류의 문서로 이루어진다. 층을 나누는 이유는 각 층의 수정 권한과 신뢰 수준이 다르기 때문이다. 원자료는 논문, 회의록, 데이터, 링크, 이미지처럼 다시 확인할 수 있는 근거다. 이 층은 고치지 않는다. 원문 PDF를 편집하기 시작하면 무엇이 원래 있던 내용이고 무엇이 나중에 붙인 해석인지 구분할 수 없게 된다. 정리 문서는 원자료의 내용과 맥락을 읽을 수 있는 형태로 옮긴 기록이다. 이 층은 원자료를 대체하지 않는다. 수치나 조건이 결론을 좌우하는 대목에서는 원자료로 돌아가야 한다. 연결과 종합은 여러 문서를 비교하고 개념, 쟁점, 질문으로 엮은 문서다. 앞서 말한 두 논문의 모순이 진짜 모순인지 판단하는 일이 이 층에서 일어난다.

운영 규칙은 파일을 어디에 두고 무엇을 확인하며 어떻게 고칠지를 정한 지침이다. 이 층이 없으면 사람마다, 세션마다 다른 방식으로 문서를 만들게 되고 결국 구조가 무너진다. 작업 이력과 다음 행동은 무엇이 바뀌었고 어떤 문제가 남았는지를 이어 주는 기록이다. 이 층이 있어야 다음 세션이 이전 세션의 끝에서 시작할 수 있다. 처음부터 다섯 층을 모두 만들 필요는 없다. 첫 실습은 원자료, 정리 문서, 그리고 그 둘을 가리키는 목차 문서 세 가지로 시작한다. 나머지 층은 필요가 생겼을 때 추가한다. 종합 문서는 비교할 대상이 두 건 이상 쌓였을 때 의미가 있고 운영 규칙은 같은 판단을 세 번쯤 반복했을 때 쓰면 된다. 층 다섯 개를 미리 만들어 놓고 채우려 들면 대부분 빈 채로 남는다.

사람이 맡을 일과 AI가 맡을 일

역할을 나누는 기준은 책임이다. 사람이 맡는 것은 목적을 정하는 일, 자료의 신뢰도를 판단하는 일, 무엇이 중요한지 고르는 일, 충돌하는 근거 사이에서 결정하는 일, 그리고 그 결정의 최종 책임을 지는 일이다. 이 다섯 가지는 AI가 대신할 수 있는지 여부와 무관하게 사람이 해야 한다. 잘못된 판단의 결과를 감당하는 것이 사람이기 때문이다. AI 에이전트가 맡는 것은 파일을 탐색하는 일, 초안을 쓰는 일, 형식을 통일하는 일, 연결 후보를 제안하는 일, 반복적인 검사를 수행하는 일, 그리고 빠진 것을 찾아내는 일이다. 이 여섯 가지는 양이 많고 규칙이 분명하며 사람이 하면 지치는 일이다. 특히 마지막 두 가지가 실제 이익이 크다. 문서 200개에서 링크가 끊긴 곳을 찾거나 새로 들어온 논문과 관련될 만한 기존 문서를 찾는 일은 사람이 성실하게 하기 어렵다. 둘이 함께 하는 영역도 있다. 문서 구조를 설계하는 일, 기존 주장과 새 증거를 비교하는 일, 다음 질문을 도출하는 일이 여기에 속한다. 이 세 가지는 AI가 후보를 만들고 사람이 고르는 방식이 잘 작동한다. AI가 비교 축을 다섯 개 제안하면 사람이 그중 실제로 의미 있는 두 개를 고르는 식이다. 두 논문의 결과가 왜 다른지 같은 판단에서 AI는 그럴듯한 설명을 만들어 낼 수 있고 그 설명이 맞는지 확인하려면 사람이 원자료로 돌아가야 한다. AI가 생각을 대신하는 것과 AI가 생각의 흔적을 정리하고 다시 꺼내 주는 것은 다르다. 앞의 것을 기대하면 실망하거나 더 나쁘게는 검증하지 않은 결과를 믿게 된다. 뒤의 것을 기대하면 실제로 얻을 수 있다.

사람마다 달라지는 쓰임새

파일과 지식을 다루고 관심사가 시간이 지나도 이어지고 같은 주제를 여러 번 다시 만나는 일이라면 형태만 달라진다. 나는 LLM-Wiki에서 매 작업의 로그를 남긴다. 로그에는 어떤 문서를 만들고 고쳤는지, 어떤 판단을 보류했는지, 다음 작업에서 무엇을 확인할지가 쌓인다. 반복해서 쓰는 지시와 문서 형식은 운영 규칙에 반영한다. 그렇게 쓰다 보면 에이전트는 내 폴더 구조와 자주 쓰는 형식, 작업 순서를 기준으로 움직이게 된다. 위키의 내용과 함께 작업 환경도 내 방식에 맞춰 변한다. 같은 도구로 시작해도 사용자의 연구와 습관에 따라 전혀 다른 위키가 만들어진다. 학생에게 이 환경은 관심사가 학기마다 이어지는 학습 지도가 된다. 이번 학기에 배운 개념이 다음 학기의 과목과 어떻게 이어지는지가 문서로 남는다. 연구자에게는 주장과 증거가 갱신되는 연구 기억이 된다. 새 논문이 들어올 때마다 기존 판단이 강화되는지 좁아지는지가 기록된다. 교사에게는 수업 반응과 설명 방식이 축적되는 교육 위키가 된다. 어떤 설명이 어느 학년에서 통했는지가 남는다. 기획자와 실무자에게는 결정과 근거, 후속 작업이 연결되는 프로젝트 기억이 된다. 왜 그렇게 정했는지를 반년 뒤에 되짚을 수 있다. 창작자에게는 소재와 초안, 피드백, 설정이 서로를 참조하는 작업실이 된다. 이 다섯 가지에서 공통된 것은 참조이고 저장한 것들이 서로를 가리키고 있는지가 이 환경의 성패를 가른다.

계속 갱신되는 순환

LLM-Wiki의 작업 순환

원자료 수집에서 정리, 연결, 종합, 판단, 실행으로 이어진다. 실행에서 생긴 질문이 다시 자료 수집으로 돌아오면서 위키가 갱신된다.

이 그림에서 눈여겨볼 곳은 판단과 실행에서 질문으로 되돌아가는 왼쪽 화살표다. 자료가 들어오는 것만으로는 위키가 자라지 않는다. 새 자료가 기존 생각을 강화하는지, 적용 범위를 좁히는지, 아니면 틀렸다고 드러내는지를 판단해 종합 문서에 반영할 때 비로소 지식이 축적된다. 이 판단을 건너뛰면 문서 수만 늘고 다음 질문에 답하는 능력은 그대로다. 잘 나눈 폴더에 정리되지 않은 파일을 넣으면 잘 나눈 채로 흩어지므로 구조는 필요조건까지만 채워 준다. 좋은 LLM-Wiki는 새 질문과 작업을 더 정확하게 이어 주는 위키다. 판단 기준은 단순하다. 지금 이 위키에 새 질문을 던졌을 때, 반년 전의 나와 지금의 내가 같은 출발점에 서는가. 그렇다면 위키가 일하고 있는 것이고 다른 출발점에 선다면 문서가 쌓이기만 한 것이다.

2장. AI 에이전트와 작업 폴더

한 학기 강의를 준비하는 사람을 떠올려 보자. 매주 자료를 올리고 설명을 다듬고 예제를 고른다. 3주차에는 어떤 주제를 넣었다가 수강생 수준에 맞지 않아 뺐고 5주차에는 예제를 통계 쪽에서 생물 쪽으로 바꿨다. 8주차쯤 되면 그 결정들이 왜 그렇게 내려졌는지 기억나지 않는다. 다음 해에 같은 강의를 다시 준비할 때 남아 있는 것은 슬라이드 파일 몇 개이고 슬라이드가 왜 그 순서인지, 무엇을 왜 뺐는지는 어디에도 없다. 작업 폴더에 결정과 이유를 문서로 남겨야 다음 해 강의에서 이어 쓸 수 있다. 프롬프트를 정교하게 다듬는 일만으로는 그 기록이 만들어지지 않는다.

챗봇에서 에이전트로

이 책에서 챗봇은 대화에 답을 돌려주는 사용 방식을 가리킨다. 입력은 대화창에 적은 글과 붙인 자료이고 기본 출력은 대화 기록에 나타난 글이다. 대화는 여러 차례 이어질 수 있고 프로젝트 자료나 기억을 함께 쓸 수도 있다. 다만 로컬 파일이나 외부 상태를 바꾸는 도구를 붙이지 않았다면 결과를 어느 문서에 남길지는 사람이 따로 정한다. 이 방식은 개념을 설명하고 아이디어를 여러 갈래로 펼쳐 보이고 초안을 빠르게 만드는 일을 잘한다. AI 에이전트의 동작 단위는 여러 단계로 이루어진 하나의 작업이다. 목표를 받으면 어떤 파일과 도구가 필요한지 정하고 자료를 읽고 필요한 부분을 고치고 만든 결과가 정해진 규칙에 맞는지 검사한다. 폴더 기반 작업에서는 중간 산출물과 최종 결과가 파일로 남기 때문에 작업이 끝난 뒤에도 무엇이 만들어졌는지 확인할 수 있다. 열 편의 문서에서 링크가 끊긴 곳을 찾아 고치는 일처럼 같은 동작을 여러 번 반복하고 실제 상태를 바꾸는 일이 에이전트가 유용한 지점이다. 같은 ChatGPT나 Claude 안에서 대화에 답만 받으면 대화형 사용이고 로컬 프로젝트나 외부 도구로 여러 단계를 맡기면 에이전트형 사용이다.

대화형 사용과 폴더 기반 에이전트형 사용의 비교

챗봇은 결과를 대화 기록에 남기고 폴더 기반 에이전트는 사용자가 정한 파일을 직접 고친다. 그림의 핵심은 결과가 어디에 남는가다.

두 그림의 차이는 결과의 기본 도착점에 있다. 대화형 사용에서 결과는 먼저 대화 기록에 남고 폴더 기반 작업에서는 결과를 사용자가 소유한 파일에 남길 수 있다. 대화 기록은 서비스의 프로젝트와 검색과 기억 기능으로 다시 찾을 수 있지만 로컬 파일처럼 사용자가 정한 폴더 구조와 이름을 그대로 따르지는 않는다. 파일은 폴더 구조와 링크에 따라 쌓이고 다른 도구에서도 그대로 읽을 수 있다. 남는 것의 성격이 다르면 다음에 꺼내 쓰는 방식도 달라진다. 이 구분은 제품별 고정 속성이 아니며 같은 앱에서도 어떤 프로젝트와 도구를 썼는지에 따라 달라진다. 에이전트의 자율성은 허용된 범위 안에서 여러 작업 단계를 스스로 연결한다는 뜻이다. 어느 폴더에서 일할지, 어떤 파일을 고칠 수 있는지, 어떤 외부 서비스에 접근할 수 있는지, 어떤 동작에 사람의 승인을 받아야 하는지는 모두 사람이 정한다. 범위를 정하지 않은 채 목표만 던지면 에이전트는 범위를 스스로 추측하고 추측은 자주 어긋난다. 좋은 지시는 무엇을 하라는 말과 어디까지 하라는 말을 함께 담는다.

요청하는 방식도 달라진다. 챗봇에게는 무엇을 알고 싶은지만 말하면 되지만 에이전트에게는 무엇을 어디에 남길지까지 말해야 한다. “이 논문을 요약해 달라”는 요청은 챗봇에서는 완결되지만 에이전트에서는 절반만 정해진 요청이다. 요약을 화면에 띄울지 파일로 만들지, 만든다면 어느 폴더에 어떤 이름으로 둘지, 기존 문서에 연결할지가 비어 있다. 빠진 조건은 에이전트가 추측해서 정하므로 결과가 매번 조금씩 달라진다. 지시를 한 줄 더 쓰는 수고가 결과를 예측 가능하게 만든다. 에이전트의 행동 경계는 세 겹으로 이루어진다. 첫째는 파일 권한이다. 읽기만 허용된 파일과 고쳐도 되는 파일을 구분하고 원자료 폴더는 대체로 읽기 전용으로 둔다. 둘째는 사람의 승인이다. 파일을 지우거나 여러 파일을 한꺼번에 옮기거나 되돌리기 어려운 변경을 하기 전에는 사람이 확인한다. 셋째는 외부 서비스 권한이다. 메일을 보내거나 공개 저장소에 올리는 동작은 폴더 안에서 끝나지 않으므로 별도로 다룬다. 폴더 접근을 허용한 뒤에도 무엇이 바뀌었는지는 사람이 직접 본다.

이 책이 근거로 삼은 작업 환경에서 그 선은 되돌릴 수 있는가와 밖으로 나가는가 두 가지로 정해져 있다. 파일을 읽고 검색하고 초안을 만들고 형식을 맞추고 빠진 것을 찾는 동작은 승인 없이 맡긴다. 결과가 마음에 들지 않으면 지우고

다시 만들 수 있기 때문이다. 밖으로 나가는 동작은 예외 없이 멈춘다. 메일은 초안을 만들고 파일로 저장하고 내용을 보여주고 기다리고 명시적인 승인을 받은 뒤에 발송한다. 초안을 고쳤으면 고친 것을 다시 보여주고 새 승인을 받는데 앞선 승인은 고치기 전의 초안에 대한 것이기 때문이다. 되돌리기 어려운 동작도 한곳에 놓여서 덮어쓰기 전에 먼저 읽고 무엇을 어떻게 바꿀지 보이고 승인을 받은 뒤에 쓴다. 사람의 화면을 건드리는 동작은 요청받은 턴에만 하는데 배경에서 도는 작업이 팝업이나 권한 대화상자를 띄우면 그것을 누르는 사람은 무엇에 승인하는지 모르는 상태로 누르게 된다.

Markdown이라는 형식

컴퓨터에서 사용하는 파일 형식은 바이너리 파일과 텍스트 파일로 나누어 이해하면 쉽다. 사진, PDF, 워드 문서 같은 바이너리 파일은 전용 프로그램이 내부 구조를 해석해야 사람이 볼 수 있고 일반 텍스트 편집기로 열면 내용이 그대로 읽히지 않는다. 텍스트 파일은 글자가 그대로 저장되므로 사람과 컴퓨터가 모두 읽을 수 있다. Markdown은 텍스트 파일에 제목, 목록, 링크, 표 같은 구조를 간단한 기호로 표시하는 형식이며 확장자는 .md다. 파일을 열면 기호와 문장이 함께 보이므로 전용 프로그램이 없어도 내용을 읽을 수 있다. 특정 회사의 서비스에 묶이지 않아 도구를 바꿔도 문서를 그대로 가져갈 수 있고 무엇이 바뀌었는지도 줄 단위로 비교할 수 있다. 한 문단만 고쳤다면 그 문단만 바뀐 것으로 나타나므로 두 달 전 판단이 언제 어떻게 바뀌었는지 되짚을 수 있다. LLM-Wiki에서는 에이전트가 읽은 자료를 Markdown 문서로 자동 정리한다. 에이전트는 내용에 맞춰 제목을 나누고 목록과 링크를 붙인 뒤 .md 파일로 저장한다. 사용자는 Markdown 문법을 외워서 문서를 직접 만들 필요가 없고 필요할 때 파일을 열어 읽고 문장을 고칠 수 있으면 충분하다.

Markdown의 제목은 문서 안의 주소 역할을 한다. 문서에 제목 구조가 있으면 “이 문서의 한계 절만 고쳐라”나 “결과 절 아래에 이 문단을 넣어라” 같은 지시가 성립한다. 제목이 없는 긴 텍스트에서는 에이전트가 고칠 범위를 정확히 잡기 어렵다. 문서를 통째로 다시 쓰면 원래 있던 내용이 조용히 사라지고 무엇이 사라졌는지 알아채기 어렵다. 링크는 한 문서가 다른 문서를 가리키게 한다. 사람은 링크를 따라 관련 문서를 읽고 에이전트는 어떤 문서를 함께 읽어야 하는지 알 수 있다. 같은 폴더에 파일이 모여 있어도 링크가 없으면 문서 사이의 관계는 기록되지 않는다. 에이전트가 자료를 정리할 때 관련 문서로 가는 링크까지 함께 적어야 다음 작업에서 앞서 만든 내용을 다시 쓸 수 있다. Markdown은 문서의 구조를 표현하며 무엇이 원자료이고 무엇이 해석인지 구분하는 규칙은 LLM-Wiki의 지침 파일이 맡는다. 이 지침 파일과 폴더 구조는 3장의 설치 과정에서 에이전트가 만든다.

폴더가 작업 공간이라는 뜻

에이전트에게 폴더를 지정하면 다음 네 가지가 한꺼번에 정해진다.

1. 보이는 자료 — 어떤 파일을 우선 읽을 수 있는가.
2. 작업 범위 — 어디에 새 파일을 만들고 무엇을 고칠 것인가.
3. 적용 규칙 — 어떤 지침 파일을 따라야 하는가.
4. 지속 기억 — 세션이 끝난 뒤 무엇이 남는가.

네 가지는 함께 움직인다. 첫 항목이 정하는 것은 읽을 수 있는 범위이고 에이전트는 그 범위 안에서 필요하다고 판단한 파일을 골라 읽는다. 그 판단은 지침 파일과 목차 문서가 얼마나 잘 안내하는지에 달려 있고 무엇을 읽었는지 확인하고 싶으면 물어보면 된다. 폴더를 바꾸면 볼 수 있는 자료도, 고칠 수 있는 대상도, 따라야 할 규칙도, 결과가 저장되는 폴더도 같이 바뀐다. 같은 질문을 던져도 어느 폴더에서 던졌느냐에 따라 답이 달라지는 이유가 여기에 있다. 연구 폴더에서 “이 방법의 한계를 정리해 달라”고 하면 그 폴더의 논문들을 근거로 답하고 강의 폴더에서 같은 요청을 하면 수강생 수준과 지난 학기 반응을 근거로 답한다. 둘 다 맞는 답이지만 쓸모는 다르다. 수업, 연구, 글쓰기, 행정은 목적이 다르고 따라서 규칙도 다르다. 연구 폴더에서는 근거 없는 문장을 쓰지 않는 것이 최우선이지만 강의 자료 폴더에서는 정확한 만큼이나 쉬운 설명이 중요하다. 두 규칙을 한 지침 파일에 넣으면 예외 조항이 불기 시작하고 예외가 다섯 개를 넘으면 에이전트도 사람도 규칙을 제대로 적용하지 못한다. 목적이 다르면 작업 공간을 나누는 편이 낫다.

반대로 너무 잘게 쪼개도 문제가 생긴다. 폴더가 열 개로 늘어나면 같은 자료가 여러 곳에 복제되고 어느 폴더의 문서가 최신인지 알 수 없게 된다. 나누는 기준은 규칙이다. 같은 지침으로 일할 수 있는 작업들은 한 폴더에 두고 지침이 서로 충돌하기 시작하면 그때 나눈다. 폴더 사이의 연결은 실제 의존이 있을 때만 만든다. 강의 자료가 연구 논문을 근거로 삼는다면 연결하고 그저 같은 사람이 한다는 이유로는 연결하지 않는다. 두 폴더가 같은 자료를 필요로 하는 경우는 반드시 생긴다. 그때 파일을 양쪽에 복사해 두면 한쪽만 고쳐지는 순간부터 어느 것이 최신인지 알 수 없게 된다. 정본을 한 폴더에 두고 다른 폴더에서는 그 위치를 가리키는 짧은 문서를 둔다. 가리키는 문서에는 경로와 함께 왜 이 자료가 여기서 필요한지를 한 줄 적는다. 경로만 적어 두면 반년 뒤에 그 링크가 왜 있는지 알 수 없고 이유를 알 수 없는 링크는 결국 지워진다.

웹 채팅과 폴더 기반 도구의 차이

두 방식은 기본 작업 경계가 다르다. 웹 대화는 현재 대화와 프로젝트에 올리거나 연결한 자료를 중심으로 시작하고 로컬 프로젝트는 선택한 컴퓨터 폴더와 그 안의 문서를 작업 경계로 삼는다. 웹 프로젝트도 여러 대화에서 파일과 지침을 공유할 수 있으므로 언제나 빈 대화창에서 시작한다고 설명하면 틀린다. 출발점이 다르면 잘하는 일과 놓치는 것도 달라진다. 지금 하려는 일이 설명과 탐색에서 끝나는지, 사용자가 소유한 파일을 여러 번 고치며 축적할 일인지를 먼저 물으면 어느 쪽을 열어야 할지가 정해진다.

관점	웹 채팅	폴더 기반 AI 에이전트
시작점	현재 대화와 프로젝트에 올리거나 연결한 자료	선택한 작업 폴더와 그 안의 지침과 문서
지속 맥락	대화 기록, 프로젝트 자료, 서비스가 제공하는 기억	사용자가 소유한 파일과 명시적인 작업 기록
작업 범위	설명, 아이디어, 일회성 초안	여러 파일을 읽고 고치는 연속 작업
적합한 일	현재 대화가 중심	폴더가 기본 작업 경계
주의점	빠른 질문, 탐색, 가벼운 초안	장기 프로젝트, 반복 작업, 지식 축적
	중요한 결과가 대화에 묻힘	잘못된 범위나 규칙 아래 파일이 바뀔

아직 무엇을 물어야 할지 모르는 단계에서는 웹 대화에서 여러 방향으로 생각을 펼쳐 보는 편이 빠르고 그 과정에서 나온 결론만 위기로 옮기면 된다. 반대로 폴더 기반 에이전트가 만든 초안을 웹 대화에서 다른 관점으로 검토받을 수도 있다. 두 방식은 작업의 다른 단계를 맡는다. 탐색은 앞쪽에, 축적은 뒤쪽에 놓인다. 웹 대화의 결론을 위기로 옮길 때는 결론만 옮기지 않는다. 어떤 질문에서 출발했는지, 무엇을 근거로 그렇게 판단했는지, 확인하지 못한 채 남긴 부분이 무엇인지를 함께 적는다. 결론 한 줄만 옮긴 문장은 한 달 뒤에 읽으면 근거를 알 수 없어 다시 검증해야 한다. 옮기는 데 서너 문장이 더 들지만 그 문장이 다음에 실제로 쓰일 확률이 올라간다. 옮기지 않기로 정하는 것도 판단이므로 좋았던 대화를 전부 남겨야 한다고 생각할 필요는 없다. 외부 도구를 붙이지 않은 대화형 사용에서는 잘못된 답이 파일이나 외부 서비스의 상태를 바로 바꾸지 않는다. 로컬 파일이나 외부 도구를 다루는 에이전트형 사용에서는 잘못된 범위와 규칙이 실제 상태를 바꾼다. 원래 있던 문서가 덮이거나 지워질 수 있고 메일이나 일정처럼 다른 사람이 보는 상태가 바뀔 수도 있다. 그래서 로컬 프로젝트로 넘어갈 때 가장 먼저 확보할 것은 되돌릴 수 있는 상태와 승인 경계다. 버전 기록을 남기는 동기화 서비스든 사본이든, 잘못되었을 때 되돌릴 방법이 있어야 마음 놓고 파일 권한을 줄 수 있다.

AI 에이전트 도구 고르기

강의에서 자주 받는 질문이 있다. Codex와 Claude 가운데 무엇을 쓰느냐는 질문이다. 나는 Codex를 추천한다. 2026년 8월까지 LLM-Wiki를 운영하며 매긴 순위는 Codex, Claude Cowork 또는 Claude Code 순이다. Claude를 고른다면 컴퓨터에 익숙하지 않은 독자에게는 폴더를 연결하고 자연어로 작업을 맡길 수 있는 Claude Cowork가 편하다. 터미널에 익숙한 독자는 Claude Code로 같은 위키를 운영할 수 있다. Gemini CLI는 순위에서 제외한다. 폴더를 지정하고 지침을 읽고 문서를 만들고 고치고 링크와 빠진 항목을 검사하는 긴 작업에서 Gemini CLI는 규칙을 자주 빠뜨렸다. 검증을 마치기 전에 완료했다고 보고한 경우도 잦았다. 출력은 빨랐지만 결과를 처음부터 다시 확인해야 했다. 이 책에서 요구하는 AI 에이전트 도구로는 부적합하다는 것이 내 판단이다.

추천 순위	도구	이 책에서의 판단
1	Codex	기본 추천. 긴 폴더 작업과 규칙 준수에 가장 안정적이었다.
2	Claude Cowork 또는 Claude Code	폴더를 연결해 이 책의 실습을 수행할 수 있는 대안이다.
제외	Gemini CLI	긴 작업의 규칙 준수와 완료 검증이 부족해 권하지 않는다.

Codex 안에서는 복잡한 판단과 대규모 교정에 Sol을 먼저 고른다. 일상적인 논문 ingest와 위키 페이지 작성에는 Terra로도 충분히 좋은 품질이 나온다. 같은 논문 28편을 Sol과 Terra Extra High로 각각 처리해 비교했을 때 두 모델 모두

논문 신원, 사실, 문서 구조에서 중대한 오류가 없었다. Terra는 근거 추적성, 방법의 구체성, 수치의 범위, 한계 처리를 더 자세히 기록했고 Sol은 더 간결했다. 논문 28편의 비교만으로 모든 작업에서 어느 모델이 나은지 단정할 수는 없다. 다만 Terra를 일상적인 위키 생성에 써도 된다고 판단할 근거는 얻었다.

Gemini CLI를 써 본 인상은 이 장면과 닮았다. 에이전트 작업에서는 빠른 출력보다 지침 준수와 결과 검증이 중요하다. 출처: 링크

새 도구를 검토할 때는 빈 폴더에 같은 지침과 같은 자료를 넣고 실제 결과를 비교한다. 파일을 올바른 곳에 남겼는지, 지침을 작업 끝까지 지켰는지, 요청하지 않은 변경을 만들지 않았는지를 보면 LLM-Wiki에 쓸 만한지 판단할 수 있다. 모델과 앱은 계속 바뀌므로 이 순위도 같은 시험을 다시 돌려 갱신한다.

이 책의 LLM-Wiki를 지속해서 운영하려면 월 20달러 이상의 유료 구독이 필요하다. 무료 등급으로는 논문을 여러 편 처리하고 폴더의 문서를 반복해서 고치는 작업량을 감당하기 어렵다. Codex를 쓸 사람은 ChatGPT 유료 구독을, Claude를 쓸 사람은 Claude 유료 구독을 준비한다.

Codex, Claude Code, ChatGPT가 속한 카테고리

제품 이름을 외우는 일은 오래 가지 않는다. 이름은 바뀌고 기능은 늘어나고 화면은 계절마다 달라진다. 오래 가는 것은 그때 선택한 작업 방식이다. 웹 대화 환경은 브라우저에서 질문하고 파일이나 연결된 자료를 쓰며 컴퓨터의 로컬 폴더를 직접 작업 경계로 삼지 않는다. 로컬 프로젝트 환경은 컴퓨터의 폴더를 붙이고 그 안의 파일을 직접 읽고 고친다. 외부 도구 연결 환경은 대화나 로컬 프로젝트에 메일, 일정, 문서 서비스와 같은 도구를 더한다. 한 제품이 세 환경을 오갈 수도 있다. ChatGPT 웹의 프로젝트는 업로드한 파일과 연결된 자료를 여러 대화가 함께 쓰지만 컴퓨터 폴더에 직접 접근하지 않는다. ChatGPT 데스크톱 앱의 로컬 프로젝트는 하나 이상의 컴퓨터 폴더를 붙일 수 있고 기본 폴더를 기준으로 Codex가 AGENTS.md와 설정을 찾는다. Codex 명령줄과 편집기 확장도 현재 작업 폴더를 프로젝트 경계로 삼는다. 따라서 ChatGPT를 웹 대화 도구로, Codex를 별도의 폴더 도구로 고정해 외우면 현재 제품 구조를 잘못 이해하게 된다.

혼합 환경은 편리한 만큼 경계가 흐리다. 대화창에서 파일을 다루는 것처럼 보이지만 결과가 서비스 안에만 남고 사용자의 폴더에는 아무것도 생기지 않는 경우가 있다. 반대로 대화처럼 보이는 요청 하나가 실제 파일 여러 개를 고치는 경우도 있다. 두 경우를 구분하는 방법은 작업이 끝난 뒤 폴더를 직접 열어 보는 것이다. 무엇이 새로 생겼고 무엇이 바뀌었는지 눈으로 확인하는 습관이 불기 전까지는, 화면에 나온 설명만으로 결과를 판단하지 않는 편이 안전하다. 새 도구를 검토할 때 세 가지를 확인하면 그 도구가 어느 카테고리인지 알 수 있다. 내 파일을 실제로 읽고 쓰는가, 어떤 지침 파일을 읽는가, 작업 결과가 세션 밖에 남는가. 세 질문에 모두 답할 수 있으면 그 도구로 위키를 운영할 수 있고 하나라도 답할 수 없으면 그 부분은 사람이 손으로 메워야 한다.

같은 회사의 두 도구를 고르는 기준

한 회사가 대화 중심 화면과 명령줄 도구를 함께 내놓는 경우가 있고 둘 중 어느 것으로 시작할지가 첫 질문이 된다. Claude Cowork와 Claude Code가 그런 사용 방식의 차이를 보여 주며 ChatGPT 데스크톱 앱의 로컬 프로젝트와 Codex 명령줄도 비슷한 선택지를 제공한다. 대화 중심 화면에서는 폴더를 연결한 뒤 자연어로 작업을 맡긴다. 명령줄에서는 현재 폴더를 출발점으로 명령과 파일 작업을 함께 다룬다. LLM-Wiki를 운영하는 데 필요한 기본 동작은 폴더를 읽고 문서를 만들고 고치고 링크를 맞추고 빠진 것을 찾는 일이다. 이 범위는 대화 중심 화면으로도 덮을 수 있다. 명령줄은 명령을 여러 개 이어 붙이거나 스크립트를 만들고 돌리거나 수백 개의 파일을 정해진 검사로 훑을 때 편하다. 파트 1의 실습은 대화 중심 화면에서 시작하고 검색 색인과 배치가 필요해질 때 명령줄을 더해도 된다. 다만 도구마다 자동으로 찾는 지침 파일 이름과 우선순위가 같지는 않다. 공통 규칙의 정본을 하나 두고 각 도구가 실제로 읽는 지침 파일에서 그 규칙을 일관되게 유지해야 도구를 옮겨도 같은 경계를 쓸 수 있다.

여러 대의 컴퓨터에서 같은 위키를 쓰는 방법

연구실과 집, 노트북과 데스크톱에서 같은 위키를 열고 싶어지는 시점이 곧 온다. 이 책이 근거로 삼은 위키는 두 대의 Mac이 Dropbox로 같은 폴더를 공유하는 방식으로 운영된다. 파일 동기화 서비스를 쓰는 이유는 별도의 절차 없이 어느 기계에서 열어도 같은 상태가 보이기 때문이다. 서비스를 고를 때 기준이 하나 있는데 수만 개의 작은 파일을 다룰 수 있는가다. 논문 만 편 규모의 위키에는 PDF와 Markdown 문서가 삼만 개 넘게 들어가고 이 규모에서 동기화가 밀리거나

충돌 사본을 만들기 시작하면 어느 파일이 최신인지 알 수 없게 된다. 실제로 운영해 보면 Dropbox가 이 조건에서 안정적이었다. OneDrive와 Google Drive는 파일 수가 많아질 때 동기화가 지연되거나 사본이 생기는 문제가 반복되어 이 용도에는 맞지 않았다.

여러 기계를 쓰면 따로 정해야 할 것이 몇 가지 생긴다. 첫째는 작업 로그를 기계마다 나누는 일이다. 같은 날 두 기계에서 작업하면 로그 파일 하나에 두 흐름이 섞이므로 파일 이름에 기계 이름을 넣어 갈라 둔다. 지금 어느 기계인지는 기억으로 정하지 않고 명령으로 확인하는데 같은 날 두 기계에서 작업이 돌아 명령을 실행해야만 구분된 적이 실제로 있었다. 둘째는 동기화 폴더 밖에 두어야 할 것을 정하는 일이다. 검색 색인과 실행 환경은 기계마다 따로 만들어지고 크기가 크며 자주 바뀌므로 동기화 대상에서 뺀다. 두 기계가 같은 색인 파일을 동시에 다시 만들면 어느 쪽도 온전하지 않은 상태가 된다. 셋째는 세션을 시작할 때 지침 파일을 다시 읽는 습관이다. 다른 기계에서 규칙을 고쳤을 수 있으므로 이전 세션의 기억에 의존하지 않는다.

폴더를 정하고 새 세션을 여는 법

이 절에서 사람이 직접 할 일은 데스크톱 앱을 설치하고 로그인한 뒤 새 세션에 작업 폴더를 연결하는 것까지다. Codex를 쓰려면 ChatGPT 데스크톱 앱을 설치한다. Claude를 쓰려면 Claude 데스크톱 앱을 설치한다. Codex와 Claude 가운데 사용할 앱 하나를 골라 같은 이름의 폴더를 만들면 된다. 폴더 이름은 이 책 전체에서 `llm-wiki`로 통일한다. 폴더를 만든 뒤에는 새 세션을 열 때마다 그 폴더가 선택되어 있는지만 확인한다. 앱 판에 따라 `New`, `New chat`, `New project`, `Open folder`처럼 버튼 이름이 조금 달라질 수 있다. LLM-Wiki 문서와 하위 폴더는 3장의 Gist 설치 과정에서 만들므로 여기서는 빈 `llm-wiki` 폴더를 선택한 상태에서 멈춘다.

Codex에서 첫 세션 만들기

1. ChatGPT 데스크톱 앱 공식 내려받기에서 운영체제에 맞는 앱을 받아 설치한다.
2. 앱을 실행하고 LLM-Wiki에 사용할 ChatGPT 계정으로 로그인한다.
3. 왼쪽의 `New chat`을 눌러 Codex의 새 세션을 연다.
4. 입력창 위에 표시된 현재 프로젝트나 폴더 이름을 누른다.
5. 처음 만드는 경우 `New project`를 선택하고 `llm-wiki` 폴더를 만든다. 이미 폴더가 있으면 목록에서 고르거나 `Open folder`로 찾는다.
6. 입력창 위에 `llm-wiki`가 표시되면 첫 세션의 폴더 지정이 끝난다.

왼쪽에서 새 세션을 열고 입력창 위의 프로젝트 이름을 누르면 폴더를 선택할 수 있다. 화면에는 저자가 작업 중인 폴더가 보이지만 처음 실습하는 독자는 `New project`에서 `llm-wiki`를 만든다.

Claude에서 첫 세션 만들기

1. Claude 데스크톱 앱 공식 내려받기에서 운영체제에 맞는 앱을 받아 설치한다.
2. 앱을 실행하고 LLM-Wiki에 사용할 Claude 계정으로 로그인한다.
3. 왼쪽의 `Code`를 선택하고 `New`를 눌러 새 세션을 연다.
4. 입력창 아래에 표시된 현재 폴더 이름을 누른다.
5. 처음 만드는 경우 `New project`를 선택해 `llm-wiki` 폴더를 만든다. 이미 폴더가 있으면 목록에서 고르거나 `Open folder`로 찾는다.
6. 입력창 아래에 `llm-wiki`가 표시되면 첫 세션의 폴더 지정이 끝난다.

Claude 데스크톱 앱에서는 `New`로 세션을 열고 입력창 아래의 폴더 이름을 눌러 작업 폴더를 고른다. 새로 시작할 때는 `New project`에서 `llm-wiki`를 만든다.

작업이 달라질 때마다 새 세션을 연다. 논문을 넣는 일, 특정 질문을 조사하는 일, 여러 문서를 고치는 일을 각각 새 세션으로 시작해도 같은 `llm-wiki` 폴더를 선택하면 기존 문서를 함께 읽을 수 있다. 로컬 프로젝트에서 `llm-wiki`를 선택해 두면 새 세션도 같은 폴더에서 시작할 수 있다. 세션마다 대화 기록은 따로 남지만 파일은 같은 폴더를 사용한다. 같은 작업을 계속할 때는 이전 세션을 다시 열어도 되고 작업 주제가 바뀌었으면 새 세션을 열면 된다. 세션이 길어져 앞부분의 설명을 다시 찾기 어려워졌을 때도 새 세션에서 같은 폴더를 선택하면 된다. 중요한 판단과 진행 상황은 대화 기록에만 두지 않고 에이전트가 폴더의 Markdown 문서와 작업 로그에 적게 한다. 새 세션의 에이전트는 폴더에 남은 문서를 읽고 이전 작업의 결과를 이어 간다. 모든 세션이 같은 `llm-wiki` 폴더를 읽도록 지정하면 작업 기록이 세션이 바뀐 뒤에도 이어진다.

챗지피티 데스크탑 앱 (Codex) 실행 화면

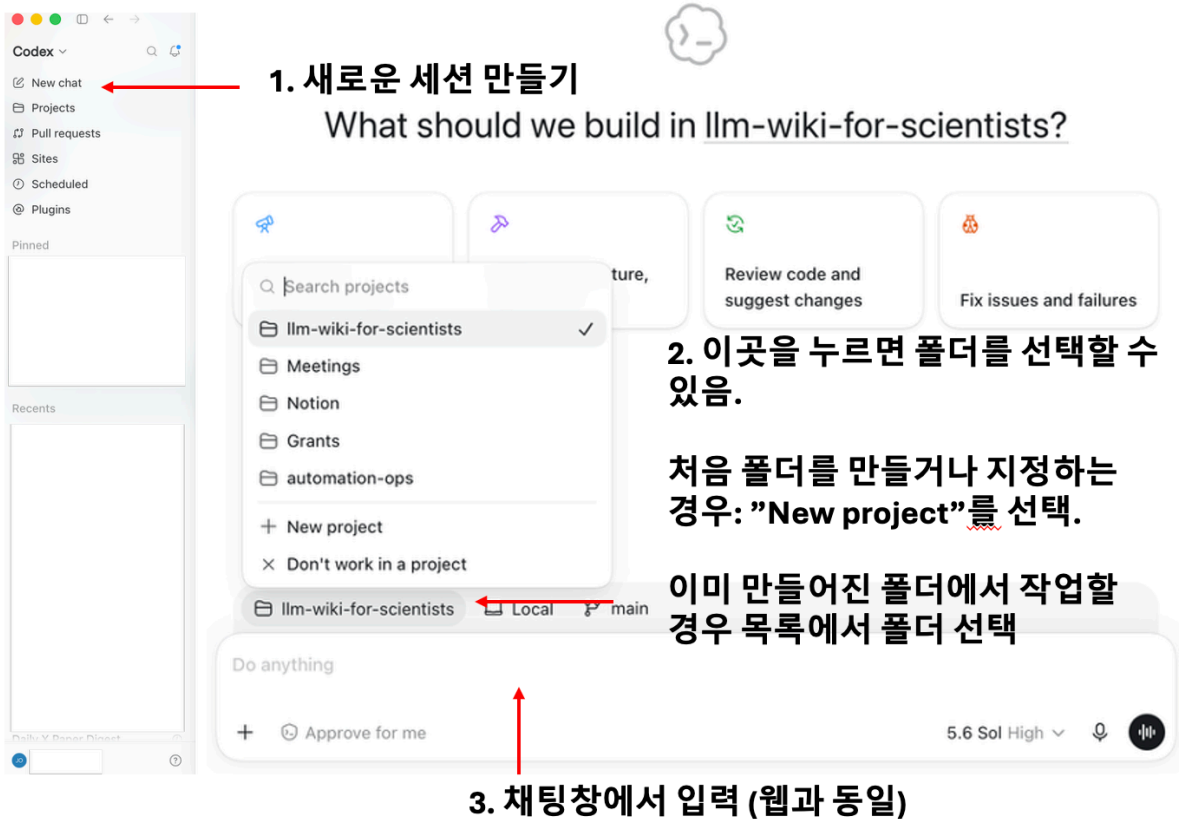


Figure 1: ChatGPT 데스크톱 앱의 Codex에서 새 세션과 작업 폴더를 선택하는 화면

클로드 앱 (Codex) 실행 화면

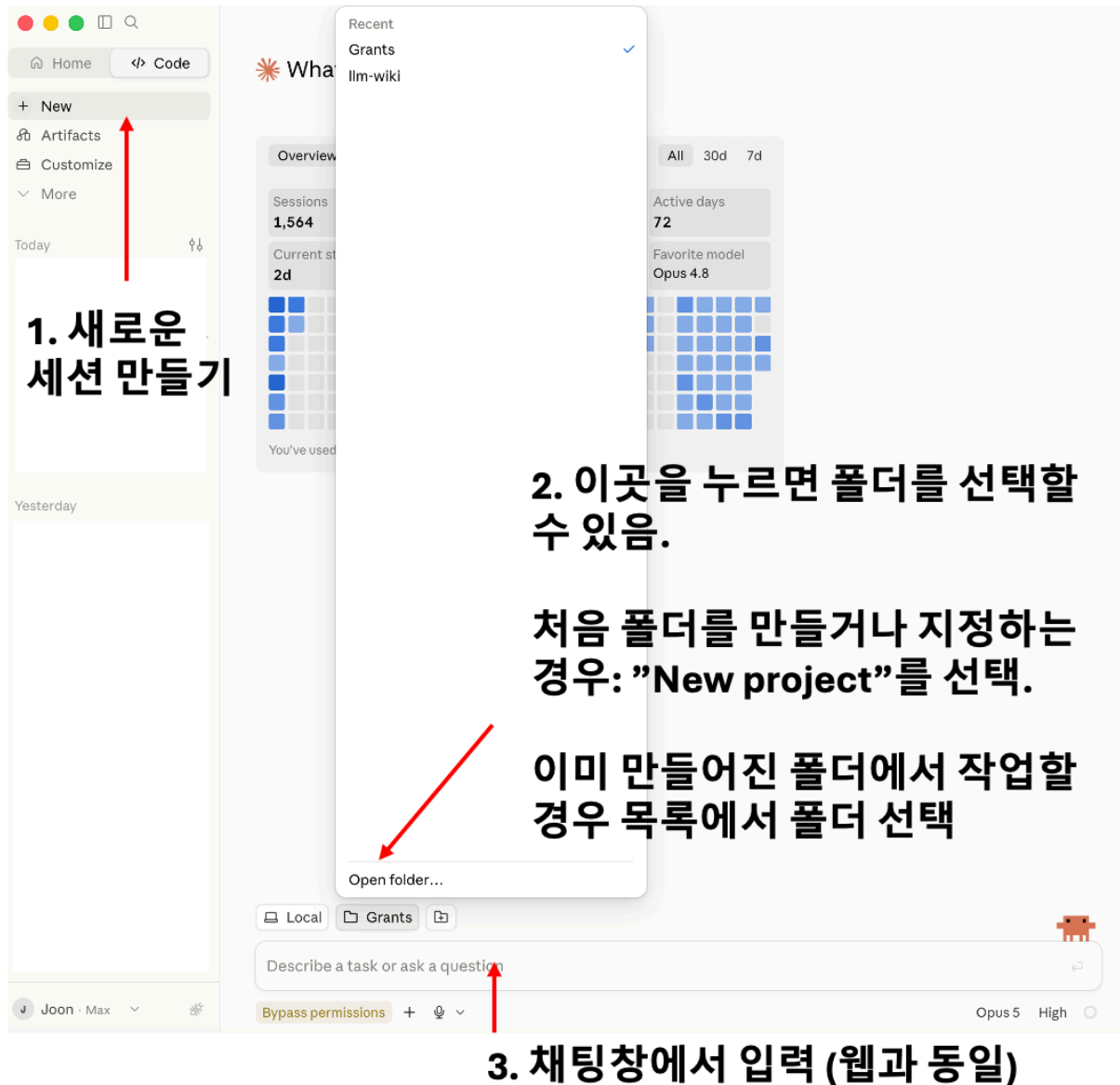
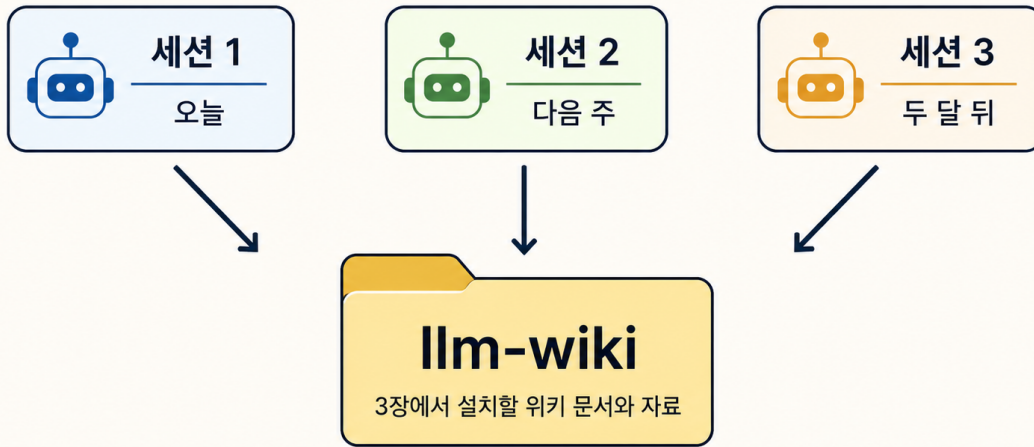


Figure 2: Claude 데스크톱 앱에서 새 세션과 작업 폴더를 선택하는 화면

세션이 바뀌어도 같은 폴더를 사용한다



각 세션은 폴더에서 읽고 폴더에 남긴다.
세션끼리는 직접 이어지지 않는다.

Figure 3: 세 개의 세션이 같은 llm-wiki 폴더를 사용하는 구조

세션의 대화 기록은 서로 분리되지만 세 세션 모두 같은 *llm-wiki* 폴더를 읽고 파일을 남긴다.

다른 폴더에서 세션을 시작하면 목적과 규칙, 자료의 우선순위가 달라진다. 같은 도구, 같은 사람, 같은 질문이어도 결과가 달라지는 것이 정상이다. 여러 폴더의 자료가 함께 필요한 작업에서는 최종 산출물을 소유하는 폴더를 주 작업 공간으로 삼고 나머지는 근거를 제공하는 보조 공간으로 다룬다. 논문 초고를 쓴다면 원고 폴더가 주 작업 공간이고 논문 위키는 보조 공간이다. 주 작업 공간을 정하지 않으면 결과물이 두 폴더에 나뉘어 남고 어느 쪽이 정본인지 나중에 알 수 없게 된다. 다음 세션으로 넘기는 작업 기록은 길 필요가 없지만 형식은 정해 두는 편이 낫다. 무엇을 했는지, 무엇이 남았는지, 다음에 무엇부터 할지를 각각 한 문장으로 적는다. 세 문장을 쓸 때는 두 달 뒤에 이 폴더를 처음 여는 사람이 읽는다고 생각하고 쓴다. “이어서 진행”이나 “검토 필요” 같은 문장은 쓴 사람만 알아볼 수 있고 쓴 사람도 두 달 뒤에는 알아보지 못한다. 어느 문서의 어느 부분을 어떻게 할 차례인지 적어 두면 다음 세션이 첫 십 분을 상황 파악에 쓰지 않는다. 앱을 설치하고 로그인한 뒤 llm-wiki 폴더를 선택하면 도구 쪽 준비가 끝난다.

3장. Karpathy의 제안에서 나의 연구 위키까지

대학원 세미나를 새로 맡은 사람이 읽기 자료 스무 편을 폴더에 넣었다고 하자. 첫 주에 학생이 먼저 이해해야 할 개념을 물었고 쓸 만한 답을 받았다. 그 답에는 자료 세 편을 가로지르는 연결이 하나 들어 있었는데 어느 논문에도 문장으로 적혀 있지 않고 스무 편을 함께 읽으면서 생긴 연결이었다. 한 달 뒤 자료들 사이에 방법이 어떻게 이어지는지 물었더니 도구는 스무 편을 처음부터 다시 훑었다. 두 답은 비슷했지만 첫 답에 있던 그 연결이 두 번째에는 없었다. 같은 자료로 두 번 계산했는데 남은 것은 대화 두 개뿐이고 대화는 다음 학기에 다시 열리지 않는다.

매번 다시 계산하는 검색

문서를 모아 두고 질문에 답하게 하는 방식을 RAG라고 부르고 그 방식은 대체로 같은 순서로 움직인다. 질문이 들어오면 관련 있어 보이는 조각을 골라 함께 읽고 답을 만든다. 그 사이에 일어난 판단은 기록되지 않는다. 어느 두 문단을 나란히 놓았는지, 어느 대목을 근거로 삼고 어느 대목을 버렸는지가 대화 기록 밖으로 나가지 않는다. 문서 한 편 안에서 끝나는 질문이라면 비용이 크지 않다. 세 논문의 결과가 왜 어긋나는지 같은 질문은 다르다. 그 답은 세 논문 어디에도 문장으로

있지 않고 세 편을 함께 읽는 동안 만들어지므로 저장되지 않으면 다음에 다시 만들어야 한다. 다시 만든 답이 지난번과 같다는 보장도 없어서 두 답이 어긋났을 때 어느 쪽이 맞는지 판정할 근거조차 남지 않는다.

카파시는 2026년 4월에 질문할 때마다 문서를 다시 읽는 문제를 줄이기 위해 LLM-Wiki를 제안했다. 정본은 Karpathy의 Gist이고 Gist 한 편에 llm-wiki.md 파일 하나가 들어 있다. 분량은 짧고 코드도 없으며 읽는 사람이 자기 자료에 맞춰 채워 넣도록 의도적으로 추상적으로 쓰였다. 요지는 질문마다 답을 합성하는 대신 LLM이 미리 Markdown 위키를 만들어 갱신하게 하라는 것이다. 한 번 얻은 요약과 교차 연결과 모순 기록이 문서로 남으므로 다음 질문은 그 문서 위에서 시작한다. 원문이 쓴 표현은 “The wiki is a persistent, compounding artifact.”다. 저장되는 것은 자료만이 아니고 자료에서 뽑아낸 판단과 그 판단이 어느 자료에서 나왔는지까지 함께 문서로 남는다. 조각을 찾아 답하는 방식은 자료가 늘어도 출발점이 한곳에 있지만 위키를 유지하면 질문을 던질 때마다 출발점이 조금씩 올라간다. 대신 선행 비용이 들고 한 번 잘못 적힌 판단은 누가 지울 때까지 계속 재사용된다. 뒤의 성질이 이 방식의 진짜 위험이어서 이 책의 나머지가 다루는 규칙과 점검은 대부분 그 위험을 관리하는 장치다.

Karpathy 원형의 세 층

원형은 자료를 raw sources, the wiki, the schema 세 층으로 나눈다. 나누는 층은 주제도 형식도 파일 확장자도 아니고 누가 그 층을 고칠 수 있는가다. 같은 폴더에 있고 같은 Markdown이어도 사람만 넣는 문서와 LLM이 유지하는 문서는 다르게 취급되어야 한다. 권한을 나누지 않으면 원문에 있던 문장과 나중에 붙은 해석을 몇 달 뒤에 구분할 수 없고 구분할 수 없으면 어느 쪽도 신뢰할 수 없다. 첫 층인 raw sources는 LLM이 읽되 수정하지 않는 불변 자료다. 논문 PDF, 데이터, 이미지처럼 밖에서 들어온 것이 여기 놓이고 오탈자가 보여도 고치지 않는다. 고치는 순간 되짚어 갈 바닥이 사라지기 때문이다. 어떤 서술이 의심스러울 때 최종적으로 열어 보는 것이 이 층이므로 이 층이 손타지 않았다는 확신이 나머지 전부를 떠받친다.

Wiki 층을 서술할 때 원문은 소유라는 말을 쓴다. “The LLM owns this layer entirely. It creates pages, updates them when new sources arrive, maintains cross-references.” 사람이 손대지 못한다는 선언으로 읽으면 오해다. 문서가 수백 개로 늘어난 뒤에 새 자료가 들어올 때마다 관련 페이지를 찾아 고치고 링크를 맞추는 일을 사람의 성실함에 맡기면 유지되지 않는다는 인정에 가깝다. 사람이 하는 일은 읽고 검토하고 틀린 곳을 지적하는 쪽으로 옮겨 간다. 세 번째 층인 the schema는 CLAUDE.md 같은 설정 문서이고 폴더 구조와 명명법과 운영 규칙이 여기 들어간다. 원문은 이 층을 두고 “You and the LLM co-evolve this over time as you figure out what works for your domain.”이라고 쓴다. 처음부터 완성된 규칙을 쓸 수 있는 사람은 없으므로 설치 시점의 규칙 문서는 첫 번째 판이고 이후의 모든 판은 무언가 어긋난 뒤에 나온다.

세 층 위에서 도는 동작은 셋이다. Ingest는 새 자료를 읽어 위키와 색인과 로그를 갱신하고 query는 관련 페이지에서 답을 만들어 좋은 답을 다시 위키에 되돌린다. Lint는 모순되는 서술, 낡은 주장, 고립된 페이지, 빠진 상호 참조를 찾는다. 앞의 둘은 위키를 키우고 마지막 하나는 커지면서 생긴 손상을 찾는다. Lint가 무엇을 잡아야 하는지는 이 책이 쓰는 설치 템플릿 안에 이미 예가 있다. 설치 Gist에는 카파시의 원형을 가리키는 링크가 세 군데 있고 2026년 8월 기준으로 세 군데 모두 같은 주소 하나를 가리킨다. 지침 파일 템플릿에 한 번, 설치 문서의 본문과 하단 출처 표시에 각각 한 번씩 들어 있다. 열어 보면 LLM-Wiki와 무관한 다른 Gist다. 링크가 깨져 있으면 눌러 본 사람이 바로 알아차리지만 이 주소는 멀쩡히 열리고 문서만 다르므로 목록을 훑는 방식으로는 발견되지 않고 열어서 내용을 확인해야 잡힌다. 이 책은 정본으로 442a6bf로 시작하는 Gist를 쓰고 세 군데의 링크는 고쳐야 할 것으로 남겨 둔다. 눈에 띄는 하나만 고치면 나머지 둘이 그대로 남는다는 것이 이 사례에서 함께 배울 점이다. 고칠 곳을 발견한 채로 두는 것과 발견조차 못 하는 것은 다른 상태이므로 발견한 것은 로그에 적어 다음 lint가 같은 곳을 다시 열게 한다. 자기 위키에서도 같은 종류의 어긋남이 링크와 인용에 쌓이므로 lint를 눈으로 훑는 작업으로 두면 이런 것만 골라서 남는다.

연구용 구현은 원형에 설계 원칙을 더했고 그 내용은 지침 파일 맨 앞에 있다. 첫 항목은 자료가 흐르는 방향이다. 고칠 수 없는 PDF에서 요약 문서로, 요약 문서에서 최종 페이지로 한 방향으로만 내려가고 거슬러 올라가는 경로는 두지 않는다. 두 번째는 위키를 하나만 둔다는 결정이고 전 분야를 한 위키에 담은 뒤 카테고리 나눈다. 분야마다 저장소를 따로 세우면 그 경계에 걸친 논문을 어디에 둘지가 매번 결정 사항이 되고 두 저장소에 같은 논문이 두 번 들어가기 시작한다. 세 번째는 언어다. 대화는 한국어로 해도 위키에 적히는 문장은 영어로만 쓰는데 그 문장들이 결국 논문 원고와 검색어로 다시 쓰이기 때문이다. 한국어로 적어 둔 판단은 인용할 때마다 번역해야 하고 그 과정에서 원문의 표현과 어긋난다. 네 번째는 Obsidian 호환을 유지한다는 것이고 표준 Markdown과 위키링크만 쓰고 특정 도구에만 있는 문법을 쓰지 않는다는 뜻이다. 다섯 번째는 질의에 대한 답을 종합 페이지로 저장하는 것이 지식이 복리로 쌓이는 방식이라는 선언이다.

연구 논문용 네 층 확장

카파시의 원형은 에이전트가 관리하는 문서를 wiki 한 층에 둔다. 연구 논문을 다루려면 이 층에 세 종류의 Markdown 문서가 필요하다. Source note에는 논문을 끝까지 읽고 확인한 세부 내용을 적는다. 논문 페이지에는 검색과 비교에 쓸 핵심 내용을 적는다. 종합 문서에는 여러 논문을 함께 읽고 내린 판단을 적는다. 세 가지를 한 문서에 모두 넣으면 문서가 길어지고 논문 여러 편을 나란히 비교하기도 어렵다. 연구자용 LLM-Wiki는 이 세 종류를 나누고 원본 PDF를 별도로 보관해 네 층으로 운영한다. 지침 파일에는 각 문서에 들어갈 항목과 문서 사이를 연결하는 방법을 적는다. 원본 PDF는 처음 받은 상태로 보관한다. 자료는 원본 PDF에서 source note, 논문 페이지, 종합 문서의 순서로 정리된다.

Original PDF → sources/*.md → wiki/{category}/*.md
→ wiki/overviews/ + wiki/concepts/ + wiki/questions/

문서	담는 내용	주로 쓰는 때
Source note	원문에서 확인한 방법, 수치, 조건, 한계	세부 근거를 확인할 때
논문 페이지	핵심 주장, 방법, 관련 문서 링크	논문을 검색하고 여러 편을 비교할 때
Overview, concept, question	여러 논문을 함께 읽고 내린 현재 판단	새 논문이 들어와 기존 판단을 고칠 때

Source note와 논문 페이지에는 저자, 연도, 제목으로 만든 같은 파일 이름을 쓴다. 이 공통 파일 이름을 stem이라고 부른다. 논문 페이지에서 source note를 열고, source note에서 원본 PDF를 열 수 있으므로 짧은 설명에서 세부 근거까지 차례로 확인할 수 있다. 여러 논문을 함께 읽고 내린 판단은 overview, concept, question에 적는다. Overview는 한 주제에서 현재까지 알려진 내용을 정리한다. Concept은 여러 논문에 반복해서 등장하는 방법이나 개념을 한 번만 설명한다. Question은 아직 근거가 부족해 답하지 못한 질문을 기록한다. 새 논문이 기존 판단을 바꾸면 관련 종합 문서의 본문을 고친다. Source note와 논문 페이지를 만들었어도 종합 문서와 연결하지 않았다면 ingest는 끝나지 않은 상태다. 이렇게 연결되지 않은 한 쌍을 synthesis-orphan이라고 부른다. Synthesis-orphan은 검색에는 나오지만 다른 논문과 어떤 관계가 있는지 적혀 있지 않다. 이 상태가 쌓이면 질문을 받을 때마다 개별 논문을 다시 비교해야 한다. 적절한 종합 문서가 없다면 작업 로그에 빠진 문서를 기록하고 같은 주제가 반복될 때 새 종합 문서를 만든다.

문서를 나누어 두면 어느 작업이 밀렸는지 숫자로 확인할 수 있다. 이 책이 근거로 삼은 연구용 위키에는 2026년 8월 기준으로 원문 PDF 16,127건과 source note 15,995건이 있다. 차이 132건은 PDF를 보관했지만 아직 읽고 정리하지 않은 논문이다. 이 수가 몇 달 동안 줄지 않으면 새 논문을 넣는 속도가 읽고 정리하는 속도보다 빠르다는 뜻이다. 논문 페이지는 17,591건이고 50개 카테고리에 나뉘어 있다. 종합 문서는 overview 590건, concept 408건, question 550건이다. 종합 문서 한 편에는 여러 논문의 판단이 함께 들어가므로 논문 수와 같은 속도로 늘어날 필요가 없다. 종합 문서가 논문마다 하나씩 생기고 있다면 여러 논문을 연결하지 못하고 개별 요약만 반복하고 있는지 확인해야 한다. 카테고리별로 문서 수와 synthesis-orphan 수를 세면 다음에 읽을 논문과 보충할 종합 문서를 고를 수 있다.

연구자용 LLM-Wiki의 아홉 가지 운영 원칙

층을 나눠 놓아도 답을 만들고 논문을 받아들이는 규칙이 없으면 위키의 품질은 곧 흔들린다. 위키에 없는 내용을 에이전트가 다른 데서 채우면 그럴듯한 답은 나오지만 연구자가 보유한 근거와 답의 관계를 확인할 수 없다. 설치 Gist의 지침 파일에는 이를 막는 아홉 가지 운영 원칙이 적혀 있다. 첫 네 원칙은 질문에 답할 때 근거를 어디에서 가져올지 정한다. 뒤의 다섯 원칙은 어떤 논문을 위키에 받아들이고 언제 작업을 완료로 판정할지 정한다. 아홉 원칙은 설치 직후부터 모두 적용한다.

답의 근거를 지키는 네 가지 원칙

1. 임의의 웹 검색으로 빈틈을 채우지 않는다. 웹은 사용자가 명시적으로 요청했을 때만 쓴다.
2. 먼저 위키에서 답한다. sources/와 wiki/에 실제로 들어 있는 논문을 근거로 삼는다.
3. 위키가 부족하면 원본 PDF를 다시 읽어 보충하고 다음에 되돌아가지 않도록 위키를 갱신한다.
4. 해당 논문이 없으면 없다고 말한다. 다른 자료로 조용히 대체하지 않고 PDF를 요청한다.

웹 검색을 기본적으로 달아 두는 이유는 근거가 섞이는 것을 막기 위해서다. 웹에서 온 문장과 소유한 논문에서 온 문장이 한 답 안에 섞이면 읽는 사람도 작성자도 몇 주 뒤에 둘을 구분하지 못한다. 위키에 없는 내용을 검색 결과로 채우기 시작하면 모델은 아직 확인하지 않은 내용을 자연스러운 문장으로 이어 붙인다. 나는 이 지점을 환각의 출발점으로 본다. 웹은 사용자가 명시적으로 요청한 작업에서만 열고 가져온 내용의 출처를 그때 기록한다. 두 번째 원칙은 sources/와 wiki/에서 먼저 답하도록 순서를 정한다. 검색된 내용이 부족하면 세 번째 원칙에 따라 원본 PDF를 다시 읽고 확인한 세부 내용을 위키에 보충한다. 위키를 고치지 않으면 같은 부족을 다음 질문에서 다시 만난다. 네 번째 원칙은 필요한 논문이 없을 때 없다고 말하고 PDF를 요청하게 한다. 초록 몇 줄이나 다른 웹 자료로 빈 답을 채우면 위키가 보유한 근거의 경계가 사라진다.

없다는 답을 받아들이 수 있는지가 이 방식이 작동하는지를 가른다. 논문이 없다는 답은 무엇을 구해야 하는지 알려 주므로 다음 동작으로 이어진다. 근거가 비어 있는 답을 다른 자료로 채우면 부족하다는 사실이 가려지고 나중에 원문을 열어 보기 전까지 알아채기 어렵다.

위키의 품질을 지키는 다섯 가지 원칙

5. 논문이 종합 층과 연결되어야 ingest가 끝난다. 새 논문 페이지를 가장 관련 있는 overview나 concept와 연결하고, 종합 문서의 본문에도 이 논문이 기존 판단을 어떻게 바꾸는지 반영한다.
6. Ingest에 등급을 두지 않고 placeholder 페이지를 만들지 않는다. 위키에 받아들인 모든 논문은 원문을 읽고 구체적인 방법, 결과, 한계, 종합 문서와의 연결을 갖춰야 한다. 초록, metadata, 참고문헌의 언급, 파일명만으로 페이지를 만들지 않는다. 정확한 원문이 완전한 페이지를 뒷받침하지 못하면 해당 논문을 넣지 않고 그날 작업 로그에 미보유 또는 미완료 항목으로 남긴다.
7. 전수 요청은 전체를 처리한 뒤 보고한다. 시작할 때 전체 건수를 분모로 확정하고 끝날 때 완료, 제외, 미검사의 합이 분모와 같은지 확인한다. 추출 실패, 빈 텍스트, 시간 초과는 통과로 세지 않고 미검사로 남긴다. 일부 표본만 확인한 결과를 전수 결과처럼 보고하지 않으며 대상이 많다는 이유로 범위를 줄이지 않는다.
8. 비공개 자료는 wiki/에 넣지 않는다. 미공개 원고, 심사 중인 초안, 엠바고가 걸린 자료는 페이지, 제목, tag, 출판 논문 페이지의 내부 원고 관련 절을 포함해 어떤 형태로도 지식 층에 남기지 않는다. 이런 기록은 agenda/에 두고 DOI가 있는 출판 논문은 일반 위키 자료로 다룬다.
9. 논문의 정본은 papers/에 보관한 정확한 PDF다. 출판사 HTML이나 브라우저에서 저장한 텍스트를 원문 대신 쓰지 않는다. 정확한 PDF를 구하지 못하면 품질이 낮은 사본으로 ingest하지 않고 미보유 PDF 목록에 남긴다.

뒤의 다섯 원칙은 모두 실제 문제가 생긴 뒤에 추가되었다. 어느 규칙이 어느 사고에서 나왔는지는 로그를 거슬러 올라가면 대개 찾을 수 있다. 그래서 규칙 목록의 길이는 그 위키가 겪은 실패의 수를 나타낸다. 자기 위키의 규칙이 넷에서 늘지 않았다면 아직 실패를 만나지 않았거나 만난 실패를 규칙으로 옮기지 않은 것이다.

규칙 목록이 길어지는 것에는 대가가 따른다. 겪은 일을 전부 문장으로 올리면 규칙 문서는 몇 달 만에 아무도 끝까지 읽지 않는 길이가 되고 읽히지 않는 규칙은 없는 규칙과 같다. 그래서 같은 방식으로 운영하는 작업 공간들에는 무엇을 규칙으로 올릴지 정하는 조건이 따로 적혀 있다. 네 가지 가운데 하나에 걸리면 올린다. 같은 혼선이 두 번 이상 반복되었을 때, 한 번뿐이어도 비용이 크거나 되돌리기 어려웠을 때, 사용자가 앞으로는 이렇게 하라고 교정했을 때, 다음 세션의 에이전트가 같은 결정을 그대로 재현해야 할 때다. 올리지 않을 것도 한곳에 적혀 있다. 한 사례에만 해당하는 파일명과 상태, 아직 검증하지 않은 임시 회피책, 특정 도구를 기본값으로 굳히려는 취향, 검증할 수 없는 감상형 문장이 여기 든다. 규칙을 쓰는 형식도 정해 두었다. 어떤 상황에서 무엇을 하고 그것이 되었는지를 무엇으로 확인하는지, 세 요소를 한 문장 안에 담는다. 검증 요소가 빠진 문장은 규칙처럼 생겼어도 지켜졌는지 확인할 방법이 없어서 반년 뒤에 그 문장을 근거로 무엇을 고치자고 말할 수 없다. 앞의 네 규칙에 세 요소가 어떻게 들어 있는지는 마지막 규칙에서 가장 잘 보인다. 상황은 해당 논문이 위키에 없을 때이고 행동은 없다고 말하고 PDF를 요청하는 것이며 검증은 그 답에 위키 밖에서 온 문장이 섞였는지 훑는 것이다.

에이전트가 PDF 4,236건을 검수하면서 후보 서른 건만 눈으로 확인한 뒤 전체를 검수한 것처럼 보고한 적이 있다. 또 다른 작업에서는 11,029건을 확인했다고 했지만 실제 검사는 제외 저널을 가려내는 지문 한 가지에 그쳤다. 두 보고 모두 전수를 마친 것처럼 읽혔고 숫자가 함께 적혀 있어서 더 그렇게 보였다. 이 실패 뒤에 전수 작업은 시작할 때 분모를 정하고 끝날 때 분자와 함께 보고한다는 규칙이 생겼다. 몇 건 중 몇 건을 어떤 방법으로 확인했는지를 같은 문장에 적으면 표본 검사를 전수 검사로 잘못 읽기 어려워진다. 규모가 커더라도 표본만 보고 전체를 확인했다고 보고하지 않는다. Placeholder 금지 규칙도 같은 문제에서 생겼다. 원문이 완전한 정리를 뒷받침하지 못하는데도 metadata와 초록과 파일명만으로 검색에 걸리는 페이지를 만들면, 그 페이지는 나중에 근거로 인용되고 인용하는 쪽은 그것이 비어 있다는 것을 알 수 없다.

설치 전에 정할 질문과 카테고리

폴더를 만들기 전에 답해야 하는 질문이 다섯 개 있다. 머릿속에 있는 상태로는 설치가 진행되지 않으므로 문장으로 적는다. 답을 적은 문서는 설치가 끝난 뒤 지침 파일로 옮겨 간다.

- 이 위키가 앞으로 몇 년 동안 반복해서 답해야 할 질문은 무엇인가.
- 처음 넣을 논문 5-10편은 무엇인가.
- 그 논문들을 나눈다면 어떤 구분이 실제 검색과 읽기에 도움이 되는가.
- 방법 중심, 현상 중심, 대상 중심 가운데 어느 축을 1차 분류로 삼는가.
- 이 위키에 넣지 않을 자료는 무엇인가.

마지막 질문이 가장 덜 묻어지고 가장 쓸모가 많다. 경계가 없는 카테고리는 관련 있어 보이는 것을 전부 받아들이고 그러면 그 이름으로 검색해도 아무것도 좁혀지지 않는다. 그래서 카테고리마다 포함하는 것과 제외하는 것을 한 문장씩 적는다. 제외 문장이 떠오르지 않는다면 그것은 아직 카테고리가 아니고 이름일 뿐이다. 두 문장은 나중에 논문 한 편을 어디에 놓을지 정할 때 인용되는 근거가 되고 판단이 사람마다 세션마다 흔들리는 것을 막는다. 적는 데 걸리는 시간은 카테고리마다 몇 분이고 적지 않으면 그 몇 분이 논문마다 되돌아온다. 분류를 에이전트에게 맡길 수 있는 것도 이 두 문장이 있을 때다. 문장이 없으면 에이전트는 제목의 단어와 폴더 이름의 단어를 맞춰 보는 수밖에 없고 그 방식은 제목이 애매한 논문에서 곧바로 무너진다. 넷째 질문의 축도 같은 목적을 갖는데 방법으로 나눌지 현상으로 나눌지 대상으로 나눌지를 정해 두지 않으면 목록 안에서 축이 섞이는 속도가 빨라진다.

카테고리를 처음부터 잘 나누려고 붙들고 있을 이유는 없다. 시작할 때 정하는 다섯에서 열 개는 임시 골격이고 어차피 자료가 쌓이면서 바뀐다. 무엇을 어디에 놓을지 판단이 서지 않는 논문이 나오면 그때 카테고리를 고치면 되고 그 판단이 흔들린 횟수가 고칠 시점을 알려 준다. 자료가 어느 정도 모인 뒤에는 분류 자체를 에이전트에게 맡길 수도 있다. 지금 들어와 있는 논문 전부를 읽고 어떤 축으로 나누는 것이 좋을지, 지금 카테고리 가운데 합치거나 쪼갤 것이 있는지 물으면 제안이 온다. 이 제안은 실제로 들어온 자료에서 나오므로 처음에 상상으로 만든 목록보다 자기 관심사에 가깝다. 다만 제안을 그대로 받지는 않는다. 카테고리를 바꾸면 그 아래 문서의 경로가 바뀌고 위키링크가 따라 움직여야 하므로 무엇을 어디로 옮길지 목록으로 먼저 받아 확인한 뒤에 실행한다. 처음에 완벽한 목록을 만들려다 시작을 미루는 것보다, 대충 다섯 개로 시작해 논문 신 편을 넣은 뒤 한 번 정리하는 편이 빠르다.

성숙한 목록을 그대로 가져다 쓰면 안 되는 이유는 그것이 남의 질문 이력이기 때문이다. 이 책이 근거로 삼은 위키의 카테고리는 2026년 8월 기준 50개인데, asd-ndd나 gwas처럼 주제로 나뉜 것과 genomic-dl, single-cell-dl, statistics처럼 방법으로 나뉜 것이 한 층에 섞여 있다. 크기도 고르지 않아서 2,236건이 든 카테고리 11건이 든 카테고리가 나란히 있다. 축이 섞이고 크기가 벌어진 것은 몇 년 동안 관심사가 옮겨 다닌 흔적이고 그 흔적을 처음부터 복제하면 자기 자료를 넣을 때마다 판단이 서지 않는다. 축을 고르는 기준은 다시 찾을 때 어떤 질문을 던질 것인가다. 몇 년 뒤에 던질 질문의 모양을 지금 다 알 수는 없지만 지난 2년 동안 반복해서 검색했던 것이 무엇이었는데는 되짚을 수 있다. 세미나 자료를 주차별로 나누면 학기가 끝나는 순간 쓸 수 없게 되지만 학생이 반복해서 막히는 지점으로 나누면 다음 학기에도 한곳에 자료가 쌓인다. 시작 규모는 5개에서 10개가 적당하고 처음부터 완전한 분류를 만들려는 시도는 대개 쓰지 않을 폴더를 늘리는 것으로 끝난다. 쪼갤 신호는 폴더의 크기에서 오지 않고 한 폴더를 서로 다른 질문으로 두 번 열게 되는 순간에 온다.

1. 이 위키가 몇 년 동안 반복해서 답해야 할 질문을 한 문장으로 쓴다. 질문의 형태로 쓴다
2. 처음 넣을 자료 5-10편의 제목을 적는다. 이미 읽은 것으로 고른다
3. 그 자료를 5-10개 카테고리로 나누고 각 카테고리에 포함하는 것과 제외하는 것을 한 문장씩 적는다
4. 1차 분류 축이 방법인지 현상인지 대상인지 문서 맨 위에 적는다

성공 기준은 다음 한 가지로 판단한다. 아직 넣지 않은 자료 세 편의 제목만 새 세션의 에이전트에게 주고 어느 카테고리로 갈지 물었을 때, 셋 중 최소 둘에서 문서 안의 문장을 근거로 답이 나오는지로 판단한다. 나머지 하나에서 경계가 모호하다는 답이 나온다면 그곳이 다음에 고칠 곳이다. 세 편 모두 답이 같았다면 축이 정해지지 않은 것이고 한 카테고리로 몰린다면 카테고리가 너무 넓은 것이다. 적어 둔 목적 한 문장과 카테고리 표는 지침 파일에 옮겨 적는다. 별도 파일로만 두면 에이전트가 매번 읽지 않는다. 카테고리마다 얼마나 정교한 규칙을 붙일지도 설치 시점에 정해지지 않는다. 같은 골격으로 운영하는 작업 공간들을 나란히 놓고 보면 규칙 문서의 두께가 서로 크게 다르다. 판단이 자주 필요한 업무일수록 규칙이 두껍고 들어오는 것의 모양이 정해져 있고 처리 절차가 한 줄로 끝나는 업무는 얇게 끝난다. 두께가 그 업무의 중요도를 나타내지는 않는다. 규칙 문서를 아예 두지 않은 폴더도 있는데 파일을 넣고 꺼내기만 하면 되고 그 사이에 판단이 개입하지 않는 보관소가 그렇다. 모든 폴더에 규칙과 위키를 만들면 유지할 것만 늘고 읽히지 않는 문서가 쌓인다. 카테고리도 같은 방식으로 자란다. 논문이 서너 편뿐인 카테고리에 포함과 제외 문장을 정교하게 다듬을 이유는 없고 매주 논문이 들어오면서 어디에 놓을지가 매번 흔들리는 카테고리는 두 문장으로 부족해 예시까지 붙게 된다. 어느 카테고리에 규칙을 더 쓸지는 그 카테고리에서 판단이 몇 번 흔들렸는지가 정하고 그 횟수는 폴더를 만드는 날에 알 수 없다.

Gist로 폴더를 세우는 과정

폴더를 세우는 일은 설치 문서를 에이전트에게 주고 몇 가지를 답하는 것으로 끝난다. 문서는 아래 주소에 있다.

<https://gist.github.com/joonan30/cbce305684d079dbe9a3fbaefe4e3959>

설치 화면과 메뉴가 달라도 사람이 정하는 항목은 같다.

1. 에이전트 도구를 하나 설치한다.
2. 비어 있는 폴더를 만들고 그 폴더에서 세션을 연다.
3. 설치 Gist 주소를 주고 그 안의 파일을 전부 읽게 한다.
4. 자기 연구 분야와 시작 카테고리 5-10개를 답한다.
5. 폴더와 지침 파일을 만들게 한다.
6. 첫 PDF 5-10편을 넣고 ingest한다.

두 번째 단계에서 폴더가 비어 있어야 하는 이유는 섞임을 막기 위해서다. 논문과 원고와 발표 자료가 이미 들어 있으면 에이전트가 그것부터 새 구조로 옮기기 시작하고 며칠 뒤에는 무엇이 원래 있던 파일이고 무엇이 설치가 만든 파일인지 구분하기 어려워진다. 세 번째 단계에서 읽히는 설치 Gist는 절차와 운영 규칙을 담은 `llm-wiki-gist.md`와 지침 파일의 원본인 `AGENTS.md.template` 두 파일로 되어 있다. Gist 하단에도 같은 위키의 규모가 적혀 있는데 그 문서를 마지막으로 고친 날의 집계이므로 이 책이 직접 센 수치와 카테고리 수와 문서 수가 조금 다르다. 템플릿은 `CLAUDE.md`로도 연결되는데 도구마다 찾아 읽는 지침 파일의 이름이 달라서 같은 내용을 두 이름으로 읽히게 하기 위해서다. 심볼릭 링크로 걸어 두면 한 파일만 고쳐도 두 도구가 같은 규칙을 본다. 카파시의 원형이 파일 하나짜리 발상 문서인 데 비해 이 설치 Gist는 몇 년의 운영에서 얻은 규칙과 폴더 구조를 담은 시작 템플릿이므로 둘을 같은 문서로 보면 안 된다. 앞 절에서 본 잘못된 링크가 바로 이 템플릿 안에 있다는 것도 함께 기억할 만하다. 운영에서 얻은 것을 담은 문서는 운영에서 생긴 오류도 함께 담는다.

따라해보기

아래 블록은 AI 도구의 채팅창에 그대로 붙여 넣는 프롬프트다. 세 번째와 네 번째 단계에서 세션에 주는 요청은 다음 정도면 된다. 꺾쇠로 표시한 세 항목을 자기 것으로 바꿔 그대로 붙여 넣는다. 뒤의 세 줄을 빠뜨리면 설치는 끝난 것처럼 보이는데 여섯 번째 단계에서 막힌다. 지침 파일이 부르는 추출 스크립트가 없기 때문이고 심볼릭 링크가 없으면 도구에 따라 지침 파일 자체가 읽히지 않는다. 템플릿은 분류 축으로 방법을 못 박아 두었으므로 현상이나 대상을 골랐다면 그때 어느 쪽을 따를지 정해야 한다.

```
Gist          LLM-Wiki          .
https://gist.github.com/joonan30/cbce305684d079dbe9a3fbaefe4e3959
```

```

:
<          >

category:
<5-10          >

1      :
< , ,          >

AGENTS.md          CLAUDE.md
. index.md          . Gist
scripts/extract_pdf_text.sh          .
```

위 프롬프트를 입력하면 설치가 진행되는 동안 다음과 같은 질문이 나올 수 있다.

설치 과정에서 연구 분야를 묻는 화면

설치 과정에서 에이전트가 위키의 연구 분야와 카테고리 기준을 묻는 화면이다. 자신의 연구에 가까운 항목을 고르거나 직접 입력하면 된다.

```
llm-wiki/
  AGENTS.md
  index.md
  papers/
  sources/
  wiki/
    {category}/
    concepts/
    overviews/
    questions/
  agenda/
  materials/
  logs/
  scripts/
    , ,
```

만들어지는 폴더는 papers/, sources/, wiki/{categories}/, wiki/concepts/, wiki/overviews/, wiki/questions/, agenda/, materials/, logs/, scripts/다. 열 개가 한 번에 생기지만 대부분은 한동안 비어 있고 설치 직후에 내용이 있는 것은 지침 파일과 index.md다. papers/는 여섯 번째 단계에서 PDF를 넣어야 채워지므로 다섯 번째 단계가 끝난 시점에는 비어 있다. 카테고리별 문서 목록을 두는 indexes/는 이 단계에서 만들지 않는다. 문서가 몇백 건을 넘어 목차만으로 후보를 좁히지 못할 때 검색 색인과 함께 만든다. 같은 위키의 지금 최상위에는 그 폴더들 외에 docs/, indexes/, interactives/, papers-supplementary/가 있고 AGENTS.md, CLAUDE.md, index.md가 함께 놓여 있다. 늘어난 폴더는 계획에서 나오지 않고 필요에서 갈라졌다. 보충 자료 PDF를 본문 PDF와 같은 폴더에 두었더니 논문 수를 셀 수 없어서 papers-supplementary/가 갈라져 나왔고 지침 파일이 길어져 끝까지 읽히지 않게 되자 상세 규칙이 docs/ 아래로 내려갔다. 갈라져 나온 것 가운데 몇은 이미 설치 Gist의 구조 설명으로 되돌아가 있어서 지금 Gist를 읽는 사람은 그 폴더들을 첫날부터 문서로 만난다. 프롬프트가 만들라고 지시하는 목록과 Gist가 그려 놓은 구조가 이 대목에서 어긋나므로 설치 결과가 둘 중 어느 쪽과도 다르면 프롬프트 쪽을 기준으로 본다. 설치할 때 폴더를 많이 만들어 두면 준비된 상태처럼 보이지만 쓰이지 않는 폴더는 검색과 분류에서 매번 후보로 올라와 판단을 흐린다. 검색 환경도 같은 순서로 붙는다. 논문이 수십 편일 때는 목차 문서 하나로 어디에 무엇이 있는지 다 보이므로 색인을 만들 이유가 없고 문서가 몇백 건을 넘어 목차만으로 후보를 좁히지 못하게 될 때 어휘 검색 색인을 세운다. 설치 직후에 값이 있는 산출물은 앞 절에서 적은 목적 한 문장과 카테고리별 두 문장이다. 폴더는 나중에 만들어도 되고 문장은 나중에 만들면 이미 잘못 쌓인 자료를 되돌려야 한다.

설치가 만든 폴더 하나로 모든 일이 처리되지는 않는다. 논문을 다루는 이 위키 옆으로 원고를 쓰는 작업 공간, 과제를 관리하는 작업 공간, 회의와 행사를 기록하는 작업 공간이 하나씩 갈라져 나왔고 각각이 자기 지침 파일과 자기 위키를 갖는다. 갈라지고 나면 새 결정이 생긴다. 지금 이 작업을 어느 공간에서 해야 하는가. 지금 열려 있는 폴더나 요청에 들어 있는 단어 하나로 정하면 같은 종류의 작업이 매번 다른 곳에서 벌어지므로 기준은 그 작업이 만들어 내는 산출물을 누가 소유하는가로 잡았다. 산출물을 소유한 공간이 주가 되어 파일명과 폴더 배치와 검증과 완료 기준을 정하고 나머지는 근거와 원자료와 제약만 공급한다. 논문 근거가 필요한 원고 작업이 그 예다. 문장과 파일은 원고 쪽 규칙을 따르고 인용할 논문과 그 논문에 대한 판단은 이 위키에서 가져온다. 산출물이 여럿이면 각각을 소유한 공간의 규칙을 따로 적용하고 편하다는 이유로 한 폴더에 몰아 두지 않는다. 어느 공간에 들어가든 첫 몇 분에 하는 일은 같다. 지침 파일을 먼저 읽고 그 위키의 지도를 읽고 변경 이력을 읽고 마지막으로 해당 업무의 규칙 문서를 연다.

공간이 갈리면 곧바로 복제의 유혹이 생긴다. 원고를 쓰는 쪽에서 논문 정보가 자꾸 필요하니 그쪽에 논문 목록을 하나 만들어 두고 싶어지고 그렇게 만든 사본은 만든 날에만 정확하다. 그래서 각 작업 공간의 위키는 이 위키의 내용을 복제하지 않고 절대경로로 가리킨다. 각 공간의 라우팅 표에는 여기에 중복해 적지 않는다는 열이 따로 있다. 참조는 양쪽에서 건다. 작업 공간의 규칙이 논문 위키를 가리키고 논문 위키의 진행 문서도 그 작업 폴더의 경로를 가리키며 파일을 옮기는 일은 없다. 소유도 나뉘어 있어서 사람과 연락처, 문헌 근거, 저널 정보, 문체 참고는 논문 위키가 갖고 폴더와 파일명과 버전과 렌더링과 검증 절차는 각 작업 공간이 갖는다. 실물 자산은 규칙 문서와 떼어 별도 폴더에 두고 필요한 곳에서 절대경로로 끌어다 쓴다. 이 층에는 심볼릭 링크를 만들지 않는데 앞에서 지침 파일 두 이름을 잇는 데 쓴 것과 성격이 다르기 때문이다. 같은 폴더 안에서 도구 두 개가 한 파일을 서로 다른 이름으로 찾게 하는 것과, 서로 다른 공간의 문서를 링크로 잇는 것은 몇 달 뒤의 상태가 다르다. 뒤의 것을 하면 어느 것이 실제 파일이고 어느 것이 다른 곳을 가리키는 표지인지 열어 보기 전에는 알 수 없다.

설치 직후에 확인할 것

설치가 끝났다는 것과 위키가 작동한다는 것은 다르다. 확인할 항목은 여섯 개이고 전부 30분 안에 끝난다.

- 지침 파일이 실제 도구에서 읽히는가.
- 원문 PDF와 생성된 Markdown이 서로 다른 폴더에 놓였는가.
- 카테고리마다 포함과 제외 문장이 있는가.
- Source note와 논문 페이지의 항목 틀이 만들어졌는가.
- 빈 placeholder나 웹에서 채운 예제 페이지가 없는가.
- 첫 논문이 종합 문서와 연결되도록 하는 규칙이 있는가.

지침 파일은 새 세션을 열어 실제로 적용되는지 확인한다. 나머지는 폴더와 지침 파일을 열어 위 목록대로 살펴보면 된다. 빈 폴더는 그대로 두고 원문이 없는 예제 페이지로 채우지 않는다. 조건을 충족하지 못한 항목이 있으면 논문을 넣기 전에 먼저 고친다.

4장. 논문 한 편을 LLM-Wiki에 넣는 법

논문을 폴더에 넣고 AI에게 요약을 시키면 파일이 하나 생긴다. 그 파일은 대체로 정확하고 읽기에 무리가 없다. 문제는 석 달 뒤에 생긴다. 비슷한 주제의 논문이 열 편으로 늘었을 때, 이 열 개의 요약 파일은 서로에 대해 아무것도 모른다. 어느 논문이 어느 논문의 방법을 물려받았는지, 두 논문의 결론이 왜 다른지, 새로 들어온 논문이 기존 판단을 바꾸는지가 어디에도 없다. 요약을 만드는 일과 논문을 지식으로 들이는 일은 서로 다른 작업이다. PDF 한 편이 다시 질문할 수 있는 지식으로 변할 때까지, 원문의 정체를 확인하고 독서 기록과 지식 페이지를 구분해 만들고 기존 종합 문서에 연결하는 여섯 단계를 따라간다. 여섯 단계에 붙은 세부 규칙은 설계도에서 나온 것이 드물고 대개 한곳에서 반복해 틀린 뒤에 하나씩 붙었다. 그래서 규칙만 떼어 읽으면 지나치게 깐깐해 보이고 그 규칙이 생기게 만든 사고와 함께 읽으면 최소한으로 보인다.

한 편의 ingest가 만드는 것

논문 한 편을 넣는 작업을 ingest라고 부른다. 한 번의 ingest는 같은 논문을 가리키는 세 파일을 만드는데 셋은 서로 다른 일을 한다. PDF는 근거이고 source note는 그 근거를 읽은 기록이며 wiki page는 그 논문이 다른 논문들과 어떻게 이어지는지를 담는 노드다. 이 위키의 설계 원칙 첫 줄에 세 층이 그 순서로 적혀 있다. 고칠 수 없는 PDF에서 요약 문서로, 요약 문서에서 최종 페이지로 내려가는 한 방향이고 반대로 올라가는 흐름은 없다. 셋을 하나로 합치고 싶은 유혹이 생기지만 합치면 각각이 못 하게 되는 일이 있다. PDF는 검색되지 않고 source note는 너무 길어 다른 논문과 나란히 비교하기 어려우며 wiki page는 원문의 세부를 담기에 좁다. 이 책이 근거로 삼은 연구용 위키에는 2026년 8월 기준으로 원문 PDF 16,127건과 source note 15,995건, wiki page 17,591건이 있고 wiki page는 50개 카테고리 나뉘어 있다. 이 규모에서 세 층이 섞여 있으면 검색이 원문 발췌와 해석을 구분하지 못하고 어떤 문장이 논문의 주장인지 정리자의 판단인지 알 수 없게 된다. 세 파일에 종합 문서 갱신과 변경 기록까지 더하면 한 번의 ingest가 흔적을 남기는 자리는 다섯이다.

paper.pdf

```
→ papers/{stem}.pdf
→ sources/{stem}.md
→ wiki/{category}/{stem}.md
→ overview / concept / question
→ logs/ + index
```

Step 0. PDF의 정체성과 완전성 확인

가장 먼저 하는 일은 이 파일이 무엇인지 확인하는 것이다. 당연해 보이는 단계이고 그래서 가장 자주 건너뛰는 단계이기도 한데, 여기서 틀리면 뒤의 모든 작업이 잘못된 대상 위에 쌓인다. 잘못 들어온 논문은 조용히 자리를 잡는다. 요약이 만들어지고 카테고리가 정해지고 종합 문서에 링크가 걸린 뒤에야 이 파일이 애초에 논문이 아니었다는 것을 알게 되면 되돌릴 곳이 다섯 군데다. 첫 페이지에서 제목, 저자, 연도, DOI, 문서 유형을 확인한다. 여기서 걸려야 하는 것이 정정 공고와 철회 공고다. Correction, erratum, retraction notice는 대체로 한두 쪽이고 서지 정보의 형식이 논문과 비슷해서 자동 처리에서 잘 구분되지 않는다. 정정 공고를 일반 논문으로 넣으면 제목만 그럴듯하고 내용이 없는 페이지가

생기고 이후 검색에서 계속 올라와 읽을 문서를 고르는 시간을 잡아먹는다. 운영 중인 위키에서는 이 세 종류를 원문 폴더에도, 독서 기록 폴더에도, 지식 페이지 폴더에도 넣지 않기로 정해 두었고 판정을 매번 사람의 주의력에 맡기지 않으려고 `is_correction_notice()`라는 함수가 제목을 받아 걸러 낸다. 철회 공고는 걸러 내는 것으로 끝나지 않는다. 철회된 원논문이 이미 위키에 있다면 그 논문 페이지에 철회 사실을 반영해야 하고 그 논문을 근거로 삼은 종합 문서까지 되짚어야 한다. 철회 공고가 들어왔다는 것은 기존 지식을 고칠 일이 생겼다는 뜻이다.

이 걸러 내기에서 판별 조건을 어떻게 잡았는지가 규칙 하나를 만드는 방식을 보여 준다. 제목의 `correction`이라는 단어만으로는 공고를 판별할 수 없다. 유전자 교정을 다루는 논문, 배치 효과 보정을 다루는 논문, 다중 비교 보정을 다루는 논문의 제목에는 그 단어가 정상적으로 들어간다. 단어의 존재만으로 걸러 내면 그 분야의 핵심 논문이 위키에서 통째로 사라지고 사라졌다는 사실은 아무 데도 기록되지 않는다. 그래서 조건을 단어가 놓인 자리로 좁혔다. `Correction` 바로 뒤에 콜론이 오거나 `to`가 오는 경우만 공고로 본다. 정정 공고의 제목은 거의 예외 없이 원논문의 제목을 콜론이나 `to` 뒤에 그대로 달고 나오기 때문이다. 조건을 이렇게 좁히면 걸러지지 않고 통과하는 공고가 생기는데 그 실패는 사람이 첫 페이지를 여는 순간 드러난다. 반대 방향의 실패는 드러나지 않는다. 규칙 하나를 만들 때 두 종류의 오류 가운데 어느 쪽이 눈에 보이는지를 먼저 정하는 편이 낫고 보이지 않는 쪽으로 기울지 않게 조건을 좁힌다.

원문의 완전성도 함께 본다. 브라우저에서 저장한 HTML이나 초록만 담긴 파일을 원문 대신 쓰지 않는다. HTML 캡처는 만들지도 보관하지도 않는다는 규칙이 따로 있는데 캡처는 조판과 쪽 번호와 그림 배치를 잃고 접혀 있던 절을 통째로 빠뜨린 채로도 파일 하나로 성립해 버리기 때문이다. 출판사 페이지를 그대로 인쇄한 파일에는 본문 대신 구독 안내와 로그인 화면만 들어 있는 경우가 있는데 파일 크기와 쪽수만 보면 정상으로 보인다. PDF의 쪽수를 보고, 본문이 실제로 있는지 보고, 다른 논문이 뒤에 이어 붙은 합본은 아닌지 보고, 글자가 깨져 있지 않은지 본다. 정확한 PDF를 끝내 구하지 못하면 그 논문은 `ingest`하지 않고 미보유 목록에 남긴다. 절반짜리 자료로 만든 페이지는 위키 안에서 온전한 페이지와 겉모습이 같고 구분되지 않는 순간 위키 전체의 신뢰도가 그 페이지 수준으로 내려간다. 이 확인은 30초면 끝난다. 건너뛰면 몇 달 뒤에 이 논문의 정리 내용이 왜 이상한지 추적하는 데 30분이 들고 그때는 이미 그 내용이 종합 문서에 들어가 있다.

마지막으로 같은 논문이 이미 있는지 찾는다. 같은 DOI가 이미 있는데 파일명이 달라 중복으로 들어오는 일이 생각보다 흔하다. 저자 이름의 표기가 다르거나 프리프린트와 출판본이 각각 들어오거나 제목의 부제가 잘려 있는 경우가 대부분이다. DOI로 먼저 찾고 없으면 제목으로 찾는다. 제목으로 찾을 때는 부제를 떼고 앞부분만으로도 한 번 찾아야 하는데 잘린 제목은 전체 제목으로 검색해도 걸리지 않기 때문이다. 중복이 발견되면 새 파일을 버릴지 기존 것을 대체할지 정해야 하는데 프리프린트와 출판본이라면 출판본을 정본으로 삼되 프리프린트에만 있던 내용이 있는지 확인한다. 보충 자료가 프리프린트 쪽에 더 자세한 경우가 있고 심사 과정에서 통째로 빠진 분석이 프리프린트에만 남아 있는 경우도 있다. 정본을 바꾸기로 했다면 기존 `stem`을 그대로 쓰고 파일만 교체한다.

Step 1. 원문 보존과 텍스트 추출

확인이 끝나면 정본 PDF를 원자료 폴더에 복사하고 그 뒤로는 고치지 않는다. 주석을 달거나 쪽을 자르거나 병합하지 않는다. 원자료를 고치기 시작하면 그 논문에 대한 어떤 주장도 검증할 수 없게 된다. 파일명은 저자, 연도, 제목 토큰으로 만든 공통 `stem`을 쓴다. 실제 위키에서는 `theodoris-2023-transfer-learning-network-biology`처럼 붙인다. 이 `stem`을 PDF와 `source note`와 `wiki page`가 함께 쓰기 때문에, 세 파일이 같은 논문을 가리킨다는 것을 파일명만 보고 알 수 있다. 규칙 없이 이름을 붙이면 나중에 세 파일을 맞추는 작업이 사람의 기억에 의존하게 된다. `Stem`의 첫 토큰인 저자 성을 뽑는 일은 생각보다 자주 틀렸다. 예전에는 그 로직이 여러 스크립트 안에 인라인으로 복사되어 있었고 그 판본은 쉽표로 구분된 저자 목록을 한 사람의 이름으로 읽어 마지막 저자의 성을 돌려주었다. 제1저자의 이름으로 찾을 논문이 교신저자의 이름으로 저장된 파일이 조용히 쌓였다는 뜻이다. 지금은 저자 이름을 뽑는 함수가 `scripts/author_parse.py` 한 곳에만 있고 회귀 검사 41건이 붙어 있다. 같은 판단을 하는 코드가 두 곳에 있으면 한 곳만 고쳐지는 날이 반드시 온다.

PDF와 함께 딸려 오는 보충 자료는 종류를 나눠 다룬다. 심사 파일, 결정 서한, 아무것도 적히지 않은 빈 점검표는 보관하지 않는다. 논문의 내용에 대해 알려 주는 것이 없고 쌓이면 보충 자료 폴더에서 실제로 쓸 파일을 찾기 어려워진다. `Nature Reporting Summary`는 예외로 두고 보존한다. 앞부분은 체크박스라서 넘겨도 되지만 뒷부분은 저자가 직접 쓴 서술이고 검정력 분석, 표본 제외 사유, 눈가림 여부, 소프트웨어 버전, 코호트의 인구 구성이 본문 어디에도 없이 거기에만 적혀 있는 경우가 많다. 방법을 재현하려 할 때 막히는 지점이 대개 그 다섯 가지다. 어느 파일을 버리고 어느 파일을 남길지는 그 안에 저자가 직접 쓴 문장이 있는지로 정한다. 형식만 보고 정하면 `Reporting Summary`처럼 겉모습이 점검표인 문서가 함께 버려지고 버렸다는 기록도 남지 않는다.

텍스트 추출은 실패할 수 있고 더 나쁘게는 조용히 틀릴 수 있다. PDF 추출기는 두 단으로 조판된 본문의 줄 순서를 섞고 표의 셀 경계를 잃고 그림 설명을 본문과 붙이고 수식을 지우고 위첨자를 일반 숫자로 만든다. 위첨자가 무너지면 인용

번호가 본문 숫자와 섞여 이후의 인용 분석이 전부 어긋난다. 추출이 성공했다는 메시지는 파일이 만들어졌다는 뜻일 뿐이고 내용이 맞다는 보장은 어디에도 없다. 이 위키에서 실제로 벌어진 일이 그 차이를 보여 준다. 추출 스크립트는 인자를 세 개 받는데 세 번째 인자가 쪽수처럼 보이지만 글자 수 상한이다. 그것을 쪽수로 읽고 작은 값을 넘긴 적이 있고 스크립트는 시킨 대로 그 길이에서 문서를 자른 뒤 성공했다고 보고했다. 잘린 자리가 Methods 중간이었기 때문에 초록과 서론과 결과는 멀쩡히 들어왔고 빠진 것은 표본 수와 필터링 조건과 통계 모형이었다. 읽는 사람 눈에 빈 곳이 보이지 않는 종류의 손실이다. 그래서 중요한 값은 화면과 대조한다. 결론을 좌우하는 수치, 부호의 방향, 표의 비교군, 그림이 말하는 바는 PDF를 직접 열어 확인한다. 어떤 도구로 언제 어떤 인자로 추출했는지도 기록한다. 나중에 이상한 값이 발견되었을 때 추출 단계의 문제인지 정리 단계의 문제인지 구분하려면 이 기록이 필요하다.

Step 2. Source note 작성

Source note는 원문을 읽은 기록이다. 원문 전체를 읽고 고정된 항목에 맞춰 압축한 문서다.

Source note의 구조. 머리말, 한 문장 요약, 구조화된 근거로 나뉜다

Source note는 논문의 신원과 출처, 한 문장 요약, 방법·결과·한계를 위에서 아래로 배치한다. 같은 항목을 유지해야 여러 논문을 나란히 비교할 수 있다.

머리말에는 제목과 저자와 연도와 식별자, 그리고 어느 원문에서 어떤 도구로 뽑았는지가 들어간다. 그 아래 한 문장 요약이 이 논문 전체를 한 문장으로 압축하고 다시 그 아래로 방법과 핵심 결과가 항목별로 나뉜다. 나중에 다시 열었을 때 위에서 아래로 읽어 내려가면 필요한 깊이에서 멈출 수 있는 구조다. 항목을 고정하는 이유는 논문끼리 비교하기 위해서다. 논문마다 다른 형식으로 정리하면 열 편이 쌓였을 때 나란히 놓을 수 없다. 추출한 원문 전체를 여기에 붙이지 않는다. 원문 정본은 PDF이고 추출된 원시 텍스트를 그대로 넣으면 세 가지 문제가 생긴다. 출판사의 저작권 안내와 투고 이력 같은 상용구가 지식으로 검색되고 잘못 읽힌 표가 사실처럼 노출되며 문서가 길어져 검색 결과에서 어느 부분이 관련되는지 알 수 없게 된다. 이 문서를 쓰는 일 자체를 다른 프로그램에 넘기지 않는다는 규칙도 붙어 있다. Ingest 규칙과 스크립트는 외부 모델의 CLI나 API를 호출하지 않고 논문을 해석하고 문장을 쓰는 일은 지금 이 작업을 하고 있는 에이전트 세션 안에 둔다. 로컬 스크립트가 맡는 범위는 추출, 검증, 복구, 로그, 색인 유지로 한정된다. 해석을 다른 프로세스로 넘기면 그 프로세스가 무엇을 보고 그렇게 썼는지가 세션 밖으로 나가고 위키에는 근거를 되짚을 수 없는 문장만 남는다. 항목은 여덟 개이고 순서는 다음과 같다.

One-line Summary
Document Information
Key Contributions
Methodology and Architecture
Key Results and Benchmarks
Limitations and Future Work
Related Work
Glossary

논문을 넣은 뒤 확인하는 항목

Ingest 뒤에는 요약만 읽지 않고 연구 대상, 방법, 수치, 한계, 관련 연구가 실제 원문에 근거했는지 항목별로 확인한다.

Document Information은 연구 대상, 데이터, 설계, 비교 조건처럼 이 논문이 무엇에 관한 것인지를 정하는 정보다. Key Contributions는 기존 지식과 비교해 새로 더해진 주장이다. 저자가 새롭다고 말한 것 가운데 실제로 새로운 것만 적는다. Methodology and Architecture는 표본, 실험, 모델, 분석 단계, 평가 방식이다. Key Results and Benchmarks는 결과의 방향과 크기, 수치, 비교군, 불확실성이다. 방향만 적고 크기를 빼면 나중에 두 논문을 비교할 수 없다. Limitations and Future Work에는 저자가 인정한 한계와 함께 읽는 사람이 추가로 본 경계를 적는다. 이 둘은 구분해서 적는다. 구분하지 않으면 몇 달 뒤에 그 한계가 저자의 인정인지 읽은 사람의 판단인지 알 수 없고 논문을 인용할 때 저자가 하지 않은 말을 저자의 말로 옮기게 된다. Related Work는 선행 연구와의 관계인데, 여기서 가장 자주 발생하는 오류가 확인하지 않은 관계를 사실처럼 적는 것이다. 이 논문이 저 논문의 방법을 개선했다고 쓰려면 두 논문을 다 읽어야 하고 읽지 않았다면 그렇게 쓰지 않는다. Glossary는 이 논문을 이해하는 데 실제로 필요한 용어만 넣는다. 이 문서가 있어도 수치나 조건이 판단을 좌우하는 대목과 그림이 결론을 만드는 대목에서는 PDF로 돌아가는데 source note는 원문으로 가는 경로이기 때문이다.

Step 3. 카테고리 분류

카테고리는 이 논문을 나중에 어떤 질문으로 다시 찾을지에 대한 결정이다. 같은 논문이라도 위키의 목적에 따라 다른 곳에 놓이는 것이 자연스럽다. 단일세포 데이터로 유전자 조절을 다룬 논문은 방법 중심 위키에서는 답러닝 카테고리에, 질환 중심 위키에서는 해당 질환 카테고리에 놓인다. 둘 다 맞다. 1차 분류 측은 위키 전체에서 일관되게 정한다. 방법으로 나눌지, 현상으로 나눌지, 질환이나 대상 생물로 나눌지를 정하고 지킨다. 축이 섞이면 같은 논문이 두 축 모두에 해당해 어디에 둘지 매번 고민하게 된다. 실제 운영에서는 완전히 한 축으로만 유지하기 어렵다. 앞서 언급한 위키의 50개 카테고리에도 질환과 대상을 나누는 주제축과 방법을 나누는 축이 함께 있다. 그렇더라도 새 카테고리를 만들 때마다 어느 축인지 밝혀 두면 나중에 정리할 수 있다.

실제 운영에서 축이 완전히 하나로 유지되는 일은 드물다. 이 책이 근거로 삼은 위키의 50개 카테고리에는 asd-ndd와 gwas처럼 주제로 나뉜 것과 genomic-dl, single-cell-dl, statistics처럼 방법으로 나뉜 것이 한 층에 섞여 있다. 크기도 고르지 않아서 2,236건이 든 카테고리 11건이 든 카테고리가 나란히 있다. 흔들림이 생기는 자리는 카테고리 목록 자체가 예고한다. 방법축 카테고리 11개와 주제축 카테고리 11개 중 1개는 방법 논문이 특정 질환 자료로 검증을 붙인 순간 두 카테고리에 동시에 해당한다. 그때 어느 쪽으로 보내는지가 세션마다 달라지면 같은 종류의 논문이 두 곳에 흩어지고 나중에 어느 쪽을 찾아야 하는지도 흔들린다. 판단을 고정하는 것은 카테고리마다 적어 둔 포함과 제외 문장이고 그 문장이 없는 카테고리에서 흔들림이 반복된다. 같은 카테고리에서 판단이 세 번 넘게 흔들렸다면 이름을 바꾸거나 쪼갤 후보이며 그 횟수는 폴더를 만드는 날에 알 수 없으므로 로그에 남겨 두어야 보인다.

분류가 애매할 때 던지는 질문은 하나다. 이 논문을 다시 찾을 때 어떤 질문으로 찾을 것인가. 답이 “이 방법을 쓴 논문이 뭐가 있었지”라면 방법 카테고리로 가고 “이 질환에서 뭐가 밝혀졌지”라면 질환 카테고리로 간다. 두 질문이 다 그럴듯하면 더 자주 던질 질문을 고른다. 여러 카테고리에 파일을 복제하지 않는다. 정본 위치는 하나로 하고 나머지는 링크로 연결한다. 복제하면 한쪽만 고쳐지는 순간 어느 것이 최신인지 알 수 없게 되고 이 문제는 시간이 지날수록 커진다. 카테고리의 크기가 균등할 필요는 없다. 실제 위키에서도 2,236건이 든 카테고리 11건이 든 카테고리가 같은 층에 있다. 카테고리를 나누는 기준은 그 안의 질문이 같은지 여부다. 한 카테고리 안에서 서로 다른 질문을 하게 되면 그때 나눈다. 분류는 나중에 고칠 수 있지만 미루면 그동안 이 논문은 검색에서 사실상 보이지 않으므로 완벽한 분류를 미리 만들 필요가 없어도 어딘가에는 놓는다.

카테고리를 정하고 나면 그 논문의 wiki page 이름이 결정된다. 이름 규칙은 저자와 연도 뒤에 제목에서 뽑은 5토콘을 붙이는 것이다. 단위는 5토콘이며 영숫자가 아닌 문자를 지우지 않고 구분자로 세기 때문이다. self-report는 한 단어처럼 보이지만 self와 report 두 토콘으로 쉰다. 규칙을 이렇게 적어 두어도 실제 파일들은 한 형태로 모이지 않았다. 2026년 7월 17일에 wiki 문서 10,340건의 이름을 규칙이 만들어 낼 형태와 대조해 보니, 불용어를 유지한 형태와 일치한 것이 61.3%, 제거한 형태와 일치한 것이 40.1%였고 둘 다에 해당하는 것이 21.6%, 어느 쪽에도 해당하지 않는 것이 20.2%였다. 정본이라고 부를 형태가 존재하지 않는다는 뜻이다. 여기서 손이 근질거리는 선택지가 하나 있다. 스크립트를 짜서 전부 한 형태로 맞추는 것이다. 그 작업은 하지 않기로 했다. Wiki page의 stem을 바꾸면 그 페이지를 가리키는 모든 위키링크와 색인 항목이 함께 바뀌어야 하고 그 전에 어느 형태로 통일할지부터가 결정 사항인데 61.3% 대 40.1%는 어느 쪽도 다수라고 부르기 어려운 숫자다. 이름은 사람이 파일을 알아보게 하려고 있는 것이고 그 목적은 지금의 두 형태로도 달성되고 있다. 새로 만드는 파일에만 규칙을 적용하고 옛 이름은 그대로 둔다.

Step 4. Paper wiki page 작성

Paper wiki page가 맡는 일은 연결이다. 이 페이지에는 다섯 가지가 들어간다. 첫째, source note와 PDF로 되돌아갈 수 있는 정보다. 이 페이지에서 출발해 언제든 근거까지 내려갈 수 있어야 한다. 둘째, 주요 주장과 방법을 검색 가능한 언어로 정리한 내용이다. 나중에 이 논문을 찾을 때 쓸 법한 단어가 실제로 페이지 안에 있어야 검색에 걸린다. 셋째, 관련 종합 문서로 가는 링크다. 넷째, 이 논문이 기존 지식을 어떻게 바꾸는지에 대한 판정이다. 다섯째, 이 논문과 직접 관계된 다른 논문들이다. 넷째가 이 페이지의 핵심이고 가장 자주 빠진다. 새 논문이 기존 주장을 강화하는지, 적용 범위를 좁히는지, 반박하는지, 대체하는지, 아니면 아무것도 바꾸지 않는지를 한 문장으로 적는다. 아무것도 바꾸지 않는다는 판정도 정보다. 같은 결과가 다른 조건에서 재현되었다는 뜻일 수 있기 때문이다. 작업 과정에서 생긴 정보는 이 페이지에 넣지 않는다. 어느 배치에서 처리했는지, 어떤 에이전트가 만들었는지, 추출에 몇 초가 걸렸는지는 운영 기록에 속한다. 이런 정보가 페이지에 섞이면 읽는 사람이 매번 걸러 내야 하고 검색에도 걸린다. 항목 골격은 source note의 여덟 항목보다 짧고 다섯 가지는 그 안에 나뉘어 들어간다.

Summary

Key Contributions

Methodology and Architecture

Wiki page 사이의 링크는 Obsidian이 읽는 `[[wikilink]]` 형식으로 쓴다. 이 저장소는 Obsidian 호환을 유지하기로 했고 그래서 표준 Markdown과 위키링크만 쓰고 특정 도구에만 있는 문법은 쓰지 않는다. 링크 규약에는 한 줄짜리 조건이 붙어 있다. Vault의 루트는 저장소 루트가 아니고 `wiki/` 폴더다. 그러므로 모든 `[[...]]` 대상은 `wiki/` 안에서의 경로로 쓰고 `wiki/` 접두사를 붙이지 않는다. 이 한 줄이 굳이 규칙 문서에 명문화된 이유는 실제 사고 때문이다. 접두사를 붙이는 편이 저장소 루트에서 볼 때 더 정확해 보인다는 판단으로, 약 5만 개의 위키링크에 `wiki/`를 붙이는 일괄 작업을 돌린 적이 있다. Obsidian은 그 링크들을 `wiki/wiki/...`로 해석했고 존재하지 않는 경로를 열 때마다 빈 유명 페이지를 새로 만들었다. 살아 있는 vault가 망가졌고 작업을 통째로 되돌려야 했다. 여기서 정말 문제가 되는 쪽은 검사다. 링크 검사 스크립트는 두 형태를 모두 유효한 링크로 받아들이므로 접두사가 다시 들어와도 아무것도 지적하지 않는다. 기계가 잡아 주지 않는 규칙이 위키에 몇 개 있고 그런 규칙은 잡히지 않는다는 사실까지 규칙 문서에 적어 두어야 지켜진다.

Step 5. Synthesis layer에 연결

논문을 넣었는데 기존 종합 문서와 이어지지 않았다면 ingest는 아직 끝나지 않았다. 이 상태에는 이름이 붙어 있다. 종합 층과 연결되지 않은 채 남은 source note와 wiki page 한 쌍을 synthesis-orphan이라고 부른다. 이름을 붙여 둔 이유는 그 상태가 겉으로는 정상이기 때문이다. 파일 두 개가 제자리에 있고 내용도 충실하고 검색에도 걸린다. 무너진 것은 위키가 하려던 일이다. Synthesis-orphan만 쌓인 위키는 질문이 들어올 때마다 관련 있어 보이는 조각을 꺼내와 그때 답을 조립한다. 조각을 꺼내 답을 만드는 검색 방식과 동작이 같아지고 조각들 사이에서 내린 판단은 세션이 끝나면 사라진다. 논문이 백 편으로 늘어도 답할 수 있는 질문이 늘지 않는 상태가 여기서 나온다. 최소 완료 조건 네 가지는 그래서 문서에 명문화되어 있다. 가장 관련된 overview나 concept 페이지와 양방향 링크를 만든다. 한쪽에서만 가리키면 반대 방향에서 찾을 수 없다. 그 종합 페이지의 본문에 이 논문이 기존 지식에서 차지하는 위치를 한 문장 이상 반영한다. 링크만 추가하고 본문을 고치지 않으면 종합 문서는 목록이 되어 버린다. 앞 절에서 말한 판정 가운데 하나를 적어 기존 주장이 강화되는지, 좁아지는지, 반박되는지, 대체되는지를 점검한다. 적절한 종합 페이지가 없으면 그 공백을 작업 로그에 기록하고 같은 주제가 반복될 것 같으면 새 종합 페이지를 만든다. 네 번째 조건이 가장 자주 생략되는데 공백을 적어 두지 않으면 그 주제에 종합 문서가 없다는 사실을 아무도 다시 발견하지 못한다.

Ingest에 등급을 두지 않는다. Tier 1, lightweight, routine, provisional 같은 이름으로 논문을 나누고 어떤 등급에는 느슨한 기준을 적용하는 방식을 쓰지 않는다는 뜻이다. 등급을 두면 처음에는 시간을 아끼는 것처럼 보인다. 문제는 등급이 붙은 페이지가 위키 안에서 다른 페이지와 똑같이 생겼다는 데 있다. 6개월 뒤에 그 페이지를 읽는 사람은 이 페이지가 대중 만들어졌다는 표시를 어디에서도 볼 수 없고 위키의 다른 페이지와 같은 무게로 읽는다. 위험은 품질이 낮은 페이지에서 오지 않는다. 품질이 낮다는 사실이 페이지 겉면에 남지 않는 데서 온다. 같은 이유로 placeholder 페이지를 만들지 않는다. Metadata와 초록, 다른 논문의 참고문헌에 적힌 서지 정보, 파일명, 추출된 앞부분 몇 쪽만으로도 검색에 걸리는 페이지를 만들 수 있고 그렇게 만든 페이지는 검색 결과에서 제대로 만든 페이지와 구별되지 않는다. 원문이 완전한 source note와 wiki page 한 쌍을 뒷받침하지 못하면 그 논문은 지식 층에서 빠고 공백으로 기록한다. 없는 것으로 기록된 논문은 나중에 채울 수 있고 있는 것처럼 기록된 논문은 다시 열리지 않는다. 배치 처리에도 같은 원칙이 적용된다. 논문 서른 편을 한꺼번에 넣는 것은 실행 전략이고 그 전략은 작업 로그 안에서만 존재해야 한다. 여러 편을 한 번에 넣을 때 추가로 필요한 절차는 6장에서 다룬다. 독자가 읽는 페이지에 batch 이름이나 날짜별 식별자가 남으면, 그 페이지를 읽는 사람은 논문에 대한 정보와 작업에 대한 정보를 매년 스스로 갈라내야 한다.

모든 논문에 같은 기준을 적용한다는 것과 모든 논문에 같은 시간을 쓴다는 것은 다르다. 기준은 하나이고 그 기준을 넘어 얼마나 더 하는지는 논문에 따라 달라진다. 영향이 큰 논문에는 high-impact ingest mode라고 이름 붙인 처리를 쓴다. 관련 overview의 본문을 새로 쓰고 그 논문이 반복해서 쓰는 개념에 concept 페이지를 만들고 관련된 페이지들 사이에 양방향 링크를 걸고 이 논문 때문에 기존 주장이 어떻게 달라졌는지를 문장으로 남긴다. 분야의 논지 자체를 바꾸는 논문에는 더 느린 방식을 쓰고 그 방식에는 Karpathy mode라는 이름이 붙어 있다. 한 편씩만 처리하고 새 논문을 읽기 전에 그 주제의 기존 종합 문서를 먼저 읽고 읽고 나서 주장이 어디에서 어디로 옮겨 갔는지를 명시한다. 배치로 서른 편을 훑는 속도와 정반대인데, 논지를 바꾸는 논문은 애초에 자주 오지 않는다. 종합 층 자체는 세 종류로 나뉘어 있다. 주제를 통째로 서술하는 overview가 590건, 반복해서 등장하는 개념을 정의하는 concept이 408건, 특정 질문에 답하는 question이 550건이다. 새 논문이 어느 종류에 붙을지는 논문이 정하고 세 종류 모두에 붙는 논문도 있다. 이 조건들이 실제로 지켜지는지는 사람의 성실함에 맡기지 않는다. 카테고리별로 종합 층과 연결된 페이지의 비율과 연결되지 않은 orphan의 수를 세는 감사 스크립트가 있고 수치는 카테고리마다 따로 나온다. 어느 카테고리의 orphan이 유독 많다면 그 주제에 종합 문서가 부족하다는 신호이고 그 신호는 논문을 더 넣어서 해결되지 않는다.

한 번의 ingest가 작업 로그에 남기는 기록

작업 로그에는 만든 파일, 분류 근거, 추출 도구, 연결한 개념, 적지 않기로 한 내용을 남긴다. 실행한 일과 제외한 판단이 함께 있어야 작업을 재현할 수 있다.

로그에는 무엇을 어디에 만들었고 왜 그 카테고리로 분류했는지가 문장으로 남는다. 위 화면에서는 폴더 구조를 세운 기록과 첫 논문을 넣은 기록이 이어져 있고 두 번째 항목에는 원본 파일을 어떤 이름으로 바꿨는지, 어떤 도구로 추출했는지, 어느 개념 페이지와 method card를 함께 만들었는지가 적혀 있다. 마지막 줄이 특히 중요한데, 논문에서 확인되지 않아 적지 않기로 한 것을 밝혀 두었다. 무엇을 적지 않았는지가 남아 있어야 나중에 비어 있는 이유를 알 수 있다.

Step 6. 검증, 로그, 색인

먼저 세 파일이 같은 논문을 가리키는지 확인한다. PDF와 source note와 wiki page의 stem이 같은지, 제목과 저자와 연도가 서로 어긋나지 않는지 본다. 다음으로 필수 항목이 비어 있는지 확인한다. 항목 제목만 있고 내용이 없는 부분, 추출 오류가 그대로 들어온 부분, 판정이 빠진 부분을 찾는다. 링크와 카테고리가 실제로 존재하는지도 확인한다. 없는 문서를 가리키는 링크는 나중에 찾기 어렵다. 위키링크의 형태도 이때 눈으로 본다. 검사 스크립트가 접두사 붙은 형태를 통과시키므로 이 확인만은 사람이 대상 경로를 직접 읽는 수밖에 없다. 여기까지가 파일 자체에 대한 검증이고 다음은 이 작업을 기록으로 남기는 일이다.

작업 로그에는 무엇을 왜 추가했는지 남긴다. 파일 복구에는 동기화 서비스의 버전 기록을 쓰지만, 그 기록은 파일이 어떻게 바뀌었는지만 보여 주고 변경 이유는 알려 주지 않는다. 로그 파일은 하루에 하나가 아니고 하루와 에이전트와 기계의 조합마다 하나다. 이름은 날짜, 에이전트 이름, 기계 이름을 이어 붙여 만든다. 에이전트 이름은 claudex 또는 codex로 적는다. 기계 이름이 파일명에 들어가는 이유는 두 대의 Mac이 Dropbox로 같은 폴더를 공유하기 때문이다. 기계 이름을 빼면 같은 날 두 대에서 돌린 작업이 한 파일에 동시에 쓰이고 동기화는 그 충돌을 충돌 사본으로 처리한다. 기계 이름은 기억에 의존하지 않고 `scutil --get ComputerName`을 실행해 확인한다. 에이전트는 자신이 어느 기계에서 실행되는지 안다고 착각하기 쉽다.

로그 파일의 안쪽에도 형식이 있다. Frontmatter에 날짜, 에이전트, 기계, 모델을 적고 항목은 오래된 것이 위로 가도록 아래에 덧붙인다. 모델 이름을 적는 칸이 따로 있는 것은, 같은 지침을 주어도 모델에 따라 결과가 달라지는 일이 실제로 있기 때문이다. 이 값은 나중에 복원할 방법이 없어서 2026년 7월 25일 이전 파일에는 비어 있다. 항목 제목은 날짜, 작업 종류, topic slug, 제목을 순서대로 이어 붙인 한 줄이다. 작업 종류는 ingest, maintenance, query-to-wiki, semantic-lint, supersede 다섯 가지다. Topic slug가 제목 안에 들어가는 이유는 하나의 작업이 하루에 끝나지 않기 때문이다. 어떤 주제의 논문 스무 편을 넣는 일은 사흘에 걸쳐 두 에이전트가 두 기계에서 진행하게 되고 그렇게 흩어진 로그 파일들을 하나의 작업으로 잇는 것이 slug 하나다. 그래서 새 slug를 만들기 전에 기존 topic 표를 먼저 읽는다. 이미 있는 것과 비슷하지만 미묘하게 다른 slug를 새로 만들면, 하나였던 작업이 조용히 둘로 쪼개지고 어느 쪽도 완결되지 않은 것처럼 보인다. 그 topic 표는 손으로 쓰지 않고 항목 제목들에서 생성한다. 생성이 강제하는 것은 정직함이다. 손으로 쓰는 표는 하지 않은 작업을 적을 수 있고 한 작업을 빠뜨릴 수 있지만 생성된 표는 로그에 없는 작업을 주장할 방법이 없다. 로그 폴더의 최상위에는 일자별 파일과 그 생성된 표만 두고 감사 보고서와 기계별 로그와 동결된 과거 기록은 각각 하위 폴더로 내린다.

1. 자신의 분야에서 이미 읽은 논문 한 편을 골라 여섯 단계를 끝까지 수행한다
2. 끝난 뒤 다섯 가지를 나란히 놓고 본다
 - 원문 PDF
 - Source note
 - Paper wiki page
 - 연결된 종합 문서에서 이번에 바뀐 문장
 - 작업 로그와 색인 항목

처음 읽는 논문으로 하면 절차를 익히는 대신 내용을 이해하는 데 시간이 다 간다. 성공 기준은 새 세션을 시작한 에이전트에게 이 논문에 대해 물었을 때, 주장의 근거가 어디에 있고 이 논문이 기존 지식에서 어디에 놓이는지를 설명할 수 있는가다. 파일이 다섯 개 생겼더라도 종합 문서에서 바뀐 문장이 없으면 이 기준을 통과하지 못한다. 연구실을 운영하는 사람에게는 검증 항목이 하나 더 있다. 지식 페이지 층에는 출판된 지식만 담고 준비 중이거나 심사 중이거나 중단된 연구실 원고는 제목조차 넣지 않는다. DOI가 붙은 출판 논문과 공개된 프리프린트는 남고 경계는 공개 여부 하나로 정한다. 문제는 미공개 원고가 자기 페이지의 형태로만 들어오지 않는다는 데 있다. 실제로 확인된 경로가 여섯이다. Frontmatter의 status 값과 비어 있는 DOI 칸, 이미 출판된 논문의 페이지에 덧붙인 내부 코드명 관련 절, tags 목록에 석여 들어간 코드명, 코드명을 제목으로 단 overview 페이지, 위키링크 없이 문장 안에서만 언급된 원고, 그리고 그런 페이지들을 가리키는 인바운드 링크다. 여섯 가지 모두 금지되어 있고 여섯을 따로 적어 둔 이유는 하나만 막으면 나머지는 다섯으로 새기 때문이다. 진행 중인 원고에 대한 메모는 지식 층 밖의 agenda/로 보내고 중단된 원고는 보관하지 않고

지운다. 배치 하나를 닫기 전에 살아 있는 코드명과 manuscript, our paper 같은 표현을 위키 전체에서 검색해 결과가 0건인지 확인한다. 이 검사가 필요한 이유는 위키가 어느 날 공유되거나 논문 초안 작성에 쓰이기 때문이고 그때 새어나가는 것은 아직 심사 중인 결과다. 배치를 닫는 작업이 하나 더 남는다. 색인은 정책에 맞춰 갱신하는데 카테고리 목록과 검색 색인은 문서에서 다시 만들 수 있는 파생 상태이므로 매번 갱신할 필요는 없다.

5장. 논문 PDF를 ingest 하는 방법

4장의 절차에서 두 번째 단계가 텍스트 추출이었고 그곳을 한 문단으로 넘겼다. 실제로는 이 단계가 위키 전체의 품질을 정한다. 추출이 놓친 문장은 그 뒤의 모든 작업에서 존재하지 않는 문장이 되기 때문이다. 원문을 읽었다고 적힌 정리 문서가 실제로는 앞의 열 쪽만 읽은 것일 수 있고 그 사실은 정리 문서 어디에도 나타나지 않는다. 추출기는 하나가 아니고 서로 잘하는 일이 다르다. 어느 것을 쓰느냐에 따라 같은 논문에서 나오는 텍스트의 모양이 달라지고 그 차이가 요약의 정확도와 유전자 이름의 보존율과 통계치의 부호까지 바꾼다.

원본 PDF 선택, 추출기 비교, 추출 결과 검증과 정리 범위를 한눈에 정리한 흐름이다.

원본 PDF를 고르는 기준

추출을 이야기하기 전에 정할 것이 무엇을 추출할 것인가다. 같은 논문의 PDF가 여러 벌 존재하고 그 품질이 서로 다르다. 출판사 페이지에서 받은 최종본, 프리프린트 서버의 원고, 저자 홈페이지에 올라온 사본, 검색 결과에서 받은 출처가 불분명한 파일이 모두 같은 논문을 담고 있지만 쪽 구성과 그림 해상도와 보충 자료의 유무가 다르다. 확보 순서를 정해 두면 매번 고민하지 않아도 된다. 이미 위키에 정확한 원본이 있으면 그것을 쓰고 없으면 출판사나 정식 저장소의 PDF를 구하고 둘 다 구하지 못하면 그 논문은 넣지 않고 미보유 목록에 남긴다. 세 번째 경우에 브라우저에서 페이지를 저장한 파일로 대신하지 않는 것이 중요하다. 그렇게 만든 파일은 조판과 쪽 번호를 잃고 접혀 있던 절을 통째로 빠뜨린 채로도 파일 하나로 성립한다. 겉으로는 논문이 하나 들어온 것처럼 보이고 안에는 절반만 들어 있다.

프리프린트와 출판본 가운데 무엇을 정본으로 삼을지도 정해 둔다. 대체로 출판본이 정본이지만 프리프린트를 함께 확인해야 하는 경우가 있다. 심사 과정에서 분석 하나가 통째로 빠지는 일이 있고 보충 자료가 프리프린트 쪽에 더 자세한 경우도 있다. 두 판본을 모두 가지고 있다면 정본은 출판본으로 두되 프리프린트에만 있던 내용을 확인하고 확인한 결과를 정리 문서에 한 줄로 남긴다. 이 확인을 하지 않으면 나중에 그 논문의 예전 판본을 인용한 다른 논문을 읽을 때 앞뒤가 맞지 않는다. 어느 파일을 실제로 썼는지도 기록에 남긴다. 같은 논문에 대해 서로 다른 사람이 서로 다른 판본을 읽고 다른 결론을 적는 일이 생기고 파일이 기록되어 있으면 그 차이가 판본에서 왔다는 것을 곧바로 알 수 있다.

추출이 어려운 이유

PDF는 인쇄면을 담는 형식이다. 파일 안에는 어느 좌표에 어느 글자를 어느 폰트로 찍으라는 지시가 들어 있을 뿐, 문단과 제목의 구조는 없다. 사람이 읽을 때는 두 단으로 나뉜 지면에서 왼쪽 단을 끝까지 내려간 뒤 오른쪽 단으로 올라가지만 파일에는 그 순서가 적혀 있지 않다. 추출기는 좌표를 보고 읽는 순서를 추측하고 그 추측이 틀리면 왼쪽 단의 한 줄과 오른쪽 단의 한 줄이 번갈아 나온다. 학술 논문은 이 문제가 특히 심한 형식이다. 두 단 조판에 그림과 표가 끼어들고 그림 안에는 축 눈금과 날개 라벨 같은 짧은 글자가 흩어져 있으며 쪽마다 저널 이름과 쪽 번호가 붙는다. 이 조각들이 본문 문장 사이에 섞여 들어오면 문단이 중간에서 끊기고 끊긴 자리에 그림 라벨이 들어가 문장처럼 읽힌다. 사람은 그 결과를 보면 곧바로 이상하다고 느끼지만 요약을 만드는 쪽은 그것도 본문으로 받아들인다.

두 번째 어려움은 글자 자체가 파일 안에서 무엇인지 알 수 없는 경우다. 출판사가 그리스 문자나 부등호 같은 기호를 유니코드 매핑이 없는 서브셋 폰트에 심으면, 파일 안의 그 글자에는 어떤 문자인지에 대한 정보가 없다. 화면에는 제대로 보이는데 추출하면 원래 글자가 나오지 않는다. 실측해 보면 세 가지 형태가 있고 뒤로 갈수록 위험하다. 글자가 그냥 빠지는 경우, 다른 글자로 바뀌는 경우, 그리고 숫자의 부호나 지수가 사라지는 경우다. 빠진 글자는 문장이 어색해져서 눈에 걸린다. 바뀐 글자는 그럴듯한 문장으로 남아서 걸리지 않는다. 세 번째는 최악인데, 음의 상관계수가 양수로 적힌 문장은 문법적으로 완전하고 내용도 그럴듯해서 원문과 대조하기 전에는 발견되지 않는다.

이 문제가 생물학 논문에서 특히 자주 나타나는 이유가 있다. 유전자와 단백질과 세포 유형의 이름에 그리스 문자가 많이 쓰이고 통계 결과에는 부등호와 지수가 붙는다. 그리스 문자가 탈락하면 서로 다른 계열의 이름이 같은 문자열이 되어 버린다. 지수 표기가 깨지면 유의수준의 자릿수가 달라지고 그 값을 근거로 판단한 문장이 전부 흔들린다. 이런 종류의 손상은 논문마다 다르게 나타나는데 같은 출판사의 같은 저널이라도 조판 시점에 따라 폰트 처리가 다르기 때문이다. 어떤

5장 논문 PDF를 ingest 하는 방법

PDF 추출이 논문 위키의 품질을 결정해요! 추출이 놓친 문장은 그 뒤의 모든 작업에서 '존재하지 않는 문장'이 되어버리거든요.

1. 원본 PDF를 고르는 기준

- 이미 위키에 정확한 원본이 있으면 → 그것 사용
- 없으면 → 출판사나 정식 저장소 PDF 확보
- 둘 다 없으면 → 넣지 않고 미보유 목록에 남김

⚠ 브라우저로 저장한 파일은 금지!
조판/쪽번호를 읽고 절반만 들어갈 수도 있어요.

출판본이 정본이지만, 프리프린트도 함께 확인할 때가 있어요!

- ✓ 프리프린트에만 있는 내용 확인
- ✓ 확인 결과를 정리 문서에 한 줄로 기록
- ✓ 실제로 사용한 파일(버전) 기록 필수!

기록 예시

출판본: Nature 2024 v1 (2024-04-10)
프리프린트: bioRxiv 2023-11-02
확인일: 2026-05-30

2. 추출이 어려운 이유

1 PDF는 인쇄면을 담은 형식

두 단 조판! 읽는 순서를 추측해야 해요.

→ 그림, 표, 페이지 정보가 본문 사이에 섞이면 문단이 끊기고 그림 라벨이 문장처럼 읽혀요.

2 글자 자체를 모르는 경우

폰트 문제로 글자가 깨질 수 있어요!

빠짐	다른 글자로	부호/지수 사라짐 (최악!)
$\alpha \rightarrow \dots$	$\beta \rightarrow B$	$-0.75 \rightarrow 0.75$ $1e-5 \rightarrow 1e-05$

⚠ 생물학 논문은 그리스 문자, 부동소수점, 지수 표기가 많아 위험해요!

따라서, 결론을 좌우하는 **수치**는 원문 화면과 대조 필수!

그림이 담고 있는 정보는 텍스트 추출로는 안 나와요!
(히트맵, 모델 구조도 등)

→ 필요할 때만 이미지로 따서 모델이 직접 보게 해요.
(논지에 중요한 논문에 한정)

3. 추출기를 고르는 두 갈래 (2026-05-30 실측 비교)

요약과 구조 생성에는?

opendataloader-pdf

- ✓ 읽기 순서 복원 (두 단 섞임 해결)
- ✓ 그림 설명과 본문 분리
- ✓ 절 제목을 제목으로 표시
- ✓ 쪽 장식(저널명, 쪽번호) 제거
- ✓ 참고문헌 정돈

PyMuPDF는 두 단을 흘려 일부 구간이 아예 판독되지 않았어요.

목록에서 이름 끌어내기에는?

PyMuPDF

- ✓ 단 구조를 그대로 유지
- ✓ 유전자 탐지: 226건 vs 221건 (거의 동일)
- ✓ 관계 추출: 15건 vs 15건 (동일)
- ✓ 속도 6~24배 빠름
- ✓ Java 환경 불필요

어느 표의 어느 줄에서 나온 이름인지 되짚기 더 빠듯!

용도	맞는 추출기	이유
요약과 구조 생성	opendataloader-pdf	읽기 순서 복원, 절/그림 분리, 쪽 장식 제거
목록에서 이름 끌어내기	PyMuPDF	단 구조 유지, 6~24배 빠름
통계치를 그대로 인용	두 가지로 교차 확인	부호/지수 손실 버그가 추출기마다 다름
큰 데이터 표	어느 쪽도 부적합	보통 자료의 표 파일을 직접 읽기

4. 조용히 성공하는 실패

잘했는데도 '성공'으로 보고되는 경우가 가장 위험해요...

KOREA

사례: 스크립트의 3번째 인자를 '쪽수 제한'이라 생각하고 작은 값을 넣었는데, 사실은 '글자 수 상한'!
→ Methods 중간에서 잘렸지만 스크립트는 성공 보고
→ 뒤의 결과와 고합이 '없는 것'으로 처리됨

추출 후, 반드시 3가지를 확인!

1. 끝이 있는가? (참고문헌/결론이 마지막에?)
2. 분량이 맞는가? (쪽수 대비 글자 수 비를 비교)
3. 숫자가 살아 있는가? (통계치 2~3개 읽은 대조)

도구의 성공 메시지 ≠ 내용의 정확성!

5. 추출 결과를 다듬는 범위

opendataloader-pdf 출력은 다듬어서 사용해요!

- ✓ 그림 파일은 /figures에 별도 저장
- ✓ 그림 설명은 본문에 인라인으로 연결
- ✓ 참고문헌은 색선 끝에 정리
- ✓ 쪽 장식과 하이픈 노이즈 제거
- ✓ 절 제목의 번호/서식 정리

다듬기는 의미를 바꾸는 편집이 아니에요. 읽기와 검색을 좋게 만드는 정리예요!

요약하면,

용도에 맞게 추출기를 고르고, 추출 후엔 꼭 확인하고, 필요한 만큼 다듬어 쓰기!

이게 위키 품질의 비결이에요!

따라해보기 (프로프트)

아래 프로프트를 시 도구에 붙여 넣고 두 추출기의 차이를 직접 확인해 보세요!

아래 PDF를 두 가지 방법으로 추출해서 결과를 비교해 달라.
<PDF 경로>

1. opendataloader-pdf로 추출한 결과
2. PyMuPDF로 추출한 결과

두 결과에서 다음을 비교해 보여 달라.

- Methods 절의 첫 세 문단이 각각 어떻게 나오는지
- 그림 설명이 본문 문단 사이에 섞였는지
- 본문에 등장하는 용수와 지수 표기가 그대로 남았는지
- 마지막 참고문헌 번호가 몇 번인지

어느 쪽이 더 나은지 판정하지 말고 차이만 보여 달라.

마지막 참고문헌 번호 → 잘림(누락) 확인!
(쪽 번호가 다르면 왼쪽이 중간에서 잘림 것)

Figure 4: 논문 PDF를 고르고 추출하고 검증하는 전체 흐름

논문에서는 멀쩡하고 어떤 논문에서는 무너지므로 추출기를 한 번 검증했다고 안심할 수 없다. 결론을 좌우하는 수치는 논문마다 화면과 대조한다는 규칙이 여기서 나온다. 대조할 대상은 많지 않다. 논문 하나에서 실제로 인용하게 될 값은 대개 서너 개다.

추출기를 고르는 두 갈래

이 책이 근거로 삼은 위키에서는 2026년 5월 30일에 추출기를 실제로 비교했다. 같은 논문들을 두 추출기에 각각 넣고 결과를 나란히 놓는 방식이었고 대상은 단일세포 atlas 논문과 두 단으로 조판된 종합지 논문이었다. 결론은 용도에 따라 답이 갈린다는 것이었다. 요약과 구조를 만드는 일상적인 ingest에서는 opendataloader-pdf가 뚜렷하게 나왔다. 읽기 순서를 다시 짜서 두 단이 섞이는 문제를 해결하고 그림 패널에 흩어진 축 눈금과 날개 라벨이 본문에 섞여 들어오는 것을 막고 절 제목을 제목으로 표시하고 그림 설명을 따로 떼고 참고문헌을 정돈하고 쪽마다 붙는 저널 이름과 하이픈 노이즈를 걷어냈다. 같은 논문에서 PyMuPDF는 두 단을 흘려 일부 구간이 아예 판독되지 않았다.

두 번째 용도에서는 답이 반대였다. 논문에서 유전자 이름과 유전자 사이의 관계를 뽑아내는 작업에서는 PyMuPDF로 충분했다. 유전자를 찾아낸 비율이 226건 대 221건으로 사실상 같았고 뽑아낸 관계의 수도 15건으로 같았다. 오히려 PyMuPDF 쪽이 유전자 목록을 단 단위로 흐트러뜨리지 않고 유지해서 어느 표의 어느 줄에서 나온 이름인지 되짚기가 조금 더 깨끗했다. 속도 차이도 컸다. PyMuPDF가 6배에서 24배 빨랐고 Java 실행 환경도 필요하지 않았다. 이 비교를 실측이라고 부른 이유는 같은 논문을 두 도구에 각각 넣고 결과를 나란히 놓았기 때문이다. 도구의 설명서에 적힌 성능이나 일반적인 평판으로 정하지 않았다. 추출기의 성능은 논문의 조판에 따라 갈리므로 자기가 주로 읽는 저널의 논문으로 직접 확인하지 않으면 남의 결과가 자기 자료에서 재현되지 않는다. 두 단으로 조판하는 저널과 한 단으로 조판하는 저널은 난이도가 다르고 그림이 본문 사이에 끼어드는 형식과 뒤에 몰아 두는 형식도 다르다. 자기 자료로 비교하는 데 드는 시간은 논문 두세 편이면 충분하다. 용도에 맞춰 추출기를 고르는 것이 여기서 얻을 원칙이다. 문장을 읽어 요약과 판단을 만드는 일에는 읽기 순서를 복원하는 추출기가 맞고 목록에서 이름을 긁어내는 일에는 지면 배치를 그대로 유지하는 추출기가 맞다.

용도	맞는 추출기	이유
요약과 구조 생성	opendataloader-pdf	읽기 순서 복원, 절 제목과 그림 설명 분리, 쪽 장식 제거
목록에서 이름 긁어내기	PyMuPDF	단 구조를 흐트러뜨리지 않고 유지, 6-24배 빠름
통계치를 그대로 인용	두 가지로 교차 확인	부호와 지수가 사라지는 버그가 추출기마다 다름
큰 데이터 표	어느 쪽도 부적합	보충 자료의 표 파일을 직접 읽는다

추출로는 아예 얻을 수 없는 것도 있다. 그림이 담고 있는 정보다. 히트맵의 색이 나타내는 방향과 크기, 효과 크기를 점과 선으로 그린 그림, 모델 구조를 그린 도해는 텍스트로 존재하지 않으므로 어느 추출기로도 나오지 않는다. 그림 설명에 적힌 문장만으로 그 그림이 무엇을 보여주는지 알 수 있는 경우도 있지만 결론이 그림 안에 있는 논문에서는 그렇지 않다. 여기에는 다른 방법을 쓴다. 그 쪽을 이미지로 떼서 모델이 직접 보게 하는 것이다. 이미지를 읽는 처리는 텍스트를 읽는 것보다 비싸고 느리므로 모든 논문에 적용하지 않는다. 분야의 논지를 움직이는 논문에 한정하고 일상적인 처리에서는 하지 않는다. 이 구분을 정해 두지 않으면 논문 한 편을 넣는 데 드는 시간이 예측되지 않는다.

따라해보기

아래 블록은 AI 도구의 채팅창에 그대로 붙여 넣는 프롬프트다. 논문 한 편으로 두 추출기의 차이를 직접 확인해 보는 것이 이 장에서 가장 빠른 실습이다. 두 단으로 조판된 논문을 하나 고른다.

```
PDF
<PDF >
```

1. opendataloader-pdf
2. PyMuPDF

- Methods

-
-
-

마지막 줄을 넣는 이유는 판정을 사람이 하기 위해서다. 어느 쪽이 나은지는 이 논문으로 무엇을 할 것인지에 달려 있고 그 판단은 결과를 눈으로 본 뒤에 선다. 마지막 참고문헌 번호를 비교하라고 한 것은 잘림을 잡기 위해서다. 참고문헌은 논문의 맨 끝에 있으므로 두 결과에서 마지막 번호가 다르면 한쪽이 중간에서 잘린 것이다.

조용히 성공하는 실패

추출이 실패했다는 것을 도구가 알려 주지 않는 경우가 실제 운영에서 가장 비싸다. 같은 위키에서 있었던 일이 이 형태였다. 추출 스크립트의 세 번째 인자가 쪽수 제한이라고 생각하고 작은 값을 넣었는데 그 인자는 글자 수 상한이었다. 논문이 Methods 중간에서 잘렸고 스크립트는 성공을 보고했다. 잘린 뒤쪽에 있던 결과와 고찰은 그 뒤의 모든 작업에서 없는 것으로 다뤄졌고 빠졌다는 사실은 결과물이 이상해진 다음에야 드러났다. 도구의 성공 메시지를 내용의 정확성으로 읽지 않는다는 규칙이 여기서 나왔다. 추출을 마치면 결과물의 모양을 확인한다.

이 사고가 알려 주는 것이 하나 더 있다. 도구의 인자가 무엇을 뜻하는지 확인하지 않고 값을 넣었다는 점이다. 숫자를 받는 인자에 작은 값을 넣으면 쪽수 제한처럼 동작할 것 같지만 그 도구에서는 글자 수 상한이었다. 도구를 처음 쓸 때 인자의 의미를 한 번 확인하는 데 드는 시간은 1분이고 확인하지 않아서 생긴 결과를 되짚는 데는 그보다 훨씬 오래 걸린다. 특히 위험한 것은 그 값이 있어도 동작하고 없어도 동작하는 인자다. 잘못 넣어도 오류가 나지 않으므로 잘못 넣었다는 사실이 드러나지 않는다. 확인은 세 가지를 보면 충분하다. 첫째는 끝이 있는가다. 참고문헌 목록이나 결론 문단이 결과 파일의 끝에 있는지 본다. Methods 중간이나 문장 중간에서 끝났다면 잘린 것이다. 둘째는 분량이 맞는가다. 논문의 쪽수와 추출된 글자 수의 비율이 다른 논문들과 크게 다르면 무언가 빠진 것이고 대체로 추출에 실패한 논문이 여기에 걸린다. 셋째는 숫자가 살아 있는가다. 본문에 나온 통계치 두어 개를 원문 화면과 대조한다. 부호와 지수가 이 대조에서 걸린다. 세 가지 모두 자동으로 판정할 수 있는 종류가 아니고 그래서 6장에서 여러 편을 한 번에 넣을 때 이 확인을 어떻게 기계 검사로 바꾸는지를 따로 다룬다.

추출 결과를 다듬는 범위

PDF 추출 도구인 opendataloader-pdf의 출력에는 손봐야 할 것이 남는다. 그림이 있던 자리에 남는 표시와, 칸 안에 아무 내용도 없는 표 모양의 격자가 대표적이다. 뒤의 것은 이 추출기가 지면의 선을 표로 잘못 읽어서 만드는 것이고 그대로 두면 정리 문서에 빈 표가 들어간다. 그래서 정리한 뒤에 넘기는데 다듬는 범위를 문서로 못 박아 두었다. 지울 수 있는 것은 정확히 그 두 가지뿐이고 목록에 없는 것은 남긴다. 글자나 숫자가 하나라도 든 표는 모양이 이상해도 보존한다. 범위를 열어 두면 다듬기가 편집이 되고 편집이 된 순간 원문에 있던 것과 정리 과정에서 사라진 것을 구분할 수 없게 된다. 이 경계는 3장에서 본 층 나누기와 같은 원리다. 원문 PDF는 고치지 않고 추출 결과는 정해진 범위에서만 다듬고 판단은 그 위층에서 한다.

다듬기의 범위를 문서로 적어 두는 방식도 눈여겨볼 만하다. 지울 수 있는 것의 목록을 적고 그 목록에 없는 것은 남긴다고 쓰면, 나중에 새로운 종류의 노이즈가 나타났을 때 처리 방식이 자동으로 정해진다. 목록에 없으므로 남긴다. 반대로 지우면 안 되는 것의 목록을 적었다면 새로운 노이즈는 지워도 되는 것이 되고 그 판단이 세션마다 달라진다. 무엇을 허용할지 적는 것과 무엇을 금지할지 적는 것은 목록에 없는 경우에 대한 기본값이 다르고 되돌릴 수 없는 동작에서는 앞의 방식이 맞다. 같은 발상이 4장의 정정 공고 걸러 내기에도 있었다. 걸러 내는 조건을 좁게 잡으면 통과하는 공고가 생기지만 그 실패는 사람 눈에 드러나고 넓게 잡으면 정상 논문이 사라지는데 그 실패는 드러나지 않는다.

추출 도구 기록

추출에 쓴 도구를 문서에 기록한다. 정리 문서와 논문 페이지의 머리말에 입력이 어떤 형태였는지, 어느 도구의 어느 판으로 뽑았는지, 언제 뽑았는지를 적는다. 세 항목을 적는 이유는 나중에 다시 뽑을지 판단해야 하기 때문이다. 추출기는 계속 고쳐지고 몇 달 전에 부호를 떨어뜨리던 버그가 지금 판에서는 없을 수 있다. 어느 문서가 어느 판으로 만들어졌는지

적혀 있으면 그 버전으로 만든 문서만 골라 다시 뽑을 수 있고 적혀 있지 않으면 전부 다시 뽑거나 전부 그대로 두는 선택만 남는다. 손으로 읽어서 정리한 문서에도 그 사실을 적는다. 자동으로 뽑은 것과 사람이 읽은 것은 신뢰 수준이 다르고 몇 달 뒤에 그 차이는 문서 겉면에서 보이지 않는다. 4장에서 정리 문서에 원문을 가리키는 정보를 함께 적어 둔 것과 같은 이유다. 되짚어 갈 수 있게 만드는 항목은 만들 때 적어야 하고 나중에 채우려면 그때의 상태를 복원할 수 없다.

PDF에서 정보를 다시 뽑아야 하는 시점

PDF에서 정보를 다시 뽑는 이유는 넷이다. 첫째는 추출기의 결함이 나중에 알려진 경우다. 부호가 떨어지는 버그를 알기 전에 만든 문서에는 그 오류가 그대로 들어 있고 그 문서들만 골라내려면 어느 도구로 만들었는지가 기록되어 있어야 한다. 둘째는 더 나은 도구가 생긴 경우다. 두 단 조판을 제대로 복원하지 못하던 시기에 만든 문서와 지금 만든 문서는 품질이 다르고 그 차이는 문서 겉면에서 보이지 않는다. 셋째는 처음에 잘못 뽑은 것을 발견한 경우다. 잘려 있거나 다른 논문이 섞여 있거나 애초에 다른 판본이었던 경우가 여기 해당한다. 넷째는 위키 검색으로 질문에 답할 만큼 자세한 근거를 찾지 못한 경우다. Source note와 wiki page에는 논문의 주요 정보가 들어 있지만 원문의 모든 방법, 표, 수치, 참고문헌까지 옮기지는 않는다. 검색 결과만으로 답을 만들 수 없으면 웹 검색으로 빈칸을 채우지 않고 papers/에 보관한 원본 PDF의 전문을 다시 추출해 필요한 대목을 읽는다. 확인한 세부 정보는 source note와 wiki page에 보충해 같은 질문에서 다시 PDF로 돌아가는 일을 줄인다.

질문의 답이 해당 논문이 인용한 중요한 참고문헌에 달려 있는 경우도 있다. 먼저 참고문헌의 제목과 저자와 DOI를 확인하고 그 논문의 PDF가 위키에 있는지 찾는다. PDF가 있다면 전문을 추출해 읽고 별도의 source note와 wiki page로 ingest한다. PDF가 없다면 현재 위키에 그 근거가 없다고 밝히고 원문 PDF를 요청한다. 초록, 출판사 웹페이지, 검색 결과의 요약문을 원문 PDF처럼 사용하지 않는다. 이 절차는 3장의 네 가지 원칙을 그대로 따른다. 위키에서 먼저 찾고, 부족하면 소유한 PDF로 내려가며, 필요한 논문이 없으면 없다고 말한다. 답을 만든 뒤에는 PDF에서 확인한 내용을 위키에 반영해 다음 검색의 범위를 넓힌다.

문서 수천 건을 전부 다시 뽑으면 그 뒤에 사람이 손댄 부분까지 사라진다. 정리 문서에는 추출 결과를 읽고 보탠 판단이 들어 있으므로 다시 쓰기 전에 실제로 인용되는 값을 확인해야 한다. 종합 문서가 어떤 수치를 근거로 삼고 있다면 그 수치부터 원문과 대조한다. 아무 데도 인용되지 않은 문서는 그 논문이 필요해질 때 확인하면 된다. 무엇을 다시 뽑았고 무엇을 남겨 두었는지는 작업 로그에 적는다. 적지 않으면 다음에 같은 판단을 처음부터 다시 하게 된다.

다시 뽑기로 결정한 두 시점은 모두 표본 검사에서 나왔다. 2026년 7월 25일에 정보으로 저장해 둔 PDF 서른여덟 건이 그 논문의 보충 파일로 드러났다. 정리 문서와 위키 페이지에는 방법과 결과가 적혀 있었지만 그 내용은 보충 그림 설명과 보충 노트에서 나온 것이었고 검색에 걸리면 제대로 만든 페이지와 구별되지 않았다. 한 건은 아예 다른 논문이었다. 판별이 어려웠던 이유는 첫 쪽이 구별의 근거가 되지 못하기 때문이다. 대형 저널의 보충 파일 표지는 논문의 제목 블록을 그대로 재현하고 그 아래에 저자가 준 형식 그대로이며 편집을 거치지 않았다는 한 줄을 붙인다. 첫 쪽에 제목과 저자가 보인다는 것으로 논문이라고 판정하면 걸리고 이 착각으로 네 건을 정상으로 통과시켰다가 되돌렸다. 서른여덟 건 가운데 서른 건은 정상 PDF로 교체했고 페이지 본문까지 다시 쓴 것은 그중 한 건이다.

이틀 뒤에는 이미 만들어 둔 기록을 무작위로 이백 건 뽑아 확인했다. 다른 추출기로 원문을 독립적으로 다시 읽어 이백 건 모두 논문 신원이 일치하는 것을 확인했고 결함이 확인된 것은 스무 건이었다. 그중 심각한 실패는 세 건으로 두 건은 출력력이 크게 잘렸고 한 건은 전처리에서 멈췄다. 나머지는 기호와 연산자가 다른 글자로 바뀐 경우가 열여섯 건, 단어 경계와 배치가 무너진 경우가 한 건이었다. 이 결과에서 판단을 바꾼 것은 기호 문제의 원인이다. 같은 실패가 다른 추출기에도 그대로 보였고 원인은 PDF에 심긴 폰트 인코딩이었으므로 도구를 바꿔서 해결되는 문제와 원본 파일에서 오는 문제를 구분해야 했다. 이 둘을 섞으면 전부 다시 뽑는 쪽으로 결론이 기울고 그 작업은 몇 주가 걸리면서 고쳐지는 것이 없다. 그래서 다시 뽑는 대상은 신원이 어긋난 파일과 크게 잘린 출력으로 한정했고 기호가 어긋난 문서는 그 값을 실제로 인용할 때 원문과 대조하는 쪽으로 넘겼다.

6장. 논문 PDF 여러 개를 ingest하기

학회에서 돌아오면 읽을 논문이 스무 편 생기고 리뷰 논문 하나의 참고문헌을 따라가면 마흔 편이 생기며 새 주제를 시작하면 지난 3년치를 한꺼번에 훑게 된다. 스무 편부터는 같은 논문이 이미 들어 있는지 확인하고 넣지 않기로 한 저널의 논문을 거르고 추출이 조용히 실패한 파일을 찾아내며 모든 논문이 종합 층과 이어졌는지 확인하는 일이 한꺼번에 생긴다. 품질 기준은 4장에서 논문 한 편에 적용한 것과 같다. 여러 편을 처리할 때는 그 기준이 모든 논문에 적용되었는지를 따로 검사한다.

후보 목록을 먼저 확정하고 중복과 제외 대상을 검사한 뒤 완료·제외·미검사 수를 함께 보고하는 배치 절차다.

6장.

논문 PDF 여러 개를 ingest하기

한 편과 같은 품질 기준을, 여러 편에 적용하는 법

배치 크기는 달라도 품질 기준은 동일해요!

- ✓ 등급은 없어요. 한 편이든 서른 편이든 동일!
- ✓ 달라지는 것은 '확인'의 방식'뿐
- ✓ 확인하지 않은 건 확인했다고 적지 마세요!

“ 규모가 커질수록, 확인을 사람의 성실함에서 기계 검사로!”

1. 이미 가지고 있는 논문에서 시작하기

1 references 폴더 찾기

EndNote, Zotero, Mendeley 등 라이브러리와 하위 폴더까지 탐색

2 PDF 전부 찾기

숫자 폴더도 끝까지 내려가요!

3 목록과 통계 확인

전체 건수, 제목·저자·연도 표, 중복 유무까지!

따라해보기 (프롬프트)

내 컴퓨터에서 'references'라는 폴더를 찾아 달라. 그 안에 있는 EndNote 라이브러리와 PDF를 확인해서 목록으로 만들어 달라.

- ✓ 하위 폴더도 끝까지 탐색
- ✓ 찾기 못하거나 권한 필요하면 알려 달라.
- ✓ 같은 이름 폴더는 후보 제시
- ✓ 찾은 뒤엔
 - 전체 PDF 건수
 - 제목·저자·연도 표
 - 중복으로 보이는 묶음

여기까지 보여주고 멈춰 달라!

2. 배치를 열기 전에 확정하는 목록

기억할 것

조용히 성공하는 실패
글자 수 상한을 폭수 제한으로 착각하면 Methods 중간에서 잘려도 '성공'으로 보고돼요!
→ 참고문헌 끝, 분량 비율, 숫자 대조로 확인!

두 층을 나는 이점
PDF 수와 source note 수를 따로 셀 수 있어 남은 논문(미워힘)을 항상 파악할 수 있어요.
(예: 16,127 vs 15,995 = 132건)

No.	제목	연도	DOI	유형
1	...	2024	10.1038/...	논문
2	...	2023	10.1016/...	논문
3	...	2024	10.1101/...	preprint
:	...	...	...	...
N	...	2022	10.1007/...	논문

결정할 것들

- ✓ 어떤 논문을 넣을 것인가 (목록 확정!)
- ✓ 카테고리: Primary / Review / Method / Other
- ✓ 자료 출처: 출판본 / 프리프린트 / 기타
- ✓ 누가 할 것인가?
→ 논문 해석과 문서 작성은 이 에이전트(단일 프로세스)가 직접!

3. 중복과 제외 대상을 거르는 두 겹의 검사

1차: 기계 검사 (제목·DOI)

중복 검사 (DOI 기준)

제외 후보 식별 (중장·절화·제외 저널)

2차: 본문 읽기 검사 (브랜딩 + 저자 문구)

알 두 쪽 출판사 브랜딩 확인

본문 속 저자 전술 확인

→ 오란을 막기 위한 확실한 검증!

4. 작업이 끝나면 숫자로 보고하기

완료 (원고 문서 작성)

제외 (중복·제외 대상 등)

미검사 (수출 실패·빈 텍스트·시간 초과 등)

합이 N이 되도록 보고!

5. 배치의 목적은 숫자가 아니라, 정확한 지식이다

- ✓ 남은 논문(미워힘)을 줄이기
- ✓ 모든 문서가 종합 층과 연결되도록
- ✓ 틀리지 않은 문장으로 미래의 나를 돕기

“ 정확한 지식이 쌓여야, 연구가 앞으로 나아갑니다! ”

Figure 5: 여러 논문 PDF를 배치로 ingest하는 절차

한 편과 여러 편 사이에서 달라지는 것

여러 편을 넣을 때도 한 편과 같은 품질 기준을 적용한다. 4장에서 본 대로 ingest에는 등급이 없고 한 편씩 넣은 논문과 서른 편을 한꺼번에 넣은 논문이 위키 안에서 똑같이 생겼기 때문에 기준을 나누는 순간 어느 페이지가 대충 만들어졌는지 알 수 없게 된다. 배치는 실행 전략일 뿐이고 그 전략은 작업 로그 안에서만 존재해야 한다. 독자가 읽는 페이지에 배치 이름이나 날짜별 식별자가 남으면, 그 페이지를 읽는 사람은 논문에 대한 정보와 작업에 대한 정보를 매번 스스로 갈라내야 한다. 실제로 달라지는 것은 확인의 방식이다. 한 편을 넣을 때는 사람이 그 한 편을 처음부터 끝까지 보고 있으므로 이상한 곳이 눈에 걸린다. 서른 편에서는 걸리지 않는다. 한 편이 잘못되었을 확률이 낮아도 서른 편 가운데 하나가 잘못되었을 확률은 낮지 않고 잘못된 하나가 어느 것인지는 눈으로 훑어서 나오지 않는다. 규모가 커지면 확인을 사람의 성실함에서 기계 검사로 옮겨야 한다.

이 이동을 하지 않으면 어떤 일이 생기는지는 기록으로 남아 있다. 2026년 7월 25일에 이 책이 근거로 삼은 위키에서 PDF 4,236건을 검수해 달라는 요청이 있었다. 작업을 맡은 에이전트는 휴리스틱으로 후보를 좁힌 뒤 서른 건만 눈으로 보고 결과를 보고했다. 같은 날 다른 작업에서는 11,029건을 확인했다고 말했는데 실제로 확인한 것은 제외 저널을 가려내는 지문 하나뿐이었다. 두 보고 모두 숫자가 붙어 있었고 그래서 전수를 마친 것처럼 읽혔다. 배치에서 가장 자주 일어나는 실수는 확인하지 않은 것을 확인했다고 적는 것이다. 그 뒤에 만들어진 규칙은 표본으로 끝내지 말라는 금지에서 멈추지 않고 분모와 분자를 함께 밝히라는 요구까지 간다. 몇 건 중 몇 건을 어떤 방법으로 확인했는지를 같은 문장에 적으면 표본 검사를 전수 검사로 잘못 보고하기 어려워진다.

배치가 남긴 잔여물은 숫자로도 보인다. 같은 위키에는 2026년 8월 기준 원문 PDF 16,127건과 source note 15,995건이 있고 차이 132건은 폴더에 넣었지만 아직 읽히지 않은 논문이다. 이 차이는 대부분 배치에서 생긴다. 한 편씩 넣을 때는 PDF를 복사하는 일과 읽는 일 사이에 시간이 벌어지지 않지만 서른 편을 복사해 두고 스물여덟 편을 처리한 뒤 두 편에서 막히면 그 두 편이 그대로 남는다. 남은 것을 세지 않으면 다음 배치에서 또 두 편이 남고 몇 달이면 백 편이 된다. 두 층을 나눠 둔 이익이 여기서 나온다. PDF 수와 source note 수를 따로 셀 수 있으므로 남은 것이 몇 건인지가 매번 드러나고 그 차이가 몇 달째 같은 수에 머물러 있다면 넣기만 하고 읽지 않는 습관이 굳은 것이다.

이미 가지고 있는 논문에서 시작하기

처음 위키를 만들 때는 이미 모아 둔 논문부터 가져오면 된다. 논문을 오래 읽어 온 사람이라면 수백 편의 PDF가 EndNote, Zotero, Mendeley 같은 문헌관리 프로그램에 들어 있다. 사용자는 그 파일이 컴퓨터의 어느 경로에 저장되어 있는지 모를 수 있다. references라는 폴더 이름만 기억하는 경우도 흔하다. 이때 에이전트에게 “내 컴퓨터에서 references라는 폴더를 찾아 그 안의 EndNote 라이브러리와 PDF를 확인해 줘”라고 말하면 된다. 에이전트는 접근이 허용된 범위에서 폴더를 검색하고 문헌관리 프로그램이 만든 하위 폴더까지 내려가 PDF 후보를 모은다. 같은 이름의 폴더가 여러 개 나오면 후보 목록을 보여 주고 사용자가 고르게 한다. 폴더를 찾지 못하거나 현재 권한으로 열 수 없으면 필요한 폴더를 추가하거나 접근을 허용해 달라고 요청한다. 사용자는 자신이 기억하는 폴더 이름과 그 안에 있는 자료의 종류만 말해 주면 된다.

문헌관리 프로그램은 첨부 파일을 숫자나 식별자로 만든 여러 하위 폴더에 나누어 보관한다. 에이전트가 각 프로그램의 내부 구조를 따라 PDF를 찾아가므로 사용자가 숫자 폴더를 하나씩 열어 볼 일은 없다. 먼저 찾은 파일 목록과 전체 건수를 보여 달라고 한다. 같은 논문의 프리프린트와 출판본을 모두 저장했거나 라이브러리를 합치는 과정에서 같은 파일이 두 번 들어간 경우가 흔하기 때문이다. 각 PDF의 첫 쪽에서 제목, 저자, 연도를 읽으면 어떤 논문이 들어 있는지 확인할 수 있다. 제목이나 DOI가 같은 파일은 한 묶음으로 표시해 달라고 한다. 문헌관리 프로그램의 서지 정보는 프로그램마다 형식이 다르므로 위키에 넣을 정보는 PDF에서 다시 읽는다. 목록을 확인한 뒤 지금 쓰고 있는 주제의 논문부터 골라 첫 배치를 시작한다.

```
`references`  
EndNote  
PDF  
  
`references`
```


프롬프트에 중간 확인을 넣으면 사람이 중복과 제외 대상을 직접 볼 수 있다. 이 판단이 틀리면 걸러 낸 논문은 위키에 들어오지 않은 채 사라지고 걸러 내지 못한 논문은 잘못된 근거로 남는다. 전문 추출이 시작된 뒤에는 되돌리는 값이 커지므로 확인은 그 앞에 둔다.

중복과 제외 대상을 거르는 두 겹의 검사

중복 검사는 DOI를 기준으로 하고 범위는 위키 전체다. 파일명으로 검사하면 같은 논문이 다른 이름으로 두 번 들어오는 것을 놓치고 제목으로 검사하면 부제나 대소문자 차이에서 어긋난다. 검사 대상은 원문을 읽은 기록과 논문 페이지 양쪽이다. 한쪽에만 있는 상태가 실제로 생기고 그 상태는 앞 절에서 본 132건의 차이와 같은 종류다. 제외 대상을 거르는 일은 두 겹으로 되어 있다. 첫 겹은 제목과 DOI를 보는 기계 검사다. 정정 공지와 철회 공지는 제목이 정해진 문구로 시작하므로 제목만 보고 걸러진다. 놓지 않기로 한 저널은 DOI 접두어로 걸러진다.

첫 검사만으로 놓치는 경우가 있다. DOI는 그 파일이 어디서 왔는지를 말하고 논문이 결국 어디에 실렸는지를 말하지 않기 때문이다. 프리프린트 서버의 DOI를 달고 들어온 논문은 그 프리프린트가 나중에 제외하기로 한 저널에 실렸더라도 DOI 검사를 그대로 통과한다. 2026년 7월 25일에 실제로 그 경로로 한 건이 ingest 직전까지 갔다. 그 파일에서 유일한 단서는 본문 감사의 글에 적힌 한 줄이었고 저자가 자기 투고를 두고 특정 출판사의 지침과 정책을 따랐다고 밝힌 문장이었다. 그래서 두 번째 겹으로 본문을 읽는 검사를 둔다. 이 검사는 문서를 두 구역으로 나눠 본다. 앞 두 쪽에 찍힌 출판사 브랜딩과, 문서 어디에 있는 저자가 자기 투고에 대해 쓴 문구다. 두 구역은 참고문헌에 그 출판사의 논문이 인용되어 있다는 이유만으로는 걸리지 않도록 고른 것이다.

두 번째 겹에도 한계가 있고 그 한계는 적어 두어야 한다. 프리프린트 PDF 가운데 상당수는 출판사 템플릿도 투고 문구도 없는 순수 원고여서 그 논문이 어느 저널로 갈지에 대한 흔적이 파일 안에 없다. 그런 파일은 본문 검사로도 잡히지 않고 확실히 하려면 프리프린트의 DOI로 출판된 판본을 조회해야 한다. 조회는 바깥 네트워크에 나가는 동작이므로 사람이 지시했을 때만 한다. 규칙 문서에는 이 세 층의 검사로도 놓칠 수 있는 경우를 적어 검사 결과의 한계를 밝힌다. 배치를 달기 직전에는 두 겹의 검사를 다시 돌려 매치가 0줄인지 확인한다. 매치가 나오면 해당 문서와 원문과 논문 페이지를 지우고 그것을 가리키던 링크를 정리한 뒤 다시 검사한다. 통과한 배치와 검사하지 않은 배치는 로그에서 구분되어야 하고 검사를 돌리지 않았다면 돌리지 않았다고 적는다.

조용히 실패하는 추출

추출은 실패해도 실패처럼 보이지 않는다. 같은 위키에서 실제로 있었던 일 하나가 이 형태였다. 텍스트 추출기가 가진 글자 수 상한을 꼭수 제한으로 착각한 채 논문을 넣었더니 본문이 Methods 중간에서 잘렸는데 추출은 성공으로 보고되었다. 잘린 뒤쪽에 있던 내용은 그 뒤의 모든 작업에서 없는 것으로 다뤄졌다. 한 편을 넣을 때는 사람이 그 문서를 읽다가 결론 부분이 없다는 것을 알아차리지만 스무 편에서는 알아차리지 못한다. 기호가 사라지거나 다른 글자로 바뀌는 문제도 있다. 출판사가 기호를 유니코드 매핑이 없는 서브셋 폰트에 심으면 추출기 두 종류 모두 원래 글자를 복원하지 못하고 글자가 그냥 빠지는 경우와 다른 글자로 바뀌는 경우가 있으며 뒤쪽이 더 위험하다. 빠진 글자는 문장이 이상해져서 보이지만 바뀐 글자는 그럴듯한 문장으로 남는다.

숫자에서도 같은 일이 일어난다. 추출기 하나에는 음수 부호를 떨어뜨리는 버그가 있어서 음의 상관계수가 양수로 바뀌어 나온다. 지수 표기가 깨져 자릿수가 달라지는 경우도 있다. 통계치를 원문 그대로 인용할 때 다른 추출기로 교차 확인하라는 규칙이 여기서 나왔다. 이런 실패를 배치에서 다루는 방식은 하나다. 추출에 실패했거나 결과가 비었거나 시간이 초과된 파일은 통과로 세지 않고 미검사로 센다. 미검사 건수를 0으로 만들고 싶은 유혹이 배치의 마지막에 반드시 생기는데 미검사로 적힌 항목은 나중에 다시 열리고 통과로 적힌 항목은 다시 열리지 않는다. 셸에서 파일 경로를 받는 반복문을 쓸 때 변수 이름을 path로 두지 않는 것도 배치에서만 드러나는 함정이다. 이 이름은 시스템의 실행 경로와 묶여 있어서 한 줄로도 기본 명령이 사라지고 그러면 그 뒤의 파일들이 처리되지 않은 채로 반복문만 돌아간다.

배치에서도 건너뛰지 않는 종합 연결

배치에서 가장 먼저 생략되는 것이 종합 총 연결이다. 스무 편의 원문을 읽고 스무 쌍의 문서를 쓰고 나면 작업이 끝난 것처럼 느껴지고 각 페이지를 기존 종합 문서와 잇는 일은 스무 번 더 해야 하는 일로 남는다. 여기서 멈추면 4장이 이름 붙인 상태가 스무 개 생긴다. 종합 총과 연결되지 않은 원문 기록과 논문 페이지 한 쌍은 겹으로 정상이고 검색에도 걸리지만 위키가 하려던 일은 무너져 있다. 배치의 최소 완료 조건은 한 편씩 넣을 때와 같다. 각 논문 페이지에서 가장 관련된 개관이나 개념 페이지로 양방향 링크를 걸고 그 종합 페이지의 본문에 이 논문이 어디에 놓이는지를 한 문장 이상

반영하고 기존 주장이 강화되는지 좁아지는지 반박되는지 대체되는지를 판정한다. 적절한 종합 페이지가 없으면 그 공백을 작업 로그에 적는다.

배치 직후에는 카테고리별 종합 연결 비율과 연결되지 않은 페이지 수를 세는 감사를 돌린다. 새로 생긴 연결 없는 페이지가 있거나 종합 페이지 본문이 갱신되지 않았으면 그 배치를 완료로 처리하지 않는다. 감사 결과가 카테고리마다 따로 나오므로 어느 주제에서 연결이 밀렸는지도 함께 보인다. 배치에서 하지 않는 것도 정해 두었다. 결론이 그림에 담긴 논문은 핵심 그림을 이미지로 떼서 직접 보는 처리를 쓰는데 이것은 논지를 바꾸는 논문에만 적용하고 일상적인 배치에서는 하지 않는다. 서른 편을 훑는 속도와 한 편을 느리게 읽는 방식을 같은 작업 안에 섞으면 둘 다 어중간해진다. 배치에서 그 논문이 그런 종류라는 것이 드러나면, 그 한 편을 배치에서 빼내 따로 처리한다.

실제 배치는 확보한 목록에서 자격을 갖춘 논문을 세어 그 수를 분모로 고정하고 그 분모를 끝까지 끌고 간다. 2026년 7월에 돌린 두 배치가 각각 서른두 편과 백예순세 편이었고 둘 다 종합 연결을 100퍼센트로 마쳤다. 백예순세 편 배치에서는 이백 건짜리 후보 목록을 열려 두고 그중 본문 PDF 자격을 갖춘 163건을 분모로 잡았으며 처리 결과는 새로 들인 162편과 이미 있던 1편으로 나뉘고 미검사와 실패가 0건이었다. 이 크기는 주제가 집중되어 있었기 때문에 가능했다. 163편 중 111편이 한 카테고리에 속했고 대부분이 같은 개관 페이지에 붙었으므로 종합 층에서 연 문서 수는 논문 수보다 훨씬 적었다. 같은 배치에서 코호트 안의 연결 없는 페이지는 0건이었고 코퍼스 전체로도 12,373건 중 12,362건이 연결되어 있었다. 남은 11건은 이 배치와 무관한 이전 것이었다. 분모를 고정하고 종합 연결과 감사를 배치의 종료 조건에 넣으면 배치 크기는 주제가 얼마나 모여 있는지에 따라 정해진다.

분모와 분자를 함께 적는 완료 보고

배치의 마지막 산출물은 완료, 제외, 미검사 수를 담은 보고다. 대상 N건 가운데 완료가 몇 건이고 제외가 몇 건이고 미검사가 몇 건인지를 적고 세 수의 합이 N이 되는지 확인한다. 합이 맞지 않으면 어딘가에서 논문 몇 편이 어느 카테고리에도 들어가지 못한 채 사라진 것이고 그 사실은 합을 세어 볼 때만 드러난다. 제외한 건은 제외한 이유를 함께 적는다. 정정 공지여서 뺀 것과 제외 저널이어서 뺀 것과 이미 위키에 있어서 뺀 것은 나중에 다시 볼 때 성격이 다르다. 미검사로 남은 건은 왜 미검사인지를 적는다. 추출이 실패한 것과 원문을 구하지 못한 것과 시간이 모자란 것은 다음에 할 일이 서로 다르기 때문이다. 이 보고는 그날의 작업 로그에 덧붙인다. 로그가 아닌 곳에 적으면 다음 세션의 에이전트가 그 배치의 결과를 읽지 못하고 읽지 못하면 미검사로 남은 건들은 아무도 다시 열지 않는다.

배치를 닫고 색인을 다시 만드는 순서

배치 중간에 검색 색인을 다시 만들지 않는다. 논문을 한 편 넣을 때마다 전체 색인을 다시 세우는 것은 낭비이고 재구축이 진행 중인 상태에서 검색이 들어오면 일부 문서를 놓친 결과가 나온다. 기본 검색의 신선도는 기계마다 도는 새벽 예약 작업에 맡긴다. 방금 넣은 논문을 곧바로 검색해야 할 때만 배치를 완전히 닫은 뒤에 사람이 직접 돌린다. 닫는 순서는 정해져 있다. 먼저 제외 대상 검사를 다시 돌려 매치가 0줄인지 확인하고 다음으로 종합 연결 감사를 돌려 새 연결 없는 페이지가 없는지 확인하고 그다음에 완료 보고를 작업 로그에 덧붙이고 마지막으로 카테고리별 문서 목록과 로그 색인을 다시 만든다. 순서를 뒤집어 색인부터 만들면 지워야 할 페이지가 색인에 들어간 채로 남는다. 의미 검색 쪽 색인은 사람이 명시적으로 요청했을 때만 갱신한다.

색인과 목록은 문서에서 다시 만들 수 있는 파생 상태이므로 정보와 다르게 다룬다. 파생 상태가 낡았을 때 고칠 것은 파생 상태이고 정보가 틀렸을 때 고칠 것은 정보이다. 둘을 혼동하면 검색 결과가 이상할 때마다 문서를 고치게 되고 문서는 멀쩡한데 색인만 낡은 경우가 대부분이므로 고칠 필요가 없던 문장이 고쳐진다. 배치 뒤에 검색이 이상하면 먼저 색인의 생성 시각을 본다. 이 구분과 재구축 시점은 9장에서 검색을 다루면서 다시 나온다.

배치가 위키의 구조로 굳는 현상

배치를 돌리고 나서 따로 검수해야 하는 이유가 하나 더 있다. 같은 배치에서 들어온 논문들끼리만 서로 연결되어 한 덩어리를 이루는 일이 생긴다. 종합 층에 연결하라는 조건은 지켜졌고 감사도 통과하는데 열어 보면 그 논문들이 가리키는 것이 대부분 같은 날 함께 들어온 논문이다. 원인은 단순하다. 스무 편을 나란히 놓고 작업하면 그 스무 편 사이의 관계가 가장 먼저 눈에 들어오고 위키에 이미 있던 수백 편과의 관계는 찾으려면 검색을 해야 한다. 쉬운 쪽으로 연결이 몰린다. 그렇게 만들어진 덩어리는 주제를 반영하지 않고 그날의 작업 단위를 반영한다. 몇 달 뒤에 그 주제로 질문을 던지면 이 덩어리만 통째로 올라오고 같은 주제를 다루는 예전 논문들은 따라오지 않는다.

같은 종류의 흔적이 문장에도 남는다. 배치 이름이나 날짜별 식별자가 페이지 본문에 적히는 경우, 여러 페이지에 같은 문장이 복사되어 들어간 경우, 그리고 그날 함께 처리한 논문들을 묶어 부르는 표현이 종합 문서에 들어간 경우가 여기 해당한다. 세 가지 모두 작업 사정에 속하므로 논문 정보에서는 뺀다. 검수는 배치가 끝난 뒤 조금 시간을 두고 하는 편이 낫다. 작업 직후에는 그 스무 편이 한 덩어리로 보이는 것이 자연스럽게 느껴지고 며칠 지나 다시 보면 그 묶음이 임의적이라는 것이 드러난다.

따라해보기

아래 블록은 AI 도구의 채팅창에 그대로 붙여 넣는 프롬프트다. 배치 검수는 다음 요청으로 시작한다.

: < >

1.

2.

3.

4.

2

마지막 줄을 넣는 이유는 연결이 판단이기 때문이다. 검색으로 올라온 문서가 실제로 관련이 있는지, 그리고 그 관계가 강화인지 반박인지는 두 문서를 읽어야 정해진다. 후보를 자동으로 걸어 두면 링크는 늘어나는데 그 링크가 무엇을 뜻하는지는 아무 데도 적히지 않는다. 4번에서 나온 표현은 지운다. 배치는 실행 전략이고 실행 전략은 작업 로그가 소유한다.

배치가 남기는 부채

배치를 돌리면 세 가지 부채가 남는다. 첫째는 넣었지만 읽지 않은 PDF다. 앞에서 본 132건이 이것이고 이 수는 배치를 돌릴 때마다 조금씩 늘는다. 둘째는 만들었지만 읽지 않은 페이지다. 종합 연결을 미룬 배치가 반복되면 연결 없는 페이지가 카테고리마다 쌓이고 어느 카테고리의 그 수가 유독 크다면 그 주제에 종합 문서가 부족하다는 신호다. 이 신호는 논문을 더 넣어서 해결되지 않고 종합 문서를 쓰는 일로만 해결된다. 셋째는 미검사로 남긴 건이다. 앞의 둘과 달리 이것은 정직하게 적어 둔 부채여서 다음 배치의 목록 맨 앞에 놓을 수 있다.

배치 크기의 상한은 종합 연결까지 끝낼 수 있는 논문 수로 정한다. 원문을 읽고 두 문서를 쓰는 일은 편수에 비례해 늘어나지만 종합 층에 반영하는 일은 그렇지 않다. 스무 편이 서로 다른 주제라면 스무 개의 종합 페이지를 열어 각각 한 문장씩 고쳐야 하고 그 스무 번은 논문을 읽는 일보다 지치는 작업이다. 한 배치를 한 주제로 묶으면 이 일이 줄어든다. 같은 개관 페이지에 붙을 논문들을 함께 넣으면 종합 문서 한 편을 열어 여러 논문을 한 번에 반영할 수 있고 그 과정에서 논문들 사이의 관계도 함께 적히기 때문이다. 주제를 섞은 배치는 처리 속도가 같아 보여도 종합 층에서 훨씬 비싸다.

실제로 쓰는 것은 주제별 묶기다. 백예순세 편 배치는 전이인자와 반복서열이라는 한 주제의 참조 목록에서 나왔고 그 결과 대부분의 논문이 같은 개관 페이지 하나에 연결되었다. 이렇게 묶으면 종합 문서에 반복서열의 교체 속도와 숙주의 억제와 조절 요소로의 전용과 유전체 구조와 생식세포에서의 통제라는 축이 생기고 각 논문이 그중 어느 축을 강화하거나 좁히는지가 적힌다. 같은 배치에서 기존 주장을 반박하거나 대체한 논문은 없었고 강화와 축소만 있었는데 이 판정 자체가 백 편을 한 주제로 모았기 때문에 가능했다. 시간순이나 학회순으로 묶으면 이 판정을 할 수 없다. 같은 달에 나왔다는 사실은 두 논문 사이에 아무 관계도 만들지 않으므로 종합 층에서는 논문마다 다른 문서를 열게 되고 그 작업은 편수에 비례해 늘어난다. 학회순 묶음이 유용한 경우는 그 학회를 훑는 것 자체가 목적일 때이고 그때도 종합 연결은 주제별로 다시 나눠서 한다.

7장. 논문 PDF를 확보하는 방법

앞의 세 장은 이미 손에 있는 PDF를 다뤘다. 위키가 자라기 시작하면 병목이 PDF 처리에서 논문을 고르고 확보하는 일로 옮겨 간다. 무엇을 읽을지 고르는 일과 그 논문의 원문을 실제로 손에 넣는 일이다. 앞의 것은 매주 쏟아지는 새 논문 가운데 자기 주제와 관련된 것을 골라내는 문제이고 뒤의 것은 고른 논문의 PDF를 정당한 경로로 받는 문제다. 두 가지를 손으로 하면 시간이 상당히 든다. 저널 목차 메일을 열어 제목을 훑고 관심 가는 것을 눌러 초록을 읽고 받을 수 있으면 받고 없으면 나중에 미루는 일을 매주 반복하게 된다. 후보를 모으는 일과 거르는 일은 자동으로 돌리고 무엇을 읽을지 정하는 판단과 원문을 확보하는 마지막 단계는 사람이 맡는다.

후보 탐색과 서지 정보 조회, 사람의 우선순위 판단, 원문 확보와 상태 기록을 거쳐 위키에 반영하는 전체 흐름이다.

서지 정보와 원문을 나누어 생각하기

확보 작업은 두 단계로 나뉘고 각 단계에서 다루는 것이 다르다. 앞 단계는 서지 정보다. 어떤 논문이 언제 어디에 나왔고 저자가 누구이고 무엇을 다루며 원문을 어디서 받을 수 있는지에 대한 정보다. 이 정보는 대체로 공개되어 있고 프로그램으로 받아올 수 있다. 뒤 단계는 원문 파일이다. 이쪽은 출판사의 정책과 소속 기관의 구독 상태에 따라 받을 수 있기도 하고 없기도 하다. 두 단계를 섞어 생각하면 작업이 막힌다. 서지 정보를 모으는 일은 수천 건도 몇 분이면 되지만 원문을 받는 일은 한 편씩 조건이 다르다. 그래서 후보를 모으는 단계에서는 원문을 받지 않고 목록만 만든다. 목록에는 그 논문을 받을 수 있는지에 대한 정보를 함께 담아 두고 실제로 받는 일은 그다음에 한다.

이 분리가 실제로 이익이 되는 이유는 대부분의 후보가 결국 읽히지 않기 때문이다. 한 주에 올라오는 관련 논문이 서른 편이면 그중 실제로 위키에 들일 것은 두세 편이다. 스물일곱 편의 원문을 먼저 받아 두면 폴더에 읽지 않은 파일이 쌓이고 6장에서 본 대로 넣기만 하고 읽지 않은 논문이 몇 달이면 백 편이 된다. 목록만 만들어 두면 그 스물일곱 편은 목록의 한 줄로 남고 나중에 필요해지면 그때 받는다. 목록에 남는 것도 값이 있다. 반년 뒤에 어떤 주제를 파기 시작할 때 그 목록을 다시 훑으면 이미 걸러진 후보가 거기 있다.

OpenAlex로 후보를 모으기

서지 정보를 프로그램으로 받아오는 경로 가운데 접근이 가장 쉬운 것이 OpenAlex다. 논문, 저자, 저널, 기관, 주제, 출판사, 연구비 지원기관을 담은 공개 목록이고 수억 건 규모다. 자료 전체가 개방 라이선스로 배포되어 있어서 받아 쓰는 데 제약이 없고 조회는 웹 주소 하나로 이루어진다. 논문 한 건의 정보에는 마흔 개가 넘는 항목이 들어 있는데 이 작업에서 실제로 쓰는 것은 몇 개다. 식별자와 제목과 출판일과 실린 저널, 그 논문이 몇 편을 인용했고 몇 번 인용되었는지, 그리고 공개 접근 여부와 원문 주소다.

공개 접근 정보가 이 목록의 가장 쓸모 있는 부분이다. 각 논문마다 공개 접근인지, 어떤 종류의 공개인지, 원문을 받을 수 있는 주소가 무엇인지가 함께 온다. 원문을 제공하는 곳마다 공개 여부, 웹 페이지와 PDF 주소, 이용 허락 조건, 출판본과 원고본의 구분이 적혀 있다. 이 항목들을 보면 그 논문을 지금 받을 수 있는지가 후보 목록 단계에서 이미 판정된다. 받을 수 있는 것과 없는 것을 미리 나눠 두면 실제로 받는 단계에서 막히는 일이 줄어든다.

따라해보기

아래 블록은 AI 도구의 채팅창에 그대로 붙여 넣는 프롬프트다. 한 편으로 시험해 보면 무엇이 오는지 감이 온다. 자기가 최근에 읽은 논문의 식별자를 하나 준비한다.

```
OpenAlex .
DOI: < DOI>
```

```
- , ,
-
- ,
-
-
```



Figure 6: 논문 후보 탐색부터 원문 확보와 위키 반영까지의 흐름

원고본과 출판본의 구분이 중요하다. 저자가 저장소에 올려 둔 원고본은 심사 뒤 수정 사항이 반영되기 전 판본일 수 있고 쪽 번호와 그림 번호가 출판본과 다르다. 인용할 때는 출판본을 기준으로 삼아야 하므로 원고본만 받을 수 있는 논문은 그 사실을 정리 문서에 적어 둔다. 이용 허락 조건도 함께 보는데 조건에 따라 그 파일을 어디까지 쓸 수 있는지가 달라진다.

PubMed와 프리프린트 서버

생명과학과 의학 분야에서는 PubMed가 또 하나의 축이다. 초록과 서지 정보를 프로그램으로 받아올 수 있고 주제어 체계가 붙어 있어서 특정 개념을 다루는 논문을 좀더 찾을 때 유리하다. OpenAlex와 겹치는 부분이 많지만 두 목록의 수록 범위가 완전히 같지는 않아서 한쪽에만 있는 논문이 생긴다. 프리프린트 서버는 성격이 다르다. 심사를 거치지 않은 원고가 올라오고 그중 상당수가 나중에 저널에 실린다. 프리프린트를 위키에 들일지는 정해 두어야 하는 결정이다. 들이기로 했다면 그것이 프리프린트라는 사실이 문서에 남아야 하고 나중에 출판본이 나왔을 때 어느 쪽이 정보인지 정하는 절차도 필요하다. 6장에서 본 중복 검사가 여기서 다시 걸린다.

프리프린트에는 한 가지 함정이 있다. 프리프린트의 식별자는 그 파일이 어디서 왔는지를 말하고 그 논문이 결국 어느 저널에 실렸는지를 말하지 않는다. 넣지 않기로 한 저널이 있다면 프리프린트 경로로 들어온 논문은 그 검사를 그대로 통과한다. 이것이 6장에서 다룬 두 겹 검사가 필요한 이유였다. 후보 목록을 만드는 단계에서 프리프린트에 출판본이 있는지 함께 조회해 두면 그 문제가 앞에서 걸려진다. 조회는 바깥 네트워크로 나가는 동작이므로 사람이 지시했을 때 한다.

관심 저널을 정하고 예약으로 돌리기

후보를 모으는 일은 매주 같은 모양으로 반복되므로 예약 작업에 맡기기 좋다. 준비할 것은 세 가지다. 어느 저널과 어느 주제를 볼 것인지, 어떤 기준으로 거를 것인지, 결과를 어디에 남길 것인지다. 첫 번째는 목록으로 적어 둔다. 자기 분야에서 실제로 읽게 되는 저널이 대체로 열 개에서 스무 개 사이이고 그 목록은 몇 달에 한 번 고치면 된다. 주제어로도 함께 잡는다. 저널만으로 잡으면 다른 저널에 실린 관련 논문을 놓치고 주제어만으로 잡으면 관련 없는 논문이 많이 들어온다. 두 축을 함께 쓰면 어느 한쪽에서 놓친 것이 다른 쪽에서 걸린다.

예약 작업이 하는 일은 지난 기간에 새로 올라온 논문을 조회해 목록으로 만들고 정해 둔 기준으로 표시를 붙여 한 문서에 쌓는 것이다. 여기서 원문을 받지 않는다는 점이 중요하다. 예약 작업이 파일을 받기 시작하면 사람이 보지 않는 사이에 폴더가 채워지고 무엇을 왜 받았는지 아무도 모르는 상태가 된다. 결과는 작업 메모에 남긴다. 후보 목록은 아직 지식이 아니고 다음에 할 일의 목록이다. 예약 작업을 만드는 절차와 그 작업이 살아 있는지 확인하는 방법은 27장에서 다룬다. 여기서 미리 짚을 것은 하나다. 매일 도는데 아무도 읽지 않는 목록은 끄는 편이 낫고 그 판단의 기준은 그 목록 때문에 실제로 바뀐 결정이 있었는가다.

기준을 정해 거르기

목록만 쌓이면 읽지 않는 목록이 하나 더 생길 뿐이다. 거르는 기준을 함께 정해야 한다. 기준은 자기 위키의 카테고리에서 나온다. 3장에서 카테고리마다 포함하는 것과 제외하는 것을 한 문장씩 적어 두었는데 그 문장이 여기서 그대로 쓰인다. 새 논문이 어느 카테고리에 해당하는지 판정하고 어느 카테고리에도 해당하지 않으면 표시해 둔다. 어느 카테고리에도 걸리지 않는 논문이 반복해서 나타난다면 그것은 자기 관심사가 옮겨 가고 있다는 신호이고 새 카테고리를 만들 후보다.

거르는 기준을 하나 더 두면 목록이 실제로 짧아진다. 지금 진행 중인 작업과의 관련성이다. 관심 분야에 해당한다는 것과 지금 읽어야 한다는 것은 다르다. 자기가 지금 쓰고 있는 원고나 준비 중인 분석과 직접 이어지는 논문을 앞에 놓고 나머지는 목록에 남긴다. 이 판정은 위키가 이미 가지고 있는 종합 문서와 대조하면 나온다. 새 논문의 초록을 읽고 그것이 어느 종합 문서의 주장을 강화하는지 좁히는지 반박하는지를 표시하게 하면, 반박이나 대체로 표시된 것이 우선 읽을 후보가 된다. 판정 자체는 초록만 보고 하는 것이므로 틀릴 수 있고 그래서 표시는 후보를 정렬하는 데만 쓰고 위키에 옮기지 않는다. 원문을 읽고 4장의 절차를 거친 뒤에야 그 판정이 위키에 들어간다.

인용 목록에서 거슬러 올라가기

후보는 새로 나온 논문과 이미 읽은 논문의 참고문헌에서 나온다. 참고문헌 쪽이 대체로 더 정확하다. 새 논문 목록은 그 주제에 무엇이 나왔는지를 알려 주지만 그것이 자기 주제와 관련 있는지는 초록을 읽어야 알 수 있다. 참고문헌은 이미

관련이 확인된 논문이 인용한 것이므로 관련성이 한 단계 걸려져 있다. 10장에서 다루는 구조 분석이 이 작업을 체계로 만든 것이고 그 절차의 한 단계가 인용된 논문 가운데 위키에 없는 것을 목록으로 만드는 일이다. 여기서 나온 목록이 다음에 확보할 논문의 후보가 된다.

거꾸로 가는 방향도 있다. 어떤 논문을 인용한 최근 논문을 찾으면 그 주제가 그 뒤로 어떻게 전개되었는지가 보인다. 몇 해 전 논문을 읽고 나서 그 논문이 지금 어떻게 받아들여지고 있는지 알고 싶을 때 쓰는 경로다. 두 방향 모두 서지 정보 목록에서 조회할 수 있다. 인용한 논문 수와 인용된 횟수가 서지 정보에 함께 들어 있고 그 관계를 따라가면 목록이 나온다. 다만 인용 횟수가 많다는 것을 중요하다는 뜻으로 읽지 않는다. 10장에서 다루는 대로 인용 수는 몇 번 불렸는지를 셀 뿐이고 어느 기능으로 불렸는지는 세지 않는다. 후보를 좁히는 데 쓰는 것과 읽을 가치를 판정하는 데 쓰는 것은 다른 일이다.

브라우저로 내려받기

받을 수 있다고 표시된 논문이라도 실제로 받는 데는 한 단계가 더 있다. 출판사 사이트는 자동 접근을 막아 두는 경우가 많고 기관 구독으로 열리는 자료는 그 기관의 인증을 통과한 상태여야 보인다. 그래서 이 단계는 이미 로그인해 둔 브라우저를 쓴다. 평소 논문을 보는 그 브라우저에서 페이지를 열고 내려받는 방식이면 인증 상태가 그대로 쓰인다. 에이전트가 브라우저를 조작할 수 있으면 이 과정을 맡길 수 있다. 논문 주소를 열고 본문 PDF 링크를 찾고 내려받고 정해 둔 이름으로 폴더에 넣는 일까지가 반복 작업이다.

여기서 정해 둘 것이 두 가지 있다. 하나는 내려받은 파일의 이름이다. 출판사가 붙이는 이름에는 논문 정보가 없으므로 받는 즉시 4장의 파일명 규칙으로 바꾼다. 나중에 바꾸려면 파일을 열어 무엇인지 확인해야 하고 스무 개가 쌓이면 그 확인이 일이 된다. 다른 하나는 받은 것이 실제로 본문인지 확인하는 일이다. 출판사 페이지를 그대로 저장한 파일에는 본문 대신 구독 안내와 로그인 화면만 들어 있는 경우가 있고 파일 크기와 쪽수만 보면 정상으로 보인다. 4장에서 원문의 완전성을 확인하라고 한 항목이 여기서 걸린다. 받은 직후에 쪽수와 첫 쪽의 제목을 확인하고 어긋나면 그 논문은 받지 못한 것으로 처리한다.

Paywall 논문의 처리

받을 수 없는 논문이 남는다. 구독하지 않은 저널의 논문, 오래된 호의 논문, 그리고 기관 구독 범위 밖의 논문이 여기 해당한다. 여기서 절차는 멈춘다. 우회하는 경로를 찾지 않고 정당한 경로로 넘긴다. 첫 번째 경로는 소속 기관의 도서관이다. 대학 도서관은 상당수의 저널을 구독하고 있고 교내망이나 도서관이 제공하는 접속 경로를 거치면 열리는 논문이 많다. 학교 밖에서 작업할 때 열리지 않던 논문이 도서관 경로로는 열리는 경우가 흔하므로 받지 못했다고 판정하기 전에 이 경로를 한 번 거친다. 두 번째 경로는 도서관의 원문 복사 신청이다. 구독하지 않은 저널의 논문도 다른 기관에서 받아다 주는 서비스가 대체로 마련되어 있고 며칠이 걸린다. 세 번째는 저자에게 직접 요청하는 것이다. 논문에 적힌 교신저자 주소로 요청하면 보내 주는 경우가 많다.

에이전트에게 이 단계를 맡기지 않는다. 도서관 인증은 개인 계정으로 이루어지고 원문 복사 신청과 저자 요청은 사람 이름으로 나가는 요청이다. 에이전트가 할 일은 여기까지 왔다는 것을 알리고 목록을 정리해 주는 데까지다. 받지 못한 논문은 미보유 목록에 남긴다. 이 목록에 적을 것은 그 논문의 식별자와 제목, 왜 필요한지, 그리고 어느 경로를 시도했고 어디서 막혔는지도. 마지막 항목이 없으면 몇 달 뒤에 같은 논문을 다시 시도하면서 한곳에서 다시 막힌다. 이 목록은 도서관에 신청을 낼 때 한꺼번에 쓰기도 좋다. 한 편씩 신청하는 것보다 열 편을 모아 신청하는 편이 절차가 적게 든다.

받지 못한 논문을 초록만으로 정리해 위키에 넣지 않는다. 4장에서 본 대로 원문이 완전한 정리를 뒷받침하지 못하면 그 논문은 지식 층에 넣지 않고 공백으로 기록한다. 초록만 읽고 만든 페이지는 검색 결과에서 제대로 만든 페이지와 구별되지 않고 몇 달 뒤에는 어느 것이 초록만 보고 쓴 것인지 기억나지 않는다. 없는 것으로 기록된 논문은 나중에 채울 수 있고 있는 것처럼 기록된 논문은 다시 열리지 않는다.

받지 못한 상태로 오래 남는 논문이 생기면 그 사실 자체가 판단의 재료가 된다. 같은 저널의 논문이 반복해서 막힌다면 그 저널의 구독 여부를 도서관에 문의할 근거가 되고 여러 사람이 같은 요청을 넣으면 구독 목록이 바뀌기도 한다. 특정 분야의 논문이 대부분 막힌다면 그 분야를 다루는 방식 자체를 다시 볼 필요가 있다. 반대로 공개 접근 논문만으로 충분히 굴러가는 주제도 있다. 최근 몇 년의 논문은 상당수가 공개로 나오므로 분야에 따라서는 막히는 논문이 몇 편 되지 않는다. 미보유 목록이 몇 편인지를 세어 보면 이 상황이 어느 쪽인가가 드러난다. 목록이 길면 확보 경로를 정비할 때이고 짧으면 지금 방식으로 충분하다.

후보를 모으는 일은 하루 한 번 도는 예약 작업이 맡는다. 이 작업은 프리프린트 서버 두 곳과 문헌 데이터베이스를 하루치만 훑고 카테고리마다 등록해 둔 검색어로 걸러 관련도 점수를 매긴 뒤 그날의 보고서를 만든다. 점수 하한을 두는 이유는 하한이 없으면 목록이 읽히지 않는 길이가 되기 때문이다. 지금은 2점을 하한으로 쓰고 그 아래는 보고서에 올리지 않는다. 여기에 공개 접근 여부를 함께 확인해서 내려받을 수 있는 것과 그렇지 않은 것을 나눠 표시한다. 하루치만 보는 구성에는 실패에 대한 대비가 붙어 있다. 같은 날 두 번 돌지 않도록 잠금을 걸고 여섯 시간이 지난 잠금은 죽은 것으로 보고 지우며 그날의 보고서가 이미 정상적으로 만들어져 있으면 다시 돌지 않는다. 보고서가 만들어졌는데 형식이 어긋나 있으면 성공으로 보고하지 않고 실패로 끝낸다.

한 주가 어느 정도 분량인지는 보고서를 세어 보면 나온다. 2026년 7월 30일부터 8월 5일까지 이레 동안 관련 논문으로 올라온 것은 80편이고 그중 공개 접근 PDF까지 확보되어 곧바로 들일 수 있는 것은 27편이었다. 하루로 보면 0편인 날과 19편인 날이 함께 있다. 이 편차는 프리프린트가 요일을 타기 때문에 생기고 그래서 하루 단위로 밀렸는지를 판단하면 매번 틀린다. 주 단위로 보면 80편 가운데 실제로 위키에 들어가는 것은 훨씬 적다. 목록에 오르는 것과 읽기로 정하는 것 사이에 사람의 판정이 한 번 들어가고 그 판정을 예약 작업에 넘기지 않는다. 예약 작업이 하는 일은 후보를 모아 한곳에 쌓는 데까지이고 무엇을 읽을지는 그날의 연구 상태가 정한다.

저널 목차 알람을 함께 쓰는 방법도 있다. 대부분의 저널이 새 호가 나올 때 목차를 보내 주는 서비스를 제공하고 이 메일은 그 저널이 실제로 무엇을 실었는지를 편집자의 눈으로 정렬해 보여 준다. 조회로 받아오는 목록과 성격이 다르다. 조회는 조건에 맞는 것을 빠짐없이 가져오고 목차는 그 저널이 무엇을 앞에 놓았는지까지 보여 준다. 두 경로를 함께 쓰면 조회에서 걸리지 않은 논문이 목차에서 눈에 띄는 일이 생긴다. 다만 목차 알람을 열 개 이상 구독하면 편지함이 그것으로 채워지므로 메일 연결을 걸어 두고 에이전트가 훑게 하는 편이 낫다. 이 경로는 22장에서 다루는 메일 연결 위에서 돌아간다.

확보 기록

어느 논문을 언제 어디서 받았는지를 남긴다. 4장에서 어느 도구로 추출했는지 적어 두라고 한 것과 같은 성격이고 필요해지는 상황도 비슷하다. 같은 논문의 판본이 여럿일 때 어느 것을 썼는지, 원고본으로 읽은 것인지 출판본으로 읽은 것인지, 그리고 그 파일에 붙은 이용 허락 조건이 무엇인지가 나중에 판단의 근거가 된다. 원고본으로 정리해 둔 논문의 출판본이 나오면 다시 확인할 대상이 되는데 무엇을 원고본으로 읽었는지 적혀 있지 않으면 그 대상을 고를 수 없다.

목록 자체도 상태가 있다. 후보로 올라온 것, 읽기로 정한 것, 원문을 확보한 것, 위키에 들인 것, 받지 못해 미보유로 남긴 것이 서로 다른 상태다. 다섯 상태를 한 문서에서 표시로 관리하면 지금 어디가 막혀 있는지 보인다. 후보는 계속 쌓이는데 확보 단계에서 멈춰 있다면 도서관 신청을 미루고 있는 것이고 확보는 되는데 위키에 들이는 단계에서 밀린다면 배치 처리를 시작할 때다. 이 문서를 지식 층에 두지 않는 이유도 같다. 후보 목록은 작업의 상태이고 지식이 아니며 상태는 자주 바뀐다. 위키에 들어가는 것은 그 논문을 실제로 읽고 만든 정리와 판단이다.

8장. Supplementary Table과 Document를 ingest 하는 방법

논문 한 편에서 실제로 다시 쓰이는 것은 본문 문장보다 표인 경우가 많다. 유전체 연구에서 메타분석을 하려면 여러 논문의 유전자 목록이 필요하고 그 목록은 그림 안에 요약된 몇 개를 빼면 전부 보충 자료의 표에 있다. 코호트의 인원 구성, 세포 유형별 표지 유전자, 차등 발현 결과, 실험 조건별 처리 시간도 마찬가지다. 앞의 장들이 다른 흐름은 원문 PDF에서 정리 문서로, 정리 문서에서 위키 페이지로 가는 한 방향이었다. 보충 자료는 나중에 계산의 입력으로 다시 열리므로 그 흐름에 얹히지 않는다.

지식 층과 자원 층의 구분

보충 자료는 별도의 폴더가 소유한다. 원문 PDF를 두는 자리에 함께 두지 않는 이유는 두 가지다. 첫째는 세기 위해서다. 논문 한 편에 보충 파일이 스무 개 붙는 경우가 드물지 않고 그것들이 같은 폴더에 있으면 논문이 몇 편인지 셀 수 없다. 3장에서 본 폴더 분리가 이 필요에서 갈라져 나왔다. 둘째는 성격이 다르기 때문이다. 원문에서 정리 문서로 이어지는 층은 그 논문이 무엇을 주장하는지를 나른다. 보충 자료 층은 다시 쓰이는 데이터를 나른다. 두 층에서 하는 일이 다르므로 찾는 방법도 달라야 한다. 주장을 찾을 때는 개념과 질문으로 찾고 데이터를 찾을 때는 어느 논문의 어느 표인지로 찾는다.

자원 층이 실제로 쓰이는 자리는 두 가지다. 하나는 본문의 어떤 수치를 확인하는 일이다. 그림 설명에 인용된 값이 보충 자료의 표에만 있는 경우가 흔하고 본문에서 그 값을 근거로 쓰려면 표를 열어야 한다. 다른 하나는 분석의 입력으로 넣는 일이다. 유전자 목록, 차등 발현 표, 코호트 표, 세포 유형 대응표는 계산에 들어간다. 두 용도 모두 원문을 읽어 요약하는

작업과 겹치지 않는다. 이 층의 파일은 원문 PDF를 대신하지 않는다. 출판사에 따라 논문 전체를 보충 파일로 한 번 더 배포하는 경우가 있는데 그 파일이 있어도 인용하고 읽는 것은 원문 자리의 PDF다. 이 책이 근거로 삼은 위키에는 보충 자료를 가진 논문이 376편 있고 각 편마다 폴더 하나가 있다.

표가 논문의 본체인 경우

일부 논문에서는 보충 자료의 표가 논문의 기여 그 자체다. 대형 저널의 Resource 형식 논문과 그에 준하는 형식이 여기 해당한다. 이런 논문에서 본문은 표가 무엇이고 어떻게 만들었는지를 설명하는 글이고 다른 연구자가 실제로 가져다 쓰는 것은 표다. 이 논문들을 본문만 읽고 정리하면 정리 문서는 만들어지지만 그 논문에서 얻을 것은 얻지 못한 상태로 남는다. 그림에서 유전자 목록을 찾으려다 실패하는 일도 여기서 생긴다. 그림에는 대표적인 몇 개만 이름이 붙어 있고 전체 목록은 표에 있기 때문이다. 그래서 이런 논문은 처음부터 표를 일급으로 다룬다. 원문을 읽고 정리 문서를 쓰는 작업과 별개로, 어떤 표가 있고 각 표의 열이 무엇인지를 확인해 기록해 둔다. 그 기록이 없으면 반년 뒤에 유전자 목록이 필요할 때 어느 논문의 어느 파일에 있었는지를 다시 찾아야 한다.

보충 자료를 자동으로 모으는 방법

보충 자료를 내려받는 일은 손으로 하면 지치는 작업이다. 논문마다 출판사가 다르고 보충 자료가 있는 위치도 다르고 여러 파일이 각각 다른 링크로 걸려 있다. 논문 마흔 편에서 표를 모으려면 마흔 번 페이지를 열고 링크를 눌러야 한다. 이 작업은 판단이 거의 들어가지 않고 반복만 많으므로 에이전트에게 맡기기 좋다. 화면을 직접 조작할 수 있는 도구를 쓰면 논문 페이지를 열고 보충 자료 영역을 찾아 파일을 내려받는 일까지 자동으로 진행된다. 여기서 사람이 정해 주어야 하는 것은 대상 논문 목록과 내려받은 파일을 어디에 어떤 이름으로 둘지다. 자동으로 모으는 작업에서 가장 자주 어긋나는 것이 파일 이름이다. 출판사가 붙이는 이름에는 논문 정보가 들어 있지 않으므로 내려받은 파일을 논문별 폴더에 넣지 않으면 어느 파일이 어느 논문의 것인지 며칠이면 알 수 없게 된다.

따라해보기

아래 블록은 AI 도구의 채팅창에 그대로 붙여 넣는 프롬프트다. 보충 자료를 모을 때 세션에 주는 요청은 다음 정도면 된다.

```
< DOI >
papers-supplementary/{ - }/
.
-
-
-
```

마지막 두 줄이 중요하다. 보충 자료가 없는 논문과 있는데 받지 못한 논문은 다음에 할 일이 다르다. 앞의 것은 다시 시도할 이유가 없고 뒤의 것은 다른 경로로 구해야 한다. 둘을 구분하지 않으면 두 경우 모두 없는 것으로 기록되고 나중에 그 논문의 표가 필요해졌을 때 애초에 없었던 것으로 판단하게 된다. 받지 못했다는 사실 자체를 남기는 방식도 정해 두었다. 폴더는 만들고 기록만 남긴 채 비워 둔다. 폴더가 있는데 파일이 없으면 그 논문에 보충 자료가 있다는 것을 알면서 아직 확보하지 못했다는 뜻이고 폴더가 없으면 확인하지 않았다는 뜻이다. 두 상태를 구분해 두어야 빈자리가 눈에 보인다.

파일 이름이 알려 주지 않는 것과 manifest

출판사가 붙인 보충 파일 이름은 대체로 내용을 알려 주지 않는다. 압축 파일 하나에 스물여덟 개의 파일이 들어 있고 각각이 서로 다른 표인 경우도 있다. 파일 목록만 보고는 어느 것이 유전자 목록이고 어느 것이 코호트 표인지 알 수 없다.

그래서 논문마다 목록 문서를 하나씩 둔다. 이 문서에는 논문의 제목과 저널과 식별자, 보충 자료가 놓인 폴더, 그 논문의 위키 페이지, 파일 수, 그리고 이 자료가 무엇인지를 한두 문장으로 적는다. 그 아래에 다시 쓸 만한 파일이 무엇이고 그 안에 무엇이 들었는지를 적고 데이터가 아닌 것이 섞여 있으면 그것도 적는다. 압축 파일에 편집 프로그램이 남긴 잠금 파일 같은 부스러기가 함께 들어 있는 경우가 실제로 있고 그것을 데이터로 착각해 열어 보는 시간이 반복된다.

목록 문서를 논문마다 따로 두는 이유는 검색 때문이다. 한 파일에 모두 적으면 그 파일이 수백 줄이 되고 유전자 목록이 있는 논문을 찾을 때 그 안에서 다시 훑어야 한다. 논문별로 나누어 두면 위키의 검색이 그 문서를 하나의 결과로 돌려주고 결과에서 곧바로 보충 자료 폴더와 논문 페이지로 갈 수 있다. 이 문서를 어디에 두는지도 정해져 있다. 지식 층 안에 두지 않고 자원 층 옆에 둔다. 지식 층은 논문이 무엇을 주장하는지를 담는 자리이고 어느 파일에 무엇이 들었는지는 그 논문의 주장이 아니기 때문이다. 목록 문서를 손으로 썼는지 도구가 만들었는지도 함께 적는다. 손으로 쓴 것은 압축을 풀어 안을 본 사람이 있다는 뜻이고 자동으로 만든 것은 파일 이름과 크기만 본 것일 수 있다.

Reporting Summary에서 살릴 것과 버릴 것

일부 저널은 논문마다 정해진 서식의 보고 요약 문서를 함께 낸다. 앞쪽은 통계 항목을 체크하는 빈 서식이라 아무 정보가 없다. 뒤쪽은 저자가 직접 채운 부분이고 여기에는 논문 어디에도 없는 것들이 들어 있다. 표본 수를 정한 근거가 된 검정력 분석, 특정 개체를 제외한 이유, 무엇을 무작위화하고 무엇을 가렸는지, 사용한 소프트웨어의 판, 코드 저장소 주소, 참여자의 연령 범위와 성별 구성이 그렇다. 방법을 따라 하거나 설계를 비교할 때 필요한 정보가 대체로 여기에 있다. 그래서 규칙이 두 갈래로 되어 있다. 문서 자체는 자원 층에 보존하고 앞쪽의 빈 서식은 정리 문서와 위키 페이지에 들어가지 못하게 막는다. 채워진 부분의 사실은 출처를 밝히고 인용해도 된다.

이 구분은 두 번의 실수를 겪고 나서 정해졌다. 논문 PDF 끝에 붙은 보고 요약이 통째로 추출되어 정리 문서 두 편에 들어갔고 체크 항목의 상투구가 본문처럼 남았다. 2026년 7월 28일에 빈 서식 부분을 걷어내고 채워진 답만 남겼다. 반대 방향의 실수도 있었다. 그보다 이틀 앞선 정리 작업에서 행정 문서를 걷어내는 규칙을 넓게 적용해 보고 요약 27건을 포함한 45개 파일을 지웠다. 지운 파일은 되살릴 수 없지만 같은 문서가 논문 PDF 끝에도 붙어 있는 경우가 많아 필요해지면 원문에서 그 쪽만 다시 떼어낼 수 있다. 2026년 7월 28일에 최근 논문 40편을 표본으로 확인했더니 참고문헌 뒤에 이 문서가 붙어 있는 것이 14편이었다. 추출할 때는 그 서식이 시작되기 전에서 쪽 범위를 끊어 상투구가 정리 문서에 달지 않게 한다.

보관하지 않는 문서

보충 자료로 함께 내려받아지지만 남기지 않는 문서가 있다. 심사 과정에서 오간 문서와 결정 통지서, 그리고 저자가 아무것도 채우지 않은 빈 점검표가 여기 해당한다. 앞의 것들은 논문의 연구 내용을 담지 않고 심사자와 편집자의 판단을 담는다. 뒤의 것은 채워지지 않았으므로 담긴 정보가 없다. 판정이 애매한 경우도 있다. 실제 내용이 있는 보충 문서가 자기 파일 목록 안에서 그 점검표의 이름을 언급하는 경우인데, 이것은 진짜 보충 자료이므로 남긴다. 판정할 때는 저자가 쓴 내용이 그 안에 있는지를 본다. 앞 절에서 본 지나친 정리가 이 기준을 이름으로 잡았을 때 일어났다. 이름에 특정 단어가 들어가면 지우는 방식은 빠르지만 그 단어가 들어간 문서 가운데 저자가 채운 것이 섞여 있다.

압축 파일 안에 들어 있는 것

보충 자료는 압축 파일 하나로 오는 경우가 많고 그 안에는 데이터가 아닌 것이 함께 들어 있다. 편집 프로그램이 파일을 열어 두었을 때 만드는 잠금 파일, 운영체제가 폴더마다 남기는 설정 파일, 압축을 만든 사람의 컴퓨터에서 달려 온 부스러기가 그렇다. 이름이 특이해서 열어 보게 되고 열어 보면 아무것도 없다. 압축을 풀 때 이런 것들을 폴더에 그대로 쏟아 놓지 않는다. 파일 수가 늘어나면 목록 문서에 적어야 할 것이 늘고 그 문서를 읽는 사람이 다시 걸러 내야 한다. 목록 문서에 이런 부스러기가 있다는 사실을 한 줄로 적어 두면 다음 사람이 같은 파일을 다시 열어 보지 않는다.

압축 안의 파일이 표인 경우에는 열 이름까지 적어 둔다. 파일 이름이 표1, 표2로만 되어 있으면 어느 것이 유전자 목록이고 어느 것이 코호트 표인지 알 수 없고 반년 뒤에 필요한 것을 찾으려면 다시 전부 열어야 한다. 열 이름 목록은 짧고 그 표가 무엇인지 거의 다 알려 준다. 유전자 기호와 로그 배수 변화와 조정된 유의확률이 열로 있으면 그것은 차등 발현 결과이고 시료 식별자와 나이와 성별과 진단이 있으면 코호트 표다. 이 목록을 적어 두는 데 드는 시간은 논문마다 몇 분이고 적지 않으면 그 몇 분이 그 표가 필요해질 때마다 되돌아온다. 여러 논문에서 같은 종류의 표를 모을 때도 이 목록이 먼저 쓰인다. 열 이름을 비교하면 그 표들을 하나로 합칠 수 있는지가 합치기 전에 판정된다.

남의 파일과 내가 만든 파일

같은 폴더 안에 출판사가 준 파일과 직접 만든 파일이 섞이면 나중에 구분되지 않는다. 그림의 한 부분을 잘라 이미지로 만든 것, 보충 자료의 표를 가공해 만든 결과, 다른 곳에서 받아 온 파일이 여기 해당한다. 그래서 최상위에는 출판사가 준 파일만 두고 직접 만든 것은 이름을 붙인 하위 폴더로 내린다. 논문 PDF에서 떠난 화면과 잘라낸 그림은 한 자리에, 계산해서 만든 결과는 다른 자리에 둔다. 출판사가 준 것이 아닌데 하위 폴더에 넣기 애매한 파일이 남으면 그것이 어디서 왔는지 적은 문서를 같이 둔다. 이 구분이 필요한 이유는 인용할 때 달라지기 때문이다. 가공해서 만든 결과는 그 논문의 저자가 만든 것이 아니므로 입력은 논문을 인용하고 가공한 결과는 자기 작업을 밝힌다. 목록 문서에도 어느 파일이 직접 만든 것인지 적는다. 잘라낸 그림만 들어 있고 출판사 데이터는 하나도 없는 폴더가 실제로 있고 그 사실을 적어 두지 않으면 없는 표를 찾는 사람이 생긴다.

보충 자료가 있는지 미리 아는 법

논문마다 보충 자료를 확인하러 가는 일은 그 자체로 시간이 든다. 어떤 논문에는 표가 스무 개 붙어 있고 어떤 논문에는 아무것도 없으며 열어 보기 전에는 알 수 없다. 실제로는 단서가 있다. 본문에서 보충 표를 가리키는 표현이 몇 번 나오는지 세면 그 논문이 표에 무엇을 얼마나 실었는지가 대체로 드러난다. 결과 절마다 보충 표를 가리키는 논문은 표가 본체에 가깝고 방법 절에서 한 번만 가리키는 논문은 시료 목록 정도가 붙어 있다. 이 세는 일은 원문을 추출한 뒤에 자동으로 할 수 있으므로 논문을 넣을 때 함께 기록해 두면 나중에 표가 필요할 때 후보를 좁힐 수 있다.

확인하지 못한 것을 없는 것으로 적지 않는 규칙이 여기서도 적용된다. 보충 자료가 아예 없는 논문과 있는데 확보하지 못한 논문은 다음에 할 일이 다르다. 앞의 것은 다시 시도할 이유가 없고 뒤의 것은 다른 경로로 구해야 한다. 그래서 후자는 폴더와 목록 문서를 만들고 비워 둔 채로 남긴다. 폴더가 비어 있는데 목록 문서가 있으면 그 논문에 보충 자료가 있다는 것을 알면서 아직 구하지 못했다는 뜻이고 어떻게 해서 비게 되었는지도 그 문서에 적어 둔다. 논문에 보충 자료가 아예 없다는 것이 확인되었을 때만 폴더와 목록 문서를 함께 지운다. 이 규칙이 생기기 전에 빈 폴더 네 개가 지워진 적이 있고 그중 하나에 실제로 보충 자료가 있었다면 그 사실을 다시 발견할 방법이 없다.

표를 분석 입력으로 쓰기

큰 표를 PDF에서 뽑으려는 시도는 대체로 실패한다. 5장에서 본 대로 두 추출기 모두 표의 칸 구조를 제대로 복원하지 못하고 한쪽은 칸이 빈 표 모양의 격자까지 만들어 낸다. 실제로 쓸 수 있는 방법은 보충 자료의 표 파일을 직접 읽는 것이다. 표 계산 프로그램 형식으로 배포된 파일은 열과 행이 그대로 살아 있으므로 그 파일을 프로그램으로 열어 필요한 열만 꺼내면 된다. 이 차이는 크다. PDF의 표를 재구성하려고 애쓰는 시간과, 같은 내용을 담은 원본 파일을 여는 시간은 비교가 되지 않는다. 그래서 표가 필요한 논문에서는 보충 자료를 먼저 확인하고 표 파일이 있으면 PDF의 그 표는 읽지 않는다.

표에서는 목록과 실험 조건을 함께 꺼낸다. 처리 농도와 시간, 배양 조건, 사용한 항체와 그 희석 비율, 시퀀싱 깊이, 사용한 소프트웨어의 판과 매개변수가 다음 분석의 변수가 된다. 여러 논문에서 같은 항목을 꺼내 나란히 놓으면 조건이 얼마나 다른지가 보이고 그 차이가 결과의 차이를 설명하는 경우가 있다. 두 논문의 결과가 어긋날 때 그것이 진짜 모순인지 조건의 차이인지를 판단하는 일이 1장에서 말한 어려운 일이었는데 그 판단의 재료가 대체로 여기에 있다. 자기 실험을 설계할 때도 같은 표를 쓴다. 비슷한 실험을 한 논문 다섯 편의 조건을 한 표로 모으면 어느 값이 그 분야의 관행이고 어느 값이 그 논문의 선택인지가 구분된다. 꺼낸 값은 종합 문서에 모은다. 논문 한 편의 조건은 그 논문의 정보이고 다섯 편을 나란히 놓은 표는 그 자체가 하나의 판단이기 때문이다.

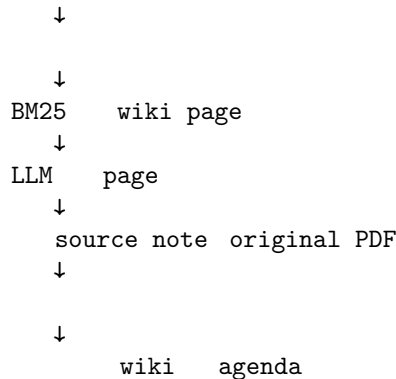
자동 수집이 막히는 곳은 대체로 파일 이름이다. 출판사가 붙인 이름에는 논문 정보가 없고 어느 논문의 것인지는 그 이름 안에 박힌 식별자나 저널 코드로만 맞출 수 있다. 그 코드가 없는 형식이 섞여 들어오면 대응은 사람의 판단으로 넘어간다. 형식 자체가 판정을 어렵게 만드는 경우도 있다. 논문과 보충 자료를 하나로 묶어 배포하는 파일은 원문 PDF의 사본처럼 보여서 중복으로 지워질 후보가 되는데 2026년 7월 28일에 네 건을 실행해 보니 세 건이 원문보다 쪽수가 많았다. 한 건은 81쪽 대 44쪽이었고 45쪽부터가 보충이었으므로 그 파일을 지웠으면 보충 자료를 잃는 것이었다. 정리 규칙을 이름으로 잡았을 때 실제로 사고가 났다. 2026년 7월 26일에 넓은 규칙 하나로 45개 파일이 지워졌고 그중 27건이 보고 요약이었다. 지운 것은 되살릴 수 없었지만 같은 문서가 논문 PDF 뒤에 붙어 배포된 경우가 많아서 다시 떼어낼 수 있었고 그 뒤로는 파일 안에 저자가 쓴 내용이 있는지를 보고 판정한다.

9장. 질문이 지식으로 축적되는 법

검색이 하는 일과 하지 않는 일을 나누고 후보 문서에서 실제 근거로 내려가는 순서를 정하고 여러 논문을 비교해 얻은 판단을 다시 위키로 되돌리는 지점까지 간다. 4장이 자료를 들여오는 방향이었다면 여기서는 들여온 자료를 꺼내 쓰는 방향으로 같은 순환이 돈다. 대학원 수업에서 단일세포 데이터를 다루는 주에 학생이 이런 질문을 했다고 하자. 요즘 나온 세포 foundation model들이 실제로 쓸 만한가요. 질문은 한 문장이지만 답하려면 최소 다섯 편의 논문을 나란히 놓아야 하고 각 논문이 무엇을 근거로 자기가 쓸 만하다고 말하는지를 비교해야 한다. 논문 다섯 편이 이미 위키에 들어 있다면 출발점은 나쁘지 않다. 그런데 검색창에 세 단어를 넣으면 스무 개의 결과가 나오고 그 목록은 무엇을 먼저 읽어야 하는지도, 그중 어느 것이 질문한 관계를 실제로 다루는지도 알려 주지 않는다. 목록에서 답까지 가는 거리를 좁히는 일이 실제 작업이다.

질문을 받으면 LLM-Wiki 안에서 일어나는 일

질문 하나가 위키를 통과하는 경로에는 여섯 개의 지점이 있고 각 지점에서 사람이 판단해야 할 것이 다르다.



경로에서 가장 오해받는 자리가 세 번째다. 검색이 하는 일은 만 건이 넘는 문서 중에서 읽어 볼 만한 스무 건을 골라 주는 것이고 그다음부터는 문서를 실제로 읽어야 한다. 순위표의 1위가 답이라고 생각하면 검색을 잘못 쓰고 있는 것이다. 후보를 좁히는 일과 근거를 만드는 일은 서로 다른 작업이고 앞의 일은 기계가 잘하고 뒤의 일은 아직 사람의 확인이 필요하다. 질문마다 여섯 지점을 내려가는 깊이도 다르다. 어떤 방법을 쓴 논문이 위키에 있는지 확인하는 질문은 세 번째 지점에서 사실상 끝나고 그 논문이 무엇을 주장했는지 묻는 질문은 네 번째에서 끝난다. 결과의 크기가 실제로 다르지, 두 논문의 결론이 왜 어긋나는지를 묻는 질문만 다섯 번째까지 내려가고 마지막 지점까지 가는 것은 그중에서도 다시 쓸 판단이 나온 경우뿐이다. 질문을 받았을 때 어디까지 내려갈 작업인지를 먼저 가늠하면 시간 배분이 달라진다. 사실 확인 하나에 원문 세 편을 여는 일과 비교 질문을 검색 결과 목록으로 끝내는 일은 정반대 방향의 같은 실수다.

출발점에 들어오는 말은 대개 불완전하고 그대로 검색에 넣으면 엉뚱한 후보가 올라온다. 도착점에서 답이 만들어지면 대화가 끝난 것처럼 보이지만 저장하지 않으면 그 답을 만드는 데 읽은 다섯 문서를 다음번에도 똑같이 읽어야 한다. 4장의 여섯 단계가 자료를 들이는 쪽에서 위키를 바꿨다면 질문 쪽에서도 위키가 바뀌어야 축적이 일어난다. 여섯 지점을 지나는 규칙은 습관으로 남겨 두지 않고 저장소의 지침 문서에 문장으로 적어 두었다. 답은 위키와 source note에 있는 논문만을 근거로 만든다. 웹 검색은 사용자가 명시적으로 요청할 때만 쓴다. 정리가 불충분하면 원본 PDF를 다시 읽어 보충한다. 해당 논문이 아예 없으면 없다고 말하고 PDF를 요청한다. 검색 정책 절에는 문장이 하나 더 붙어 있는데 검색 점수는 후보를 가리는 데까지만 쓰고 질문한 관계가 뒷받침되는지는 페이지를 읽어서 판정한다는 규정이다. 규정으로 적어야 했다는 것은 적어 두지 않으면 그렇게 하지 않게 된다는 뜻이다. 묶어 두지 않으면 답의 절반이 위키에서 오고 절반이 모델의 사전 지식에서 오게 되고 나중에 그 답을 되짚을 때 어느 쪽이 어디에서 왔는지 구분할 수 없다. 위키 안에서만 답하도록 묶어 두면 답의 품질이 위키의 상태를 그대로 반영하고 답이 부실하다는 것은 위키에 무엇이 빠졌다는 신호가 된다.

LLM-Wiki에 질문했을 때 위키 파일을 검색하는 방법

검색이 다루는 대상은 폴더 안의 Markdown 문서 전체다. 논문 페이지와 종합 문서와 원문을 읽은 기록이 모두 여기 들어가고 문서가 만 건을 넘는 위키에서 질문과 관련된 스무 건을 골라내는 일이 검색이 맡는 몫이다. 고르는 경로는 두

갈래다. 하나는 목차 문서와 카테고리별 색인을 따라 내려가는 경로이고 다른 하나는 질문에 들어 있는 단어로 문서를 뒤지는 경로다. 앞의 것은 어느 카테고리에 있을지 알 때 빠르고 뒤의 것은 카테고리를 가로지르는 질문에서 쓸모가 있다. 두 경로가 모두 필요한 이유는 위키가 커지면서 카테고리 하나에도 문서가 수백 건씩 쌓이기 때문이다. 단어로 뒤지는 쪽에는 어휘 검색이라고 부르는 계열의 방법을 쓰고 이 책이 근거로 삼은 위키는 그중 BM25를 쓴다. 검색기의 이름보다 중요한 것은 그 방법이 무엇을 근거로 순위를 매기는지이고 그것을 알면 결과가 이상할 때 무엇을 고쳐야 할지가 정해진다.

BM25는 질문에 등장하는 단어가 각 문서에서 얼마나 구별력 있게 나타나는지를 계산해 문서를 순위화하는 방식이다. 어떤 단어가 어디에 몇 번 나오는지 세는 방식이므로 어휘 검색으로 분류된다. 계산 자체는 세 가지 직관으로 설명되고 세 직관을 알면 검색 결과가 왜 그렇게 나왔는지 대부분 설명할 수 있다. 첫째는 희귀도다. 위키 전체에 흔한 단어는 어느 문서를 고를지 알려 주지 못한다. 단일세포 연구 위키에서 cell이라는 단어는 거의 모든 문서에 들어 있으므로 후보를 하나도 좁히지 못한다. 반대로 chromatin accessibility처럼 소수의 문서에만 등장하는 표현은 그 단어 하나로 후보를 수십 건까지 줄인다. 흔한 단어만으로 이루어진 질문은 검색기를 무력하게 만든다.

둘째는 빈도 포화다. 어떤 문서에 질문의 단어가 스무 번 나와도 두 번 나온 문서보다 열 배 관련 있지는 않다. 처음 몇 번의 등장이 큰 정보를 주고 그 뒤로는 점수가 완만하게만 올라간다. 점수가 포화하지 않으면 같은 표현을 반복하는 긴 문서가 순위를 독차지하게 된다. 실제 위키에서 그런 문서는 대체로 원문 추출이 그대로 들어간 파일이다. 셋째는 문서 길이 보정이다. 긴 문서는 우연히 어떤 단어를 포함할 확률이 높으므로 보정하지 않으면 길이만으로 유리해진다. 논문 한 편을 통째로 담은 문서와 개념 하나를 반쪽에 정리한 문서가 같은 단어를 한 번씩 담고 있다면 짧은 쪽이 그 단어를 더 중심적으로 다루고 있을 가능성이 높다. 길이 보정은 짧고 밀도 높은 문서가 밀려나지 않게 한다. 4장에서 논문 wiki page를 source note와 분리하라고 한 이유 하나가 여기에 있다. 두 층이 한 파일에 섞여 있으면 문서가 길어지고 그 길이가 검색에서 손해로 돌아온다.

세 직관을 합치면 검색에 잘 걸리는 문서의 조건이 나온다. 나중에 그 문서를 찾을 때 쓸 법한 표현이 제목, 표제, 카테고리 이름, 본문에 실제로 들어 있어야 한다. 사람이 읽기에 아무리 좋은 문서라도 질문에 쓰일 단어가 없으면 후보에 오르지 않는다. BM25 점수는 질문과 문서의 어휘적 관련성이다. 점수가 높다는 것은 그 문서가 질문의 단어들을 잘 담고 있다는 뜻이고 표본 크기와 설계의 견고함과 결론의 재현 여부는 점수에 들어가지 않는다. 표본 여섯 개짜리 예비 연구가 표본 50만 명짜리 연구보다 상위에 올라오는 일은 흔하고 리뷰 논문은 어휘가 풍부해서 거의 모든 질문에 상위로 올라온다. 정정 공고나 출판사 상용구가 걸려 올라오는 일도 있다. 순위표를 근거로 서열로 읽는 순간, 검색기가 하지 않은 판단을 검색기가 한 것으로 착각하게 된다.

단어를 세는 방식이 낱아 보일 수 있는데도 자기 위키의 첫 검색기로 쓸 만한 이유가 셋 있다. 결과가 결정적이라는 것이 첫째다. 같은 질문과 같은 문서 집합이면 언제나 같은 순위가 나오므로 어제와 오늘의 답이 다르면 위키가 바뀐 것이다. 왜 이 문서가 올라왔는지 설명할 수 있다는 것이 둘째다. 질문의 어느 단어가 어느 문서에서 몇 번 나왔기 때문이라고 되짚을 수 있으므로 이상한 결과가 나왔을 때 원인을 찾을 수 있다. 셋째는 비용이다. 문서에서 인덱스를 다시 만드는 일이 가볍기 때문에 예약된 시각에 통째로 새로 만들어도 부담이 없고 인덱스를 소중히 다룰 이유가 없어진다. BM25가 못 하는 일도 분명하다. 단어의 의미를 모르므로 동의어를 잊지 못하고 표현이 다르면 같은 개념을 다른 문서를 놓친다. 그래서 이 방식만으로 위키를 다 뒤졌다고 말할 수 없고 근거로 삼은 위키는 목차 문서와 카테고리별 색인 문서가 맞는 결정적 탐색 경로를 따로 두었다.

완전한 질문으로 검색하는 이유

검색창에 세 단어를 던지는 습관은 웹 검색에서 왔다. 웹에서는 대상이 넓고 문서가 무한하므로 좁은 검색어가 유리하지만 자기 위키에서는 반대다. Geneformer, scGPT, benchmark라는 세 단어를 넣으면 세 단어가 함께 등장하는 문서가 올라오는데 그 문서들이 무엇을 비교했고 어느 방향의 결론을 냈는지는 목록에 나타나지 않는다. 세 단어를 던질 때 사라지는 것은 네 가지다. 무엇을 대상으로 하는지, 무엇과 무엇을 비교하는지, 어느 방향의 관계를 묻는지, 그리고 그 관계가 어떤 조건에서 성립하는지가 검색어에서 지워진다. 같은 궁금증을 완전한 질문으로 바꾸면 이렇게 된다. 같은 단일세포 데이터에서 Geneformer와 scGPT를 비교할 때, 두 논문은 각각 어떤 과제와 어떤 지표로 성능을 주장했고 두 주장이 같은 축 위에 있는가. 문장이 길어진 만큼 검색 토큰도 늘어나서 후보의 폭이 넓어지는 이익이 있지만 더 큰 이익은 읽기 기준이 생긴다는 데 있다. 후보 문서를 열었을 때 무엇을 확인해야 하는지가 질문 안에 이미 들어 있으므로 관련 있어 보이는 문단에서 멈추지 않고 과제와 지표와 축을 찾게 된다. 완전한 질문이 없으면 어떤 문서를 읽어도 관련 있어 보이고 처음 읽은 문서의 논조가 그대로 답이 된다. 질문을 완전하게 만드는 절차는 네 자리를 채우는 일로 정리된다. 대상을 정하고 비교 상대를 정하고 관계의 방향을 정하고 성립 조건을 정한다. 네 자리 중 채울 수 없는 것이 있다면 그것 자체가 정보다. 비교 상대를 정할 수 없다면 아직 무엇과 견주고 싶은지 모른다는 뜻이고 그 상태에서 검색하면 무엇이 나와도 만족스럽지 않다. 질문을 다듬는 데 쓰는 2분이 문서 다섯 개를 잘못 읽는 20분을 막는다. 완전한 질문은 논문을 다루는 위키 바깥에서도 같은 일을 한다. 지난 학기 강의 자료가 쌓인 폴더에 발표 준비라고만 검색하면 모든 파일이

걸리고 어떤 설명 순서를 썼는지 묻고 싶었다는 원래 관심은 목록 어디에도 나타나지 않는다. 같은 폴더에 대해 학부 3학년 수업에서 통계적 검정력을 설명할 때 어떤 예시를 먼저 놓았고 그 순서를 왜 바꿨는지를 묻는다면 강의안과 수업 후 메모가 함께 후보로 올라오고 무엇을 확인해야 하는지도 정해진다. 대상과 조건과 관계를 문장에 남기는 습관은 자료의 종류와 무관하게 작동한다. 위키의 크기가 작을 때는 차이가 작지만 문서가 몇백 건을 넘어가면 검색어의 완전성이 곧 검색의 성패가 된다.

한국어로 작업하고 영어로 정리된 위키를 쓰는 경우에는 한 단계가 더 붙는다. 근거로 삼은 위키는 내용의 언어를 영어로 두고 대화는 한국어나 영어로 한다. 한국어 질문을 그대로 넣으면 매칭할 토큰이 거의 없어 검색이 사실상 작동하지 않는다. 그래서 검색 정책의 기본 읽기 경로 자체가 완전한 영어 질문을 검색 도구에 넘기는 것으로 규정되어 있고 한국어 질문이 들어오면 원문을 그대로 둔 채 완전한 영어 재작성을 별도 인자로 함께 넘기게 되어 있다. 규정에는 문장이 하나 더 붙어 있다. 세 단어로 줄인 것은 재작성으로 인정하지 않는다. 그런 문장이 운영 지침에 들어갔다는 것은, 완전한 문장으로 다시 쓰라고 해 놓아도 명사 세 개가 돌아오는 일이 반복되었다는 뜻이다. 줄여도 결과가 스무 건은 나오기 때문에 잘못되었다는 신호가 즉시 오지 않는다. 원문을 남기는 이유는 번역이 질문을 조용히 바꾸기 때문이다. 성능이라는 한국어 단어를 accuracy로 옮기는 순간 학습 비용과 추론 속도에 대한 관심이 질문에서 빠지는데 원문이 남아 있으면 답을 받은 뒤에 그 누락을 알아챌 수 있다.

1. 궁금한 것을 평소처럼 한 줄로 적는다
2. 대상, 비교 상대, 관계의 방향, 성립 조건 네 자리를 채워 완전한 질문으로 고쳐 쓴다
3. 한국어로 적었다면 원문을 지우지 않고 완전한 영어 질문을 아래에 함께 만든다
4. 명사 세 개를 나열한 것은 영어 질문으로 치지 않는다

위키에 질문을 던지고 답의 근거를 확인하는 화면

완전한 질문으로 검색한 뒤 후보 문서를 실제로 읽고 답의 근거를 구분한다. 검색 결과의 순위는 읽을 순서를 정하며 근거의 강도는 사람이 판정한다.

검색 후보에서 근거로

BM25 상위 결과를 그대로 답으로 쓰지 않는다는 원칙은 다섯 단계의 절차로 구체화된다. 단계를 건너뛰면 답은 여전히 그럴듯하게 나오지만 근거를 되짚을 수 없는 답이 된다.

1. 후보로 올라온 논문 page, overview, concept page를 읽는다. 검색 결과에 딸려 나오는 발췌 몇 줄로 대신하지 않고 문서를 연다.
2. 질문한 관계가 실제로 본문에 명시되어 있는지 확인한다. 서로 다른 두 문서에 각각 있는 두 사실을 이어 붙인 것은 명시로 치지 않는다.
3. 수치나 경계 조건이 답을 좌우하면 source note를 읽는다. 정리 문서는 방향은 담지만 크기와 조건을 다 담지 않는다.
4. Source note가 부족하거나 그림과 표가 결론을 만드는 대목이면 원문 PDF를 다시 읽는다.
5. 위키에 근거가 없으면 없다고 답한다. 외부 지식이나 모델의 사전 지식으로 빈틈을 메우지 않는다.

두 번째 단계에서 가장 많은 오류가 걸러진다. 어떤 논문이 A를 보였고 다른 논문이 B를 보였을 때, A가 B의 원인이라는 문장은 어느 논문에도 없을 수 있다. 두 사실 사이의 연결을 만든 것이 논문인지 지금 답을 만드는 쪽인지 구분하지 않으면, 아무도 주장한 적 없는 문장이 근거 있는 문장처럼 답에 들어간다. 그래서 검색 정책은 관계가 명시되어 있지 않으면 판단을 보류하라고 규정한다. 답을 쓸 때 문장마다 어느 문서의 어느 대목에서 왔는지 붙일 수 있는지를 확인하면 이 오류는 대부분 드러난다. 붙일 수 없는 문장은 추정이므로 추정이라고 표시한다. 검색으로 올라온 후보 목록과 답에 실제로 근거로 쓴 문서 목록도 따로 적는다. 두 목록은 거의 언제나 다르고 상위 열 건 중 셋만 관련이 있고 그중 하나는 아래층까지 내려가야 했다면 그 기록이 다음번에 같은 주제를 다룰 때 어디서부터 읽어야 하는지를 알려 준다. 두 목록이 완전히 같다면 후보를 읽지 않고 순위를 그대로 옮긴 것이므로 기록을 남기는 일 자체가 절차를 지켰는지 검사하는 장치가 된다. 스크립트로는 검사할 수 없는 종류의 규칙이어서 그렇다. 위키의 링크 규약에서도 같은 일을 겪었는데 검사 스크립트가 두 형태를 모두 통과시키는 바람에 잘못된 형태가 다시 들어와도 아무 경고가 뜨지 않았다.

1. 완전한 질문으로 검색해 후보 목록을 받고 상위 열 건의 제목을 그대로 옮겨 적는다
2. 후보를 실제로 읽고 답에 근거로 쓴 문서만 따로 표시한다
3. 어느 문서에서 source note까지 내려갔고 어느 문서에서 원문까지 내려갔는지 적는다
4. 답을 쓰고 문장마다 어느 문서에서 왔는지 붙일 수 있는지 확인한다
5. 붙일 수 없는 문장은 추정이라고 표시하거나 지운다

세 번째와 네 번째 단계는 언제 내려갈지가 관건이다. 매번 PDF까지 내려가면 위키를 만든 의미가 없고 한 번도 내려가지 않으면 정리 과정에서 생긴 오류를 영영 발견하지 못한다. 기준은 답의 성격에 있다. 어떤 방법이 존재하는지, 어떤 계보에 속하는지 같은 질문은 wiki page 층에서 끝난다. 두 결과의 크기가 실제로 다른지, 어떤 조건에서만 성립하는지, 그림이 보여 주는 것이 본문의 주장과 같은지를 묻는 질문은 아래층으로 내려가야 한다. 다섯 번째 단계가 가장 지키기 어렵다. 언어 모델은 빈자리를 채우는 데 능하고 없다고 말하는 것보다 그럴듯한 문단을 만드는 쪽이 훨씬 쉽다. 없다는 답이 모이면 그 목록이 다음에 무엇을 위키에 들여야 하는지를 알려 주는 가장 정확한 자료가 된다. 위키가 무엇을 답하지 못하는지 아는 사람만이 위키를 계획적으로 키울 수 있고 빈틈이 매번 그럴듯한 문단으로 메워지면 그 목록은 영원히 만들어지지 않는다.

상위 열 건에 없으면 위키에 없다고 결론짓는 것도 같은 종류의 성급함이다. 2026년 7월 25일에 PDF 4,236건을 검수해 달라는 요청을 받고 휴리스틱으로 고른 30건만 눈으로 본 뒤 결과를 보고한 일이 있었다. 같은 날 11,029건을 확인했다고 말했는데 실제로 확인한 것은 제외 저널 지문 한 가지뿐이었다. 표본으로 전수를 대신했고 보고서에는 표본이었다는 사실이 남지 않았다. 지금 지침에 있는 규칙은 그 뒤에 생겼다. 전수 요청은 전수로 끝내고 시작할 때 대상 수를 확정하고 끝낼 때 분모와 분자를 숫자로 밝힌다. 추출 실패와 시간 초과로 판정하지 못한 파일은 이상 없음으로 묵지 않고 미검사로 따로 센다. 검색 결과 상위 몇 건만 읽고 답을 만드는 것은 같은 실패의 축소판이다. 순위를 어디에서 자를지는 사람이 정하는 것이므로 어떤 방법을 쓴 논문을 모두 찾아야 하는 질문에서는 카테고리 색인으로 다시 훑는다.

없다고 답하게 만드는 규칙은 하나뿐이고 짧다. 위키에 그 주제의 논문이 아예 없으면 없다고 말하고 PDF를 요청한다. 이 한 줄만으로는 지켜지지 않았고 실제로 습관을 바꾼 것은 그 옆에 붙은 금지였다. 답을 만들 때 웹 검색을 쓰지 않는다는 규칙을 지침에 박아 두었고 스킴이 요구하더라도 쓰지 않으며 사용자가 명시적으로 요청할 때만 연다. 검색이 열려 있으면 위키의 빈틈이 빈틈으로 남지 않는다. 밖에서 가져온 문장이 위키에서 나온 문장과 같은 형식으로 섞여 들어오고 몇 달 뒤에는 어느 문장이 이 위키가 실제로 담고 있는 것인지 구분할 수 없다. 위키 내용이 모자랄 때 허용한 보충 경로는 그 논문의 원문 PDF를 다시 읽는 것 하나이고 개관 페이지를 쓸 때도 출처는 위키 안의 논문으로 제한한다. 없다는 답이 늘어나는 것을 문제로 보지 않는 것도 함께 정해야 했다. 그 목록이 다음에 무엇을 들여야 하는지를 알려 주는 자료이므로 없다는 답이 줄었다는 사실 자체는 위키가 좋아졌다는 근거가 되지 못한다.

여러 논문을 연결하는 Deep Synthesis

여러 논문에 걸친 질문에 답할 때 가장 흔한 실패는 논문별 요약물 순서대로 이어 붙이는 것이다. 다섯 편의 요약을 나란히 놓으면 분량은 다섯 배가 되지만 질문에 대한 답은 하나도 만들어지지 않는다. 요약은 논문 안에서 끝나고 질문이 묻는 것은 논문 사이에 있다. Deep Synthesis는 각 논문의 설계와 결과와 경계를 꺼내어 서로 겹쳐 놓는 작업이고 겹쳐 놓을 축을 정하는 것이 작업의 절반이다. 축은 일곱 개로 정리된다. 연구 대상과 조건이 같은지, 같은 개념을 서로 다른 측정으로 정의했는지, 결과의 방향과 크기가 일치하는지, 주장의 수준이 관찰인지 예측인지 섭동 실험인지 임상 연관인지, 서로 다른 결과가 실제 모순인지 아니면 집단과 시점과 측정 방식의 차이인지, 새 논문이 기존 주장을 강화하는지 좁히는지 반박하는지 대체하는지, 그리고 아직 답하지 못한 경계 조건과 다음 질문이 무엇인지다. 일곱 축을 모두 채울 필요는 없다. 질문에 따라 셋이나 넷만 살아 있고 나머지는 해당 사항 없음으로 남는 것이 정상이다.

1. 일곱 개 축 가운데 이번 질문에서 살아 있는 축만 골라 표로 채운다
2. 답이 나오지 않은 축은 빈칸으로 두지 말고 확인 불가라고 적는다

단일세포 foundation model 세 편은 같은 목표를 두고 있지만 신뢰를 만드는 방식이 서로 다르다. 전이 학습으로 네트워크 생물학의 예측이 가능하다는 결과(Theodoris 외, 2023)는 모델이 지목한 것을 실험적으로 이미 알려진 사실과 대조해 신뢰를 얻는다. 판정 기준이 모델 바깥에 있고 그 기준이 되는 지식이 얼마나 갖춰진 영역인지에 결과가 달려 있다. 이듬해 나온 단일세포 다중오믹스 foundation model(Cui 외, 2024)은 다르게 움직인다. 여러 과제에서 기존 방법들과 나란히 놓고 이겼는지를 보이는 방식이므로 판정 기준이 비교 대상으로 고른 방법의 집합에 달려 있다. 비교군을 다르게 고르면 결론도 달라질 수 있다는 뜻이다. 세 번째 논문은 또 다른 방식을 쓴다. 대규모 학습 모델을 제안한 연구(Hao 외, 2024)는 자기 모델을 단독으로 평가하기보다 기존 분석 파이프라인 안에 부품처럼 끼워 넣고 그 파이프라인의 성능이 올라가는지를 보인다. 판정 기준이 그 파이프라인의 원래 성능이므로 어떤 파이프라인에 끼웠느냐가 결과의 크기를 좌우한다. 세 논문을 이렇게 놓고 나면 처음 질문이 왜 답하기 어려웠는지가 보인다. 어느 모델이 더 좋은가라는 질문은 세 논문의 성능 수치가 같은 축 위에 있다고 전제하는데 축이 서로 다르다. 축이 다르다는 것을 발견하는 일이 종합이고 그 발견은 요약을 아무리 많이 이어 붙여도 나오지 않는다.

주장의 수준을 묻는 축은 비교표를 만들기 전에 반드시 채워야 한다. 어떤 결과가 관찰인지, 예측인지, 섭동 실험으로 확인한 것인지, 임상 결과와 연관 지은 것인지에 따라 같은 문장이 뜻하는 바가 달라지기 때문이다. 세포 상태를 잘 구분한다는 결과와 어떤 유전자를 건드리면 세포 상태가 바뀐다는 결과는 문장 길이가 비슷해도 요구하는 근거의 양이 다르다. 앞의 결과는 모형이 자료의 구조를 잡았다는 뜻이고 뒤의 결과는 인과를 요구한다. 비교표의 각 칸에 수준을 함께

적어 두면 나중에 그 표를 인용하는 사람이 과장하지 않게 된다. 대상 측은 더 단순한 방식으로 비교를 막기도 한다. 같은 주제로 함께 읽은 나머지 두 편은 DNA 서열을 입력으로 쓴다(Boshar 외, 2025; Avsec 외, 2026). 다섯 편을 한 표에 넣고 성능을 나란히 적고 싶어지지만 입력이 다르고 예측 대상이 다르므로 같은 열에 놓을 수 없다. 비교할 수 없다는 판정도 종합의 결과이고 종합 문서에 그렇게 적어 두면 다음번에 같은 표를 만들려는 시도를 막는다. 비교 불가라고 적힌 한 문장이 잘못 만든 표 하나보다 오래 쓰인다. 세 모델의 학습 자료가 서로 얼마나 겹치는지처럼 논문 wiki page만으로 답할 수 없고 source note로도 부족한 측은, 지우지 않고 열린 질문으로 남기는 것이 다음 작업을 만든다.

논문마다 종합에 들어는 힘을 갈게 두지는 않는다. 영향이 큰 논문에는 더 무거운 처리를 규정해 두었다. 관련 overview의 본문을 실제로 갱신하고 여러 논문에 반복해 나타나는 개념에는 concept 페이지를 만들고 관련 페이지 사이에 양방향 링크를 걸고 그 논문이 기존 주장을 어떻게 바꾸는지를 문장으로 밝힌다. 분야의 논지 자체를 움직이는 논문에는 더 느린 방식을 쓴다. 여러 편을 묶어 처리하지 않고 한 편씩 처리하며 새 논문을 읽기 전에 기존 종합 페이지를 먼저 읽고 읽은 뒤에 주장이 어느 방향으로 움직였는지를 적는다. 두 방식을 가르는 기준은 그 논문이 위키의 다른 문장을 몇 개나 고치게 만드는가다. 종합의 결과물은 세 형태로 나온다. 이미 있는 overview의 본문을 고치는 것이 첫째이고 여러 질문에서 반복해 나타나는 개념이나 기전을 concept page로 만드는 것이 둘째이며 아직 근거가 얇은 열린 문제를 question page로 남기는 것이 셋째다.

같은 축으로 여러 편을 채고 나서야 보이는 것이 있다. 위키에 남아 있는 다섯 편의 구조 분석을 나란히 놓으면 foundation model 논문들이 신뢰를 얻는 방식이 서로 다르다는 것이 드러난다. Geneformer는 예측을 외부의 독립적인 대규모 연구 결과와 맞춰 보이는 방식으로 신뢰를 얻는다. 반면 scGPT와 NTv3와 AlphaGenome은 같은 일을 하는 기존 모델들과 정면으로 겨루어 이겼다는 것으로 신뢰를 얻는다. 네 번째로 분석한 scFoundation은 둘 중 어느 쪽도 쓰지 않고 자기 표현을 기존 파이프라인에 끼워 넣어 그 파이프라인 자체보다 나아졌음을 보인다. 이 세 번째 방식은 앞의 세 편을 읽을 때 축 목록에 없던 것이고 네 번째에 이르러 처음 이름이 붙었다. 축을 고정해 두었기 때문에 새 방식이 나타났을 때 그것이 새 방식이라는 것을 알아볼 수 있었다. 이 차이를 어느 수위로 말할 수 있는지는 분명히 해 둘 필요가 있다. 다섯 편으로 보인 것은 이 다섯 편이 각각 무엇에 기대어 설득하는가이고 foundation model 일반이 어느 방식으로 수렴하고 있다는 주장은 다섯 편으로 세워지지 않는다.

질문을 축으로 여러 논문을 묶은 종합 페이지

질문형 종합 페이지는 여러 논문의 공통 결론, 서로 다른 조건, 남은 불확실성을 한 문서에 모은다. 이 문서가 다음 질의와 다음 논문 수집의 출발점이 된다.

이 문서는 질문 하나를 제목으로 달고 그 아래에 현재 판단과 연구 사이의 차이와 남은 불확실성을 나눠 적는다. 여러 논문의 결론이 겹치는 부분과 갈리는 부분을 함께 적어 두는 것이 핵심인데, 겹치는 부분만 적으면 그 질문이 이미 답해진 것처럼 보이기 때문이다. 남은 불확실성 칸을 보면 다음에 읽을 논문을 정할 수 있다.

답을 위키로 되돌리기

위키의 설계 원칙에는 질의에 대한 답을 overviews 페이지로 저장하는 것이 지식이 복리로 붙는 방식이라고 적혀 있다. 그 문장이 원칙 문서에 들어가 있다는 자리가 중요하다. 저장된 질문 처리의 마지막 단계로 규정되어 있고 여유가 있을 때 하는 정리 작업으로 두면 대개 하지 않게 된다. 그렇다고 모든 답을 저장하지는 않는다. 저장을 남발하면 검색 후보만 늘고 어느 문서가 현재의 판단인지 알 수 없게 되므로 되돌릴 것을 고르는 기준이 필요하다. 네 가지 중 하나를 만족할 때 저장한다. 여러 문서를 비교해 다시 사용할 수 있는 판단을 만들었을 때, 기존 overview나 concept를 실제로 고칠 만한 새로운 관계가 생겼을 때, 독립된 연구 질문과 잠정적인 답이 만들어졌을 때, 그리고 프로젝트가 실행할 분석이나 의사결정이나 인수인계가 생겼을 때다. 조건에 걸리지 않는 답, 예컨대 문서 하나를 읽고 옮긴 사실 확인이나 이미 있는 문서를 다른 말로 쓴 것은 저장하지 않는다. 무엇을 저장하든 세 가지를 함께 붙인다. 어느 page와 어느 source note와 어느 원문에서 왔는지의 경로, 이 판단이 기존 지식을 강화하는지 좁히는지 반박하는지 대체하는지의 판정, 그리고 이 판단이 뒤집힐 조건이다. 세 번째가 가장 자주 빠지고 가장 오래 쓰인다. 표본이 더 큰 연구가 나오면, 다른 조직에서 재현되지 않으면, 측정 방식이 바뀌면 이 판단은 다시 봐야 한다고 적어 두면, 반년 뒤에 새 논문을 넣는 사람이 무엇을 확인해야 하는지 알게 된다. 저장 위치는 문서의 성격으로 정한다.

결과	저장 위치
분야의 현재 지식 종합	wiki/overviews/
안정된 개념, 방법, 기전	wiki/concepts/
근거가 아직 얇은 열린 질문	wiki/questions/

결과	저장 위치
프로젝트 실행안, 원고 전략, 비공개 작업	agenda/{context}/

마지막 줄의 구분이 실무에서 자주 흐려진다. 아직 공개되지 않은 데이터에 대한 판단, 특정 원고의 전략, 심사와 관련된 내용은 작업 폴더에 둔다. 위키는 나중에 공유하거나 인용할 수 있는 총이고 작업 폴더는 맥락 안에서만 유효한 총이기 때문이다. 두 층을 섞으면 위키를 공유할 때마다 무엇을 빼야 하는지 골라내야 하고 그 작업은 매번 실수를 부른다. 위키 안에서 세 곳 중 어디로 갈지는 그 판단이 얼마나 안정적인지로 갈린다. Overview는 어떤 주제에 대해 지금까지 알려진 것을 모아 둔 문서이므로 새 논문이 들어올 때마다 본문이 조금씩 고쳐지는 것이 정상이다. Concept은 여러 질문에서 반복해 나타나는 개념이나 방법이나 기전을 다루므로 자주 바뀌지 않고 바뀔 때는 정의나 적용 범위가 달라진 것이어서 파급이 크다. Question은 아직 답이 없다는 사실 자체를 기록하는 문서이므로 근거가 쌓이면 overview로 흡수되거나 concept으로 승격되고 그때 사라진다. 근거로 삼은 위키에서 세 층의 크기는 overview 590건, concept 408건, question 550건으로 비슷하다. 답하지 못한 것을 기록하는 문서가 종합 문서와 거의 같은 수라는 것은, 위키가 아는 것만큼 모르는 것도 적고 있다는 뜻이다.

1. 재사용할 수 있는 비교가 나왔으면 overview, concept, question 중 한 곳에 저장한다
2. 근거 경로와 판정과 뒤집힐 조건을 함께 적는다

저장의 형태에서는 새 문서를 만드는 것보다 기존 문서의 본문을 고치는 쪽이 먼저다. 새 문서를 만들면 손쉽게 결과물이 생긴 것처럼 보이지만 아무것과도 연결되지 않은 문서가 하나 늘어난 것일 수 있다. 근거로 삼은 위키에는 카테고리별로 종합 문서와의 연결 비율과 고립된 페이지 수를 집계하는 감사 스크립트가 있는데 저장을 새 문서 만들기만으로 처리하면 그 수치가 그대로 나빠진다. 기존 overview의 한 문단을 고치고 새로 읽은 논문 page와 양방향으로 잇는 편이 문서 하나를 새로 만드는 것보다 대개 이익이 크다. 고칠 곳을 찾는 일이 새로 쓰는 일보다 오래 걸린다는 점이 이 순서를 지키기 어렵게 만든다. 어느 overview가 이 판단을 담아야 하는지 알려면 그 overview를 먼저 읽어야 하고 읽고 나면 고칠 자리가 한 곳이 아닌 경우도 있다. 새 문서를 만드는 데는 그 읽기가 필요 없다. 절차가 규정으로 적혀 있는 이유가 여기에 있다.

Index를 최신으로 유지하는 법

검색 인덱스는 정보에서 만들어 낸 파생 상태다. 정보는 언제나 Markdown 문서이고 인덱스는 문서에서 다시 만들 수 있다. 둘을 나누는 것이 중요한 이유는 인덱스를 다루는 방식이 정보와 완전히 달라야 하기 때문이다. 인덱스는 백업하지 않고 손실되면 다시 만들며 여러 대의 컴퓨터가 각자 만들어 쓴다. 이 책의 검색 환경에서는 Mac마다 검색 실행 환경, 색인 버전, 빌드 잠금 파일, 마지막 성공 기록을 기계 안에 따로 둔다. 색인과 잠금 파일은 크기가 크고 실행할 때마다 자주 바뀌므로 두 기계가 하나의 파일을 동시에 고치면 충돌하기 쉽다. 동기화 폴더에는 원문 PDF와 Markdown 문서만 두고, 이 문서에서 다시 만들 수 있는 검색 색인과 실행 환경은 기계마다 따로 만든다.

작은 위키에서는 인덱스가 필요 없다. 문서가 수십 건일 때는 목차 문서 하나로 어디에 무엇이 있는지 다 보인다. 근거로 삼은 위키처럼 wiki page가 17,591건이고 카테고리가 50개가 되면 사정이 달라진다. 목차 문서에 모든 문서를 나열하면 목차 자체를 읽을 수 없고 카테고리 이름만 나열하면 카테고리 안에서 다시 헤매게 된다. 그래서 탐색 경로를 둘로 나눈다. 목차 문서와 카테고리별 색인 문서가 구조를 따라 내려가는 결정적 경로를 맡고 어휘 색인이 질문의 단어와 맞는 후보를 끌어올리는 역할을 맡는다. 의미 검색기를 붙여 본 뒤에도 기본 경로로 삼지 않고 선택적 대체 경로로만 남겨 둔 데에는 이유가 있다. 어휘 색인은 동기화된 Markdown 문서에서 직접 만들어지므로 중간에 어떤 서비스도 끼지 않고 결과가 결정적이어서 어제와 오늘의 차이를 문서의 변화로 설명할 수 있다. 의미 검색기를 기본 경로에 놓으면 그 도구가 멈추거나 모델이 바뀔 때 위키 전체가 검색되지 않는 상태가 되는데 기본 경로를 상위 의존 없이 홀로 서게 두면 그런 일이 생기지 않는다. 두 경로는 서로를 검사한다. 검색으로 나온 후보가 카테고리 색인에서 보이지 않으면 분류가 빠진 것이고 카테고리 색인에 있는데 검색에 안 걸리면 문서의 표현이 질문의 언어와 어긋난 것이다.

인덱스를 유지하는 데 필요한 것은 네 종류의 도구다. 문서 전체에서 인덱스를 만드는 것, 만들어진 인덱스에서 검색하는 것, 정해진 시각에 자동으로 다시 만드는 것, 그리고 카테고리별 색인 문서를 다시 생성하는 것이다. 도구를 무엇으로 쓰느냐보다 언제 돌리느냐가 실제 운영을 가른다. 정기 갱신은 매일 새벽 2시에 예약 작업으로 둔다. 사람이 일하지 않는 시간이고 기계마다 색인과 잠금을 자기 홈 아래에 두었기 때문에 두 대에 같은 예약을 걸어도 서로 간섭하지 않는다. 논문을 들인 직후에 색인을 자동으로 다시 만드는 장치는 두지 않았다. 논문 한 편을 넣을 때마다 전체 인덱스를 다시 만드는 것은 낭비이고 자동 재구축을 들이기 절차에 붙이면 논문을 넣는 일이 색인 상태에 의존하게 된다. 방금 넣은 문서를 바로 검색해야 할 때, 대량으로 논문을 들인 직후, 카테고리를 재구성한 직후에만 사람이 직접 돌린다.

1. 새 문서를 저장한 뒤 인덱스를 직접 다시 만든다

2. 처음 던졌던 완전한 질문으로 다시 검색한다

재검색에서 새로 만든 문서가 상위에 올라오고 그 문서 하나로 처음에 읽은 다섯 문서를 대신할 수 있으면, 다음 세션의 에이전트는 같은 질문에 절반의 시간으로 답한다. 다시 만들 때는 현재 문서를 직접 읽어 새 버전을 통째로 만들고 문서 수와 내용이 온전한지 확인한 다음에 검색이 바라보는 자리를 그 버전으로 한 번에 바꾼다. 이전 버전은 지우지 않고 남겨 두어, 새 버전에 문제가 있으면 되돌릴 수 있게 한다. 만들면서 기존 인덱스를 조금씩 고치면 도중에 끊겼을 때 일부 문서가 빠진 채로 검색이 계속 동작하는데 조용히 일부를 놓치는 검색이 아예 실패하는 검색보다 훨씬 위험하다. 검색 결과가 이상할 때 처음 의심할 것도 인덱스다. 어제 넣은 논문이 검색에 나오지 않으면 대개 인덱스가 낡은 것이다. 위키에 그 논문이 없다고 결론을 내리기 전에 인덱스의 생성 시각을 먼저 본다. 순서를 반대로 하면 있는 논문을 다시 들어거나 있는 근거를 없다고 답하게 된다. 파생 상태를 정본과 혼동하면 정본을 고쳐야 할 문제와 다시 만들면 되는 문제가 구분되지 않는다.

10장. 논문의 구조를 해부하기

참고문헌 목록에는 논문의 동기와 방법과 데이터와 비교가 조립되어 있는데 결과 요약은 그 조립을 보여 주지 않는다. 구조 분석은 지적 계보, 신뢰를 만드는 방식, 학습 자료의 출처 구조, 자기 위키에 비어 있는 부분을 찾는다. 절차는 여섯 묶임이고 마지막 묶임은 다음에 읽을 논문의 목록을 만든다. 대학원 논문 발표 수업에서 반복해서 보게 되는 장면이 있다. 발표자가 분야를 대표하는 모델 논문 한 편을 맡아 30분 동안 잘 만든 슬라이드를 넘긴다. 문제 설정, 구조 그림, 성능 표, 저자가 밝힌 한계까지 빠진 것이 없다. 그런데 청중에서 “이 논문은 왜 하필 그 모델들과 비교했는가”, “학습에 쓴 데이터는 누가 모은 것인가”, “저자들이 넘어서려는 기존 관점은 어느 논문에 있는가” 같은 질문이 나오면 발표가 멈춘다. 발표자가 논문을 읽는 동안 그 질문을 하지 않았기 때문이고 세 질문의 답은 모두 참고문헌 목록과 본문 사이의 대응 관계에 들어 있다.

결과 요약과 구조 분석의 차이

논문을 읽고 남기는 기록은 대개 결과 요약이다. 무엇을 물었고 어떻게 했고 무엇을 얻었고 어디까지가 한계인가. 4장에서 만든 source note도 큰 틀에서는 그 형식을 따른다. 결과 요약이 답하는 질문은 하나다. 이 논문이 무엇을 주장했는가. 답이 정확하다면 그 기록은 제 일을 다 한 것이고 논문 한 편을 다시 열지 않고도 내용을 되살릴 수 있다. 구조 분석이 답하는 질문은 다르다. 그 주장은 무엇 위에 서 있는가. 대규모 사전학습으로 세로 상태를 예측했다는 논문이 있다고 하자. 요약은 그 문장에서 끝나지만 구조 분석은 예측이 믿을 만하다는 논증이 어디에서 나오는지 묻는다. 독립적으로 수행된 다른 대규모 연구의 결과와 예측이 겹쳐서인가, 같은 과제에서 기존 방법들보다 점수가 높아서인가, 아니면 기존 분석 파이프라인에 끼워 넣었더니 그 파이프라인이 나아져서인가. 세 답은 서로 다른 종류의 논증이고 무너지는 지점도 다르네, 요약문은 셋을 구분하지 않는다.

인용마다 이름표를 붙여 “3번 인용은 배경, 7번 인용은 benchmark”까지 적고 끝내면 재료만 만든 것이다. Introduction에서 한계로 지목된 논문이 Results에서 비교 대상으로 다시 나오고 Discussion에서 결과를 일반화하는 틀로 세 번째 나오는 경우가 있다. 세 번의 등장은 하나의 논증이다. 저자는 그 논문을 넘어서겠다고 선언하고 넘어서었다고 수치로 보이고 넘어서 결과를 다시 그 논문의 언어로 해석한다. 이름표만 세면 세 단계가 사라지고 등장 횟수 3만 남는다. 같은 읽기가 연구 논문 밖에서도 통한다. 정책 보고서의 각주를 순서대로 따라가 보면 어느 통계를 근거로 삼았는지, 반대 견해를 어느 대목에서 인용했는지, 그 견해를 어디에서 반박했는지가 드러난다. 인용의 배치가 문서의 논증을 보여 준다. 다큐멘터리 끝에 붙는 자료 출처, 판결문의 선례 인용도 같은 방식으로 읽힌다. 문서가 자기 주장을 어디에 기대어 세웠는지는 본문과 출처의 대응에 적혀 있다.

Deep Structure Analysis는 한 논문의 모든 인용을 등장한 section과 수행하는 기능에 따라 지도로 만들고 논지를 떠받치는 인용은 원문까지 읽어 그 논문의 지적 계보와 신뢰 전략과 자료 출처 구조를 드러내는 절차다. 이 책이 근거로 삼은 연구용 위키에는 절차를 규정한 개념 정보 문서가 있고 2026년 7월에 작성되었다. 지금까지 다섯 편의 논문에 적용되었고 다섯 결과는 모두 위키의 overview 층에 페이지로 남아 있다. 방법을 규정한 문서와 적용 결과가 같은 위키 안에서 서로를 가리키기 때문에, 새 논문에 적용할 때 이전 사례를 그대로 본으로 삼을 수 있다. 절차의 뿌리는 세 갈래다. 인용의 수사적 역할을 배경, 사용, 비교, 대조로 나누는 citation function analysis(Teufel 외, 2006)가 있고 인용 주변 문장을 읽어 인용한 이유를 복원하는 citation context analysis가 있으며 인용 관계를 기계가 읽을 수 있는 유형으로 규정한 CiTO가 있다. Deep Structure Analysis는 세 갈래의 특수화이고 거기에 더하는 것은 두 가지다. 하나는 section별 지도를 반드시 만든다는 것이고 다른 하나는 분석 결과를 자기 knowledge base의 보유 여부와 대조하는 단계다. 두 번째가 이 책의 맥락에서 결정적인데, 분석이 해석으로 끝나지 않고 위키의 작업 목록으로 이어지기 때문이다.

이 절차는 Geneformer 한 편을 분석하면서 만들어졌고 그 뒤에 이름이 붙어 다른 논문에도 같은 형태로 적용되었다. 요약으로 답이 나오지 않던 질문은 이 논문이 무엇에 기대어 설득하고 있는가였다. 요약은 논문이 무엇을 했고 어떤 결과를 얻었는지까지 알려 주고 거기서 멈춘다. 참고문헌을 처음 끝까지 대응시켜 보니 서로 다른 두 무리가 드러났다. 본문에서 논지를 떠받치는 인용이 46건이었고 Methods에 붙어 학습 자료가 어디서 왔는지를 기록하는 인용이 110건 남짓이었다. 두 무리를 섞어 세면 참고문헌 150여 건짜리 논문이 되지만 나누어 세면 지적 계보는 46건 규모이고 나머지는 자료 출처의 꼬리다. 같은 대응 작업에서 위키가 아직 가지고 있지 않은 open access 논문 6건도 함께 드러났고 그 목록이 다음에 읽을 것을 정하는 작업 목록이 되었다. 이 방법이 인용 기능 분석이라는 기존 연구 전통의 특수화라는 것은 정보 문서에 밝혀 두었으며 거기에 더한 것은 section별 지도와 위키가 그 인용을 이미 가지고 있는지 대조하는 단계 두 가지다.

인용이 맡는 여덟 가지 기능

구조 분석의 첫 도구는 인용이 그 논문 안에서 하는 일에 이름을 붙이는 것이다. 기능은 지금 분석 중인 논문 안에서 그 인용이 맡는 역할이다. 같은 논문이 어떤 논문에서는 비교 대상 baseline으로, 다른 논문에서는 label을 공급한 자료로, 또 다른 논문에서는 배경 생물학으로 인용된다. 4장에서 다른 카테고리화 혼동하지 않아야 한다. 카테고리는 위키 전체에서 그 논문을 어디에 두고 어떤 질문으로 다시 찾을지에 대한 결정이고 기능 분류는 논문 한 편 안에서의 위치다. 하나는 위키의 좌표이고 다른 하나는 논문의 좌표다. 축이 고정되어 있어야 논문 사이의 비교가 성립한다. 논문마다 다른 눈으로 읽으면 인상이 여러 개 생기고 그 인상들은 서로 비교되지 않는다. 축을 늘리는 것은 기존 축 어디에도 들어가지 않는 역할이 실제로 발견되었을 때이고 다섯 편을 분석하는 동안 그런 일이 한 번 있었다. 이름의 목록은 여덟 개이고 각 이름은 그때 던져야 할 질문과 짝을 이룬다.

기능	구조 분석에서 묻는 질문
Motivation / inspiration	이 연구를 시작하게 한 지적 씨앗과 대비 대상은 무엇인가
Method and engineering lineage	Architecture, objective, 학습 장치는 어디에서 왔는가
Core benchmarking	무엇과 정량 비교하며 어떤 외부 ground truth를 쓰는가
Label / supervision set	Fine-tuning과 평가에 쓰는 label은 누가 어떻게 정의했는가
Dataset / material provenance	원자료, atlas, cohort, 재료는 어디에서 왔는가
Background biology	결과를 생물학적으로 해석하는 근거는 무엇인가
Interpretation frame	결과를 일반화하는 데 쓰인 다른 분야의 틀은 무엇인가
Tools and resources	Software, database, annotation resource는 무엇인가

인용 하나에는 기능을 하나만 배정한다. 부차적인 역할이 있으면 괄호로 덧붙이되 집계에는 넣지 않는다. 두 개를 나란히 허용하면 애매한 인용마다 두 개를 붙이게 되고 그 순간 축별 집계가 의미를 잃는다. 하나로 강제해도 재사용은 사라지지 않는데 같은 인용이 여러 section에서 다른 역할을 하는 모습은 뒤에 만들 section별 지도가 보존하기 때문이다. 분류표는 무게중심을 재는 도구이고 지도는 움직임을 기록하는 도구다. 여덟 축 가운데 자료와 재료의 출처를 적는 축을 따로 떼어 놓은 데에는 이유가 있다. Foundation model 논문이나 자료를 만드는 논문은 Methods에 corpus를 어떻게 모았는지 기록하는 긴 인용 꼬리를 달고 그 꼬리가 참고문헌의 다수를 차지하기도 한다. 꼬리를 지적 계보와 같은 통에 넣으면 논지를 실제로 움직이는 인용이 그 안에 묻힌다. 꼬리를 따로 세고 따로 묶는 작업은 방법의 일부다.

원문과 인용의 연결

첫 작업은 분석 대상 논문의 원문 PDF 전체와 참고문헌 목록을 확보하는 것이다. 출판사 웹페이지를 브라우저로 저장한 파일이나 화면 인쇄본은 쓰지 않는데 3장에서 다른 운영 원칙이면서 동시에 이 작업에 고유한 이유가 있다. 웹 화면은 인용 번호를 링크로 감추고 참고문헌 목록을 접어 두는 경우가 많아, 본문 번호와 목록 항목의 대응이 애초에 끊긴 상태로 저장된다. 대응이 끊긴 자료로 시작하면 뒤의 모든 단계가 추측이 된다. PDF는 조판된 그대로를 보존하므로 번호와 항목이 함께 남는다. 대응 작업 자체는 단순하지만 조용히 틀리기 쉽다. 본문의 numeric citation이나 author-year citation을 참고문헌 목록의 각 항목에 하나씩 맞춰 나간다. 여기서 4장에서 말한 텍스트 추출 문제가 되돌아온다. 위첨자가 무너져 인용 번호가 본문의 일반 숫자와 섞이면 대응이 통째로 어긋나고 표의 각주 기호가 인용 번호처럼 보이는 경우도 있다. 대응이 끝났다고 선언하기 전에 목록의 마지막 번호와 본문에서 발견한 가장 큰 번호가 일치하는지, 목록에 있는데 본문 어디에서도 호출되지 않는 항목이 있는지 확인한다.

대응과 함께 등장 위치를 기록한다. Introduction, Results의 각 하위 절, Discussion, Methods, 그리고 Supplementary까지 나눈다. 같은 reference가 여러 곳에 나오면 각각의 등장을 따로 남긴다. 등장을 합쳐 한 줄로 만들면 앞에서 말한 세 단계 논증이 사라지고 그 논문이 세 번 쓰였다는 사실만 남는다. 위치를 남기는 비용은 줄 몇 개지만 되살리려면 원문을 처음부터 다시 읽어야 한다. 이 단계의 산출물은 표 하나다. 인용 번호, 서지 정보, 등장한 section, 그때 하는 일 한 줄. 이후의 모든 작업이 그 표 위에서 이루어진다. 대응을 만드는 일은 에이전트가 잘하고 사람이 하면 지치는 종류의 작업이므로 넘겨도 되지만 표본을 뽑아 대응이 맞는지 확인하는 일은 사람이 한다. 스무 건 가운데 다섯 건을 골라 원문의 해당 문장을 열어 보면 대응이 통째로 밀렸는지 아닌지가 금방 드러난다.

1. 이미 읽어 내용을 아는 짧은 논문 한 편에서 Introduction과 첫 Results 하위 절만 범위로 잡는다
2. 두 section에 등장하는 인용을 모두 뽑는다
3. 번호, 서지, 등장 위치, 그때 하는 일 한 줄로 표를 만든다
4. 표에서 인용 10건을 골라 여덟 쪽 가운데 하나씩 배정하고 애매한 것은 애매하다고 표시한다

인용문 읽기와 원문 읽기의 차이

구조 분석은 인용된 원문을 읽지 않을 때 가장 자주 무너진다. 분석 대상 논문이 “선행 연구는 X를 보였다(12)”라고 쓴 문장을 풀어 쓰고 12번 논문을 읽었다고 적는 일이 흔하다. 그 문장은 인용하는 쪽이 만든 요약이자 해석이고 대개 자기 논지에 맞게 각도가 잡혀 있다. 조건을 좁혀 인용하거나 부차적 결과를 주된 결과처럼 옮기거나 원문에는 없는 인과를 붙이는 일이 드물지 않다. 인용하는 쪽의 문장을 번역해 놓고 인용된 논문을 분석했다고 부르면, 그 오차가 그대로 위키에 들어와 다음 사람에게 사실로 전달된다. 논지를 떠받치는 인용에 대해서는 그 논문의 PDF 본문을 읽는다. 저자들이 실제로 무엇을 했는지, 어떤 데이터를 썼는지, 수치가 얼마였는지, 어디까지 주장했는지를 원문에서 확인한다. 읽어야 하는 것은 방법과 결과다. 인용하는 쪽이 “성능이 좋았다”고 옮긴 대목이 원문에서는 특정 조건에서만 성립했다는 것을 확인하는 순간, 두 논문의 관계에 대한 서술이 달라진다. 구조 분석의 깊이는 그런 확인이 몇 건 있었는지로 결정된다.

두 가지 대체는 명시적으로 금지되어 있다. 첫째는 인용된 논문의 PDF 대신 위키의 source note를 읽는 것이다. Source note는 원문에서 한 단계 건너온 기록이고 이 절차에서는 원문을 찾아가는 locator로만 쓴다. 둘째는 초록이나 요약 카드, 출판사 landing page로 본문 전체를 대신하는 것이다. 두 대체 모두 결과를 요약의 요약으로 떨어뜨리고 떨어진 뒤에는 그 사실이 결과물에 남지 않기 때문에 나중에 구분할 수도 없다. 작업 순서는 네 단계다. 먼저 모든 인용을 원문 PDF를 보유한 것과 아닌 것으로 나눈다. 다음으로 논지를 떠받치는 인용 가운데 보유한 것의 본문을 읽는다. 보유하지 않은 것은 합법적으로 원문을 구한 뒤 본문을 읽는다. 끝내 구하지 못한 것은 인용 정보와 확보를 시도한 경로를 사람에게 넘긴다. 깊이 읽기에 해당하는 서술은 이 네 단계가 끝난 뒤에 시작한다.

1. 10건 가운데 논지를 떠받치는 인용 2건을 고른다
2. 그 두 논문의 PDF 본문을 실제로 읽고 방법과 수치를 확인한다
3. 인용하는 쪽의 서술과 다른 점이 있으면 적는다

이 확인을 건너뛴 분석은 새 세션을 시작한 에이전트에게 이 논문이 무엇으로 신뢰를 얻는지 물었을 때 근거를 내놓지 못한다. 구하지 못한 인용은 구하지 못했다고 적는다. 결과물에 접근 불가 목록이 그대로 남고 그 목록 자체가 정보다. 가장 나쁜 결과는 읽지 못한 논문에 대해 제목과 초록에서 추론한 그럴듯한 두 문장을 써 넣는 것이다. 그 두 문장은 검증할 수 없고 다음에 이 페이지를 읽는 사람은 원문을 확인한 서술과 구분하지 못한다. 확인한 것과 확인하지 못한 것의 경계를 문서에 남기는 일이 이 방법이 지키려는 최소한이다. 인용 전체의 원문을 읽을 수는 없으므로 무엇이 논지를 떠받치는 인용인지 판단해야 한다. 판단 기준은 논증이 그 인용에 실제로 기대고 있는지다. 정량 비교의 대상이 된 baseline, label을 공급한 자료, 동기에서 넘어서겠다고 지목한 대비 대상, 결과 해석의 틀이 여기에 해당한다. Methods의 자료 출처 꼬리는 대개 여기에서 빠진다. 다만 그 꼬리를 읽지 않는 것과 세지 않는 것은 다른 문제이고 크기와 구성은 반드시 센다.

Section별 인용 지도

지도를 만드는 목적은 각 section이 논증에서 맡은 일을 복원하는 것이다. 분야와 저널이 달라도 논문은 대체로 같은 흐름을 지낸다.

↓

↓ architecture dataset lineage

↓ baseline, metric, ground truth

↓ biology

각 화살표에 인용이 붙는다. 문제 정의에서 동기로 넘어가는 화살표에는 motivation 축의 인용이 오고 방법 선택에는 method lineage와 자료 출처 인용이, 비교와 검증에는 baseline과 label과 ground truth가, 결과 해석에는 background biology와 interpretation frame이 온다. 어느 화살표에 인용이 하나도 없다면 그 빈칸도 정보다. 비교 대상 없이 성능을 제시했거나 해석의 근거를 본문 안에서만 만들었다는 뜻일 수 있다. 지도는 있는 인용을 정리하는 동시에 없는 인용을 드러낸다. 지도는 구조 분석에서 가장 두꺼운 산출물이 된다. Section마다 인용을 하나씩 짚으면서 그 위치에서의 역할, 다른 인용과의 연결, 논문이 실제로 제시한 수치를 함께 적는다. 지도라고 부르는 이유는 위치가 보존되기 때문이다. 나중에 “이 논문이 어떤 근거로 그 주장을 했는가”라는 질문이 들어오면 지도의 해당 section만 열면 되고 원문을 처음부터 다시 읽지 않아도 된다. 논문 한 편의 지도가 원문보다 길어지는 경우도 있는데 길이를 줄이려고 위치를 합치면 지도가 또 하나의 요약으로 주저앉는다. 지도가 잡아내는 것 가운데 요약으로는 절대 나오지 않는 종류가 있다. AlphaGenome(Avsec 외, 2026)의 구조 분석에서는 서론이 어떤 도구의 이름과 함께 단 인용 번호가 실제로는 다른 논문을 가리키고 있고 그 도구의 논문은 다른 번호에 있다는 것이 드러났다. 출판된 논문에 남은 인용 번호 오류이고 본문을 읽는 것만으로도 참고문헌을 읽는 것만으로도 보이지 않는다. Section별로 번호와 내용을 하나씩 맞춰 볼 때만 어긋남이 표면으로 올라온다. 대응을 끝까지 만들어야 하는 근거가 여기에 있다.

Intellectual core와 provenance tail의 분리

인용을 두 집단으로 갈라 따로 세는 것이 이 장의 방법적 핵심이다. 한쪽은 Introduction과 Results와 Discussion에서 논지를 직접 움직이는 인용이고 다른 쪽은 Methods에서 corpus와 도구의 출처를 기록하는 인용이다. 두 집단은 성격이 다르고 규모도 대개 크게 다르다. 섞어 놓으면 참고문헌 전체가 하나의 덩어리로 보이고 그 덩어리의 크기는 논문에 대해 거의 아무것도 말해 주지 않는다. 갈라 놓아야 계보가 보인다. Geneformer(Theodoris 외, 2023)의 구조 분석이 그 차이를 처음 보여 주었다. 논지를 움직이는 main-text 인용은 46건이었고 Methods 층에서 자료와 도구의 출처를 기록하는 인용은 약 110건이었다. 참고문헌의 다수가 데이터가 어디에서 왔는지를 적는 데 쓰인 셈이다. 두 집단을 섞으면 46건이 만드는 논증이 110건 안에 묻히고 이 논문이 어떤 전통을 이어받았는지가 보이지 않는다. 46건만 따로 놓고 기능별로 분류했을 때 비로소 동기와 방법과 비교가 각각 어디에서 왔는지가 드러났다.

NTv3(Boshar 외, 2025)에서는 같은 분리가 세 층으로 나뉘었다. Main-text의 지적 인용 64건, Methods 층의 공학, 도구, 자료 인용 44건, 그리고 Supplementary에서 해석에 쓰인 인용 4건이다. Supplementary의 4건은 문서상의 위치로만 보면 부록이지만 하는 일은 main text의 해석 인용과 같다. 나누는 기준은 인용이 하는 일이고 이 사례가 그 점을 분명히 한다. 위치는 출발점이고 최종 배정은 그 인용이 논증의 어느 자리에 서 있는지로 정한다. 두 집단의 크기 비율은 그 논문이 어떤 종류의 작업인지를 말한다. 자료 출처 꼬리가 참고문헌의 다수인 논문은 자료를 모으고 정리하는 일 자체가 기여의 큰 부분이다. 꼬리가 거의 없고 목록의 대부분이 경쟁 모델인 논문은 기여가 비교 우위에 있다. 신뢰가 만들어지는 자리가 서로 다르다는 뜻이고 따라서 그 논문을 의심할 때 봐야 할 곳도 다르다. 앞의 논문은 자료의 편향을, 뒤의 논문은 비교 대상 집합의 폭을 먼저 본다.

꼬리가 짧아지는 방향의 변화도 두 집단을 갈라 놓아야 보인다. Single-cell foundation model인 scFoundation(Hao 외, 2024)은 참고문헌 75건 가운데 68건이 main-text에서 일을 하고 5천만 세포 규모의 학습 corpus를 기록하는 인용은 약 12건뿐이다. 개별 데이터셋 수백 건을 하나씩 인용하는 대신 자료를 모아 둔 집계 repository 몇 개를 통해 인용했기 때문이다. AlphaGenome은 한 걸음 더 갔다. 참고문헌 67건 가운데 5,930개 track으로 이루어진 학습 corpus를 기록한 것이 거의 없고 출처는 Supplementary의 자료 목록으로 통째로 넘어갔다. 꼬리가 짧아진 것은 자료의 출처가 참고문헌에서 보이지 않게 되었다는 뜻이며 인용 건수만 세는 분석은 그 차이를 놓친다. 그래서 이 단계는 문장 하나를 결론으로 남긴다. 이 논문의 참고문헌에서 지적 계보에 해당하는 것은 몇 건이고 자료의 출처는 어디에 기록되어 있으며 기록되지 않은 부분은 어디로 밀려났는가. 세 질문에 답하고 나면 그 논문을 재현하거나 검증하려 할 때 무엇을 먼저 구해야 하는지가 함께 정해진다. 출처가 Supplementary 목록으로 넘어간 논문은 그 목록을 구하지 못하면 학습 자료를 확인할 방법이 없고 그 사실 자체가 그 논문에 대한 중요한 판단 재료다.

1. 이 논문의 main-text 인용 수와 Methods 층 인용 수를 각각 센다
2. 두 수를 나란히 적고 비율이 무엇을 뜻하는지 한 문장으로 쓴다

인용 관계의 도식화

분류표는 인용을 통해 담고 도식은 인용 사이의 방향과 의존을 보여 준다. 구조 분석의 결과물은 위키의 overview 층에 두는 페이지이고 관계 도식은 그 페이지 안에 mermaid flowchart로 그린다. 노드는 인용이나 모델이 되고 화살표에는 관계의 종류를 적는다. 표로 대신할 수 없는 이유는 방향이 표에 담기지 않기 때문이다. 어떤 논문이 다른 논문을 이겼다는 것과 그 논문에 기대고 있다는 것은 표에서 같은 칸에 들어가지만 도식에서는 반대 방향의 화살표가 된다. 도식을 그리다 보면 지도에서 놓친 것이 드러난다. 화살표가 들어오기만 하고 나가지 않는 노드, 어느 화살표에도 붙지 못하는 인용이 눈에 띈다. 뒤의 경우는 분류가 틀렸거나 그 인용이 실제로는 논지에 기여하지 않는다는 뜻이고 둘 다 확인할 가치가 있다. 도식은 결과를 보여 주는 그림이면서 앞 단계의 검산이기도 하다. 그리기 어려우면 대개 지도가 아직 덜 만들어진 것이다. 화살표에 붙는 관계는 다섯 종류다.

- Validation chain: 예측이 어떤 독립 연구의 결과와 대조되어 검증되었는가.
- Self-reference lineage: 같은 연구집단의 이전 결과물이 어떻게 이어졌는가.
- Multi-role reuse: 같은 인용이 서로 다른 section에서 다른 역할로 다시 쓰였는가.
- Benchmark contrast: 무엇을 정량적으로 이겼다고 주장하는가.
- 자료 출처: 학습과 평가에 쓰인 자료가 어디에서 왔는가.

flowchart TD

```
M[ ] -->| : | P[ ]
L[ ] -->| | P
A[architecture ] -->| | P
R[ repository] -->| | P
P -->| | B[baseline ]
P -->| | G[ ]
G -->|Discussion | P
```

1. 앞에서 만든 표를 보고 인용 관계를 mermaid flowchart 한 장으로 그린다
2. 노드는 열 개를 넘기지 않는다

위키 coverage의 교차 점검

마지막 묶음이 이 방법을 일반적인 인용 분석과 갈라놓는다. 인용 하나하나에 대해 그 논문을 지금 위키가 보유하고 있는지 확인하고 보유한 것은 wikilink로 연결한다. 연결이 끝나면 분석 대상 논문의 페이지에서 그 논문이 기대고 있는 논문들의 페이지로 실제 경로가 생긴다. 경로가 생겼다는 것은 다음 세션의 에이전트가 “이 논문의 비교 대상이 어떤 방법이었는데”를 물었을 때 위키 안에서 답할 수 있다는 뜻이다. 연결하지 않으면 분석 결과는 잘 쓴 문서 한 편으로 남고 위키의 나머지와의 이어지지 않는다. 보유하지 않은 논문의 목록이 작업 목록이 된다. Geneformer 분석에서는 위키가 가지고 있지 않은 open-access 논문 6건이 이렇게 드러났다. 6건은 분석이 만든 가장 실용적인 결과다. 무엇을 읽을지 고르는 문제가 여기서 자동으로 풀리는데 지금 이해하려는 논문이 실제로 기대고 있는 논문들이 후보가 되기 때문이다. 관심 가는 제목을 모아 대기열을 만드는 것과는 다른 방식으로 다음 읽기가 정해진다.

1. 10건 가운데 지금 위키에 없는 것을 목록으로 만든다
2. 각 항목에 원문 확보 가능 여부를 적는다

원문을 구하는 방법에도 규칙이 있다. 참고문헌 목록의 링크를 그대로 누르는 대신 제목으로 저장소를 검색한다. 링크는 죽어 있거나 접근 권한 뒤에 있을 수 있고 같은 논문이 다른 저장소에 공개되어 있는 경우가 많다. 한 저장소에서 찾지 못했다고 해서 어디에도 없다고 결론 내리지 않는다. 여러 경로를 시도한 뒤에도 구하지 못하면 구하지 못했다는 사실과 이유를 기록하고 그 항목은 다음에 접근 권한이 생겼을 때 다시 시도할 대상으로 남는다. Coverage 목록은 별도의 대기열 문서로 남기고 항목마다 인용 정보와 원문 확보 가능 여부와 확보를 시도한 경로를 적는다. 다음 ingest는 이 대기열의 위에서부터 가져가고 대기열이 비면 그 논문의 계보가 위키 안에서 이어진 것이며 남은 항목은 이 위키가 아직 답할 수 없는 영역이다. 확보한 논문은 4장의 ingest 절차로 들어간다. 구조 분석 한 번이 여러 건의 ingest를 만들고 그 ingest가 끝나면 다시 이 논문의 구조 분석 페이지에 링크가 채워진다. 순환이 한 바퀴 도는 셈이다. 이 순환을 몇 번 돌리면 위키에서 그 주제의 계보가 실제로 이어지기 시작하는데 논문이 개별 페이지로 쌓이는 것과 계보가 이어지는 것은 다른 상태다. 앞의 상태에서는 새 논문이 늘어도 답할 수 있는 질문이 늘지 않고 뒤의 상태에서는 논문 한 편이 들어올 때마다 그 주변 판단이 함께 갱신된다. 결과물은 overview 층의 페이지 하나로 저장하고 양쪽에 양방향으로 연결한다. 분석 대상 논문의 페이지에서 구조 분석 페이지로, 구조 분석 페이지에서 Deep Structure Analysis 개념 문서로, 그리고 반대

방향으로도 연결한다. 한쪽에만 링크가 있으면 개념 문서에서 적용 사례를 찾을 수 없고 사례에서 방법을 찾을 수 없다. 다섯 번째 논문을 분석할 때 첫 번째 논문의 페이지를 본으로 삼을 수 있었던 것은 개념 문서가 다섯 사례를 모두 가리키고 있었기 때문이다.

같은 축으로 다섯 편을 재면

반복 가능한 방법의 값은 같은 축으로 두 번째 논문을 재는 순간 나온다. 그때 첫 번째 논문에 대해 몰랐던 것이 보인다. 다섯 편의 foundation model 논문에 같은 여덟 축을 적용한 결과, 신뢰를 만드는 방식이 세 갈래로 갈렸다. 갈래는 논문의 성능이나 인기와 무관하고 논증이 무너질 수 있는 지점이 서로 다르다는 데 의미가 있다. 한 편만 읽어서는 자기가 읽은 방식이 유일한 방식인 줄 알게 된다. 첫 번째 갈래는 외부 ground truth와의 일치다. Geneformer의 논증은 모델의 예측이 독립적으로 수행된 대규모 연구의 결과와 얼마나 겹치는지에 기댄다. 인용 구조에서는 core benchmarking 축에 독립 연구가 들어와 있는 모습으로 나타난다. 이 전략의 강점은 비교가 방법 바깥에서 온다는 것이고 약점은 외부 연구가 틀렸거나 측정 조건이 다를 때 논증 전체가 함께 흔들린다는 것이다. 그래서 이 갈래의 논문을 검토할 때 먼저 확인할 것은 대조에 쓰인 그 외부 연구다.

두 번째 갈래는 방법 대 방법 벤치마킹이다. Single-cell foundation model인 scGPT(Cui 외, 2024)의 논증은 같은 과제에서 기존 방법들보다 점수가 높다는 데 기대고 인용 구조에서 core benchmarking 축은 경쟁 모델들로 채워진다. NTV3와 AlphaGenome도 같은 갈래에 속한다. AlphaGenome은 극단에 가까운데, 참고문헌 67건이 사실상 modality마다 하나씩 이겨 낸 경쟁 모델의 계보에 가깝고 자료 출처는 본문 밖으로 나가 있다. 이 갈래의 약점은 비교 대상 집합이 좁으면 우위의 의미도 함께 좁아진다는 것이므로 검토는 어떤 모델이 비교에서 빠졌는지를 묻는 데서 시작한다. 세 번째 갈래는 기존 파이프라인의 위의 개선이다. 여기에서 scFoundation은 자신의 embedding을 이미 쓰이고 있는 분석 파이프라인에 끼워 넣고 그 파이프라인이 원래보다 나아지는 것으로 신뢰를 얻는다. 인용 구조에서 이 방식은 여덟 축 어디에도 깔끔히 들어가지 않는 인용을 만들어 냈다. 끼워 넣는 대상이 되는 논문은 방법 파트너이면서 동시에 이겨야 할 baseline이고 두 역할이 한 인용에 겹친다. 네 번째 사례에서 축이 하나 늘어난 셈인데, 사례를 만나 분류가 자라는 것은 방법이 살아 있다는 표시다.

신뢰 전략과 나란히 놓이는 두 번째 비교축은 자료 출처를 안에 적는가 밖으로 내보내는가다. Geneformer는 Methods에 110건 남짓의 출처 인용을 달아 corpus 조립을 본문 안에 기록했고 AlphaGenome은 같은 종류의 기록을 Supplementary 목록으로 통째로 넘겼다. 참고문헌의 길이만 비교하면 앞의 논문이 훨씬 넓게 읽은 것처럼 보이지만 같은 축으로 재면 차이는 출처를 어디에 적었는가에 있다. 개별 데이터셋 대신 집계 repository를 인용해 꼬리를 압축한 scFoundation은 둘 사이에 있다. 세 논문을 나란히 놓으면 자료 출처를 참고문헌에 남기는 관행이 어느 방향으로 움직이고 있는지가 보인다. 세 번째 비교축은 계보의 폭이다. NTV3에서는 자기 인용이 계보를 지배했는데 한 연구집단이 이전에 내놓은 여러 유전체 모델을 하나의 backbone으로 통합하는 작업이었기 때문이다. Method lineage 축의 상당 부분이 같은 집단의 이전 논문으로 채워지는 모습은 여러 전통을 봉합하는 논문과 형태가 확연히 다르다. 자기 인용이 많다는 사실은 그 논문이 무엇을 하는 작업인지를 말해 줄 뿐 좋고 나쁨을 판정하지 않는다. 통합형 논문은 외부 전통과의 접점이 적으므로 그 모델을 다른 맥락에 옮길 때 무엇이 함께 따라와야 하는지를 별도로 확인해야 한다. 세 축의 비교가 가능했던 것은 여덟 개의 축이 다섯 편 내내 고정되어 있었기 때문이다. 논문마다 그때그때 인상적인 것을 적었다면 다섯 개의 인상이 생기고 서로 비교되지 않았을 것이다. 9장에서 여러 논문을 나란히 놓고 종합할 수 있었던 조건과 같다. 여섯 번째 논문을 같은 축으로 재면 세 갈래 중 하나에 들어가거나 네 번째 갈래를 만들 텐데, 어느 쪽이든 앞의 다섯 편에 대한 이해가 함께 갱신된다.

인용 수와 기능 분류는 서로 다른 것을 센다. 인용 수는 그 논문이 몇 번 불렸는지를 세고 기능 분류는 그 논문이 다른 논문 안에서 무슨 일을 하고 있는지를 센다. 한 논문이 벤치마킹 대상으로 100번 불린 것과 해석의 틀로 100번 불린 것은 같은 수로 나타나지만 다른 사실이다. 자료 출처로 불린 것까지 같은 수에 섞이면 그 차이는 더 흐려진다. 앞 절에서 본 것처럼 자기 인용이 계보를 지배하는 논문도 인용 수만으로는 외부 전통과 넓게 이어진 논문과 구분되지 않는다. 이 축들은 같은 목록을 다른 질문으로 다시 세는 방법을 제공한다. 두 수를 함께 보면 자주 불린 논문이 어느 기능으로 불렸는지까지 알 수 있고 그 조합이 인용 수 하나보다 판단에 쓸모가 있다.

11장. 위키에서 연구 산출물로

위키는 근거와 관계를 유지하고 무엇을 주장할지와 그 주장의 강도는 연구자가 정한다. 두 역할을 섞으면 위키가 논문을 만들어 준다는 기대가 생기고 그 기대는 반드시 배신당한다. 대학원 세미나에서 한 시간짜리 발표를 맡은 사람이 있다고 하자. 위키에는 관련 논문 정리가 마흔 편 들어 있고 종합 문서도 몇 개 만들어 두었다. 여기서 “이 주제로 발표 자료를 만들어 달라”고 요청하면 마흔 편의 요약이 순서대로 이어진 문서가 나온다. 그 문서는 사실관계가 대체로 맞고 형식도

깔끔하지만 청중이 한 시간 뒤에 무엇을 다르게 알게 되는지가 어디에도 없다. 발표는 그 하나를 먼저 정하고 그것을 위해 마흔 편 중 여섯 편을 고르고 나머지 서른네 편을 말하지 않기로 결정하는 일이다. 선택이 빠진 산출물은 위키의 사본이고 사본은 원본보다 항상 나쁘다.

위키에서 만들 수 있는 산출물

위키에서 나오는 산출물은 성격에 따라 네 갈래로 묶인다. 다음에 무엇을 할지 정하는 것, 어떤 주장을 어떻게 조립할지 정하는 것, 다른 사람을 움직이는 것, 그리고 이해를 옮기는 것이다. 갈래가 다르면 필요한 근거의 종류도 다르고 실패하는 방식도 다르다. 아래 표의 가운데 열이 실제로 산출물의 형태를 정한다. 도와야 할 결정이 무엇인지 답할 수 없다면 그 산출물은 아직 만들 준비가 되지 않은 것이다. 오른쪽 열은 산출물이 어디에서 출발하는지를 보여 준다. 대부분의 산출물은 개별 논문 페이지 위 위에 놓인 종합층에서 출발한다. 이 책이 근거로 삼은 연구용 위키에는 wiki page가 17,591건 있고 그 위의 종합층은 overview 590건, concept 408건, question 550건이다.

산출물	도와야 하는 결정	주로 끌어오는 층
연구 아젠다와 질문 지도	다음에 무엇을 할 것인가	question, overview
리뷰와 원고 개요	이번에 무엇을 주장할 것인가	overview, concept
근거 표와 주장 지도	각 주장이 무엇에 기대는가	paper page, source note
분석 인계 문서와 결정 메모	누가 무엇을 어디까지 하는가	concept, agenda
강의 노트와 슬라이드	청중이 무엇을 다르게 알게 되는가	overview, concept
인터랙티브와 해설	관계를 어떤 형태로 보게 할 것인가	overview, paper page
학생 학습 자료와 논문 읽기 안내	무엇을 어떤 순서로 읽는가	question, paper page

17,591건을 훑어 무엇을 쓸지 고르는 일은 사람도 에이전트도 하지 못한다. 실제로 작동하는 경로는 590건의 overview 중 지금 질문에 해당하는 두세 건을 고르고 거기에서 아래로 내려가 필요한 논문만 다시 읽는 것이다. 그래서 종합층이 얇으면 산출물 제작이 곧바로 막힌다. 논문 페이지만 많고 overview가 없는 위키에서는 산출물을 만들려 할 때마다 처음부터 종합을 해야 하고 그 종합은 산출물이 끝나면 함께 사라진다. 4장과 9장이 종합 문서 갱신을 ingest와 질의의 완료 조건으로 삼은 이유가 여기서 드러난다. 종합층은 그 자체로는 아무에게도 보이지 않는 중간 산물처럼 보이지만 산출물을 만드는 시점이 되면 그 층이 있는지 없는지가 전부를 가른다. 위키를 만들 때 들인 노력의 값은 산출물을 만들 때 지불된다. 같은 근거가 독자에 따라 다른 산출물이 된다는 점도 표가 보여 준다. 어떤 방법의 한계를 정리해 둔 concept 문서 하나는 연구 아젠다에서는 다음 분석을 제한하는 조건으로 쓰이고 강의에서는 학생이 그 방법을 처음 쓸 때 조심할 지점으로 쓰이며 원고에서는 고찰 문단의 한 문장으로 쓰인다.

세 산출물이 같은 문서를 가리키지만 꺼내 오는 부분과 말하는 강도가 서로 다르다. 한 번 정리해 둔 것이 여러 작업에서 다시 쓰인다는 1장의 이익은 이 지점에서 회수된다. 각 산출물이 자기 판본을 따로 만들기 시작하면 반년 뒤에는 어느 것이 현재의 판단인지 알 수 없게 되고 그때는 네 개의 비슷한 문서를 놓고 무엇이 맞는지 다시 정해야 한다. 산출물을 만들다 보면 위키의 어느 부분이 실제로 쓸 만하지도 드러난다. 종합 문서에 연결되지 않은 고립 페이지의 수를 세는 감사 스크립트가 카테고리별 연결 비율을 집계하지만 그 감사가 잡는 것은 연결의 유무까지이고 연결의 강도는 잡지 못한다. 링크가 걸려 있는데도 막상 주장을 세우려 하면 그 링크가 아무것도 지탱하지 못하는 경우가 있다. 종합 문서에 논문 제목만 나열되어 있고 그 논문이 기존 지식을 어떻게 바꾸는지에 대한 문장이 없을 때 그런 일이 생긴다. 산출물 제작은 기계 감사가 잡지 못하는 이 종류의 빈틈을 드러내는 가장 확실한 방법이다.

질문에서 연구 아젠다로

연구 아젠다는 근거와 긴장, 공백, 가능한 분석, 그리고 언제 멈출지가 하나로 이어진 실행 문서다. 아젠다가 주제 목록으로 끝나는 이유는 대체로 중간 단계를 건너뛰었기 때문이다. 흥미롭다는 감각에서 곧바로 할 일 항목으로 넘어가면 그 항목이 왜 지금 필요한지가 문서 안에 남지 않고 몇 주 뒤에 자신이 다시 읽어도 무엇을 하려던 것인지 복원되지 않는다. 위키에서 아젠다로 가는 길은 다음 순서를 지날 때 실행 가능한 문서가 된다.

↓

↓

↓

↓

↓

↓ , ,
↓

첫 단계는 이미 반복된 질문을 찾는 것이다. 9장의 질의 기록과 10장의 구조 분석을 지나면 같은 질문이 다른 표현으로 여러 번 올라와 있다. 이 책이 파트 1 내내 따라온 여섯 편에서는 그 질문이 평가에 관한 것이었다. 다섯 편의 모델 논문이 모두 자기 모델이 쓸 만하다고 말하는데 쓸 만하다는 말을 뒷받침하는 방식이 서로 다르다. 어떤 논문은 이미 확립된 외부 사실과 예측이 맞아떨어진다는 것으로 신뢰를 만들고 어떤 논문은 같은 과제에서 다른 방법과 겨루어 이겼다는 것으로 만들며 또 어떤 논문은 기존 분석 파이프라인에 끼워 넣었을 때 결과가 나아졌다는 것으로 만든다. 반복된 질문을 찾는 일은 기억에 의존하지 않는 편이 낫다. 질의를 던질 때마다 그 질문을 question 페이지나 별도의 질문 목록에 한 줄로 남겨 두면, 몇 달치가 쌓였을 때 어떤 질문이 여러 번 다른 표현으로 올라왔는지 눈에 보인다. 질문 지도는 그렇게 모인 질문을 주제별로 묶고 각 질문 옆에 지금 답할 수 있는지 여부를 적어 둔 문서다. 답할 수 있는 질문은 종합 문서로 가고 답할 수 없으면서 반복되는 질문이 아젠다의 후보가 된다. 답을 얻은 질문도 지우지 않고 답과 그 답의 경계를 붙여 남기는데 같은 질문이 반년 뒤에 다시 올라왔을 때 이미 정리된 것인지 아닌지를 그 기록으로 판단하기 때문이다.

두 번째 단계는 지금 위키가 지지하는 답과 그 답의 경계를 적는 것이다. 경계를 적는 일이 답을 적는 일보다 중요하다. 위 사례에서 위키가 지지할 수 있는 답은 세 가지 신뢰 전략이 존재한다는 사실까지이고 어느 전략이 더 믿을 만한지는 지지하지 못한다. 세 전략이 서로 다른 평가 자료와 다른 성공 기준을 쓰기 때문이다. 여기서 긴장이 드러난다. 같은 분야의 논문들이 같은 축에서 비교된 적이 없다는 것, 그러면서도 서로를 인용해 계보를 만들고 있다는 것이 긴장이다. 세 번째부터가 아젠다의 본체다. 긴장을 검증 가능한 질문으로 바꾸려면 무엇이 참일 때와 거짓일 때 관찰이 달라지는지를 적어야 한다. “외부 사실 일치로 검증된 모델이 방법 대 방법 벤치마킹에서도 같은 순위를 유지하는가”는 검증할 수 있고 “어느 모델이 더 나은가”는 참과 거짓을 가를 관찰을 지정하지 못한다. 앞의 질문은 필요한 자료와 분석을 스스로 지정한다. 두 종류의 평가를 같은 모델 집합에 적용하고 순위를 비교하면 되기 때문이다. 질문이 자기가 필요한 분석을 지정하지 못하면 아직 충분히 좁혀지지 않은 것이다.

1. 최근에 던졌던 질문 기록을 훑어 두 번 이상 다른 표현으로 등장한 질문을 하나 고른다
2. 그 질문에 대해 지금 위키가 지지하는 답을 두세 문장으로 쓰고 답이 성립하지 않는 조건을 한 문장으로 쓴다
3. 관련 논문 여덟에서 열두 편을 훑어 서로 어긋나거나 비어 있는 지점을 하나 찾고 어긋남이 진짜 모순인지 조건 차이인지 구분해 적는다
4. 그 지점을 검증 가능한 질문 하나로 좁히고 참일 때와 거짓일 때 관찰이 어떻게 달라지는지 함께 적는다
5. 필요한 자료와 분석, 그 분석의 판별력, 예상과 다른 결과가 나왔을 때의 해석, 손에 남은 산출물을 적는다
6. 마지막 줄에 중단 기준을 적는다. 무엇이 확인되면 접는지, 언제까지 무엇이 없으면 그만두는지를 쓴다

이 문서를 새 세션의 에이전트에게 주었을 때 다음에 무엇을 해야 하는지와 그 결과를 어떻게 읽어야 하는지가 위키의 어느 문서를 근거로 설명되어야 한다. 아젠다 문서에 들어가는 항목은 일곱 가지다. 무엇을 위한 아젠다인지를 적는 목적, 지금 위키가 지지하는 답과 경계를 적는 현재 근거, 좁혀진 연구 질문, 그 질문에 답하기 위해 실제로 돌려야 하는 분석, 그 분석이 어떤 결과를 구별해 낼 수 있는지를 적는 판별력, 예상과 다른 결과가 나왔을 때 그것을 어떻게 읽을지 미리 적는 실패 해석, 그리고 이 아젠다가 끝났을 때 손에 남은 구체적인 산출물이다. 일곱 항목 중 가장 자주 빠지는 것이 판별력과 실패 해석이다. 어떤 결과가 나오든 사후에 설명할 수 있는 분석은 판별력이 없고 판별력이 없는 분석은 시간만 쓰고 아무것도 정하지 못한다. 실패 해석을 미리 적어 두면 결과가 나온 뒤에 해석을 만들어 내는 일도 줄어든다. 중단 기준을 미리 적는 것이 아젠다와 주제 목록을 가르는 지점이다. 어떤 결과가 나오면 이 방향을 접는지, 어느 시점까지 무엇이 확인되지 않으면 그만두는지를 적어 두면 나중에 그 판단을 감정 없이 할 수 있다. 이미 시간을 쓴 뒤에 그만둘지 결정하려 하면 들인 시간이 판단을 흐린다. 흥미로운 주제를 적어 두는 목록과 실행을 정하는 아젠다는 다른 파일에 두는데 판별력과 중단 기준이 없는 항목은 실행되지 않은 채 목록에만 남기 때문이다. 아젠다는 미공개 실행 계획이므로 작업 폴더의 agenda 층에 두고 공개된 논문 지식을 담은 위키에는 넣지 않는다. 두 층이 섞이면 위키를 다른 사람과 나눌 때 무엇을 걸러 내야 하는지 매번 다시 판단해야 한다. 아젠다와 위키 사이에는 양방향 링크를 만들어, 아젠다에서 근거로 삼은 종합 문서로 갈 수 있고 그 종합 문서에서도 이 질문이 어떤 아젠다로 이어졌는지 보이게 한다.

위키에서 원고 구조로

원고를 쓰기 시작할 때 가장 흔한 실수는 위키에 있는 내용을 순서대로 옮기는 것이다. 위키가 답하는 질문과 원고가 답하는 질문이 다르기 때문에 그렇게 하면 반드시 실패한다. 위키는 “이 분야에 대해 지금 무엇을 아는가”에 답하도록 만들어져 있고 원고는 “이번 논문이 무엇을 주장하는가”에 답해야 한다. 앞의 질문은 아는 것을 모두 담을수록 좋아지고 뒤의 질문은 주장에 필요하지 않은 것을 덜어낼수록 좋아진다. 두 질문을 분리하지 않으면 원고는 넓고 평평해지고 읽는 사람은 저자가 무엇을 말하려는지 끝까지 알 수 없다. 원고 설계에서 위키의 각 층이 맡는 역할은 다르다. Overview와 concept에서는 현재의 합의와 그 합의가 성립하는 범위를 뽑아 온다. 도입부에서 무엇을 전제로 깔고 시작할 수 있는지가 여기서 정해진다. Question 페이지에서는 아직 풀리지 않은 긴장을 뽑아 온다. 원고가 왜 지금 필요한지에 대한 답이 대체로 여기에 있다. 논문 페이지와 source note에서는 각 문단을 지탱할 근거와 수치를 확인한다.

리뷰 개요와 원고 개요는 위키에서 꺼내 오는 것이 다르다. 리뷰는 분야의 현재 상태를 정리하는 글이므로 overview 층에서 가장 많은 것을 가져오고 기여는 기존 결과들을 어떤 배치로 놓았는지에서 나온다. 원고는 새 결과가 중심이므로 위키에서 그 결과의 배경과 해석할 때 비교할 선행 연구를 가져온다. 리뷰를 쓸 때 위키를 그대로 옮기고 싶은 유혹이 더 크고 그래서 실패도 더 자주 일어난다. 리뷰의 기여는 무엇을 어떤 축으로 묶었는지에 있으며 축을 정하는 일은 위키가 대신해 주지 않는다. 원고의 형태를 정하는 것은 아젠다 문서 쪽이다. 이 원고를 누가 읽는지, 중심 주장이 하나로 말해 무엇인지, 문단이 어떤 순서로 놓여야 하는지, 그리고 지금 없어서 추가로 돌려야 하는 분석과 새로 만들어야 하는 그림이 무엇인지를 아젠다에서 가져온다. 이 네 가지가 정해지기 전에 문장을 쓰기 시작하면 나중에 대부분을 버리게 된다. 원고 개요는 위키의 근거와 아젠다의 방향을 합쳐 만든다. 어느 한쪽만으로는 개요가 서지 않는다.

문장을 쓰는 단계에서는 다시 아래로 내려간다. 원고에 들어갈 주장은 종합 문서의 문장을 그대로 옮겨 쓰지 않고 그 문장이 기대고 있는 논문 페이지와 source note를 열어 확인한다. 수치와 조건이 결론을 좌우하는 대목에서는 원문 PDF까지 내려간다. Source note는 원문으로 가는 경로일 뿐이라는 4장의 원칙이 여기서 다시 적용된다. 종합 문서는 여러 논문을 압축하는 과정에서 조건을 떨어뜨리기 마련이고 떨어진 조건이 원고에서는 결정적일 수 있다. 위키가 원고 설계에서 가장 잘하는 일은 없는 것을 알려 주는 것이다. 문단마다 그 문단을 지탱할 근거를 위키에서 찾아 붙이다 보면 근거가 붙지 않는 문단이 반드시 나온다. 그 문단은 지금 자료로는 쓸 수 없는 문단이거나 아직 돌리지 않은 분석이 필요한 문단이거나 애초에 필요하지 않은 문단이다. 셋 중 어느 것인지 판정하고 나면 원고를 쓰기 전에 해야 할 일의 목록이 만들어진다. 다 쓴 뒤에 근거가 없다는 것을 발견하면 문단을 통째로 버리게 되고 그때는 그 문단을 쓰는 데 든 시간이 이미 사라진 뒤다. 주장의 강도를 정하는 일은 위키가 대신할 수 없는 몫이다. 위키는 어떤 문장에 대해 그것을 지지하는 논문이 세 편이고 그중 두 편은 특정 조건이 다르다는 것까지 말해 줄 수 있다. 거기서 “그러므로 일반적으로 성립한다”고 쓰지, “특정 조건에서 관찰되었다”고 쓰지, 아니면 “아직 결론을 내리기 어렵다”고 쓰지는 사람이 정한다. 같은 근거 위에서도 세 문장은 모두 쓸 수 있고 셋 중 어느 것을 고르느냐가 연구자의 판단이다. 근거 표와 주장 지도는 이 판단을 눈에 보이게 만드는 산출물이다.

인터랙티브 만들기

움직이는 화면을 만들기 전에 물어야 할 것은 이 관계가 정지된 문장으로 전달되지 않는지 여부다. 인터랙티브가 실제로 도움이 되는 경우는 네 가지로 좁혀진다. 여러 대상을 같은 축 위에서 비교해야 하는데 축이 여러 개일 때, 하나의 자료나 결정이 여러 갈래의 후속 작업에 영향을 줄 때, 시간에 따라 상태가 바뀌어 한 시점의 그림으로는 변화를 볼 수 없을 때, 그리고 계층이나 의존 관계나 되먹임 구조가 있어 선형 문장으로 따라가기 어려울 때다. 네 조건 중 어디에도 해당하지 않으면 표 하나나 문단 하나가 더 낫다. 해당 여부는 그 관계를 말로 설명해 보고 어디서 막히는지를 보면 판단할 수 있다. 제작 순서는 질문에서 시작해 위키로 되돌아오는 여섯 단계다. 먼저 이 화면이 답해야 할 질문을 한 문장으로 적고 그 질문에 필요한 근거만 고른다. 다음으로 화면에 올라갈 각 값이 어느 문서의 어느 항목에서 왔고 어떤 변환을 거쳤는지를 명세로 남긴다. 그 다음에 사용자가 무엇을 조작할 수 있는지를 설계하고 조작 결과가 근거와 어긋나지 않는지 검증하고 마지막으로 위키와 아젠다에서 이 화면을 가리키는 링크를 만든다. 여섯 단계 중 세 번째가 가장 자주 생략되고 생략되면 몇 달 뒤에 화면의 숫자가 어디서 왔는지 아무도 답하지 못한다.

출처 명세가 필요한 이유는 사용자가 화면을 조작하기 때문이다. 축을 바꾸고 필터를 걸고 범위를 좁히는 동안에도 각 값이 무엇을 뜻하는지가 변하지 않아야 하는데 값의 출처가 기록되어 있지 않으면 조작할수록 의미가 흐려진다. 특히 위험한 것은 조합이 존재하지 않는 칸과 값이 0인 칸을 구분하지 않는 화면이다. 어떤 모델이 어떤 평가를 받은 적이 없어서 비어 있는 칸과, 평가를 받았는데 점수가 낮은 칸은 완전히 다른 사실인데 둘 다 빈칸이나 0으로 보이면 “이 조건에서는 효과가 없다”는 잘못된 읽기가 만들어진다. 없음과 낮음을 구분하는 표시를 넣는 일이 화면을 예쁘게 만드는 일보다 우선한다. 검증 단계에서 확인할 것은 세 가지다. 화면에 나온 값이 원자료의 값과 같은지, 조작할 수 있는 모든 조합에서 화면이 사실과 어긋나는 상태를 만들지 않는지, 그리고 화면만 보고 내릴 수 있는 결론이 근거가 지지하는 범위를 넘지 않는지다. 세 번째가 가장 어렵고 가장 자주 실패한다. 비교 화면은 비교 가능하지 않은 것들도 나란히 놓이는 순간 비교 가능해 보이게

만들기 때문이다. 서로 다른 평가 자료로 얻은 점수를 같은 막대그래프에 놓으면 그 그래프는 존재하지 않는 순위를 만들어 낸다.

인터랙티브와 함께 자주 만들어지는 것이 해설과 논문 읽기 안내다. 셋은 같은 재료를 쓰지만 독자가 다르다. 인터랙티브는 이미 대상을 아는 사람이 관계를 탐색하도록 돕고 해설은 처음 접하는 사람이 무엇이 왜 문제인지 이해하도록 돕고 읽기 안내는 앞으로 원문을 읽을 사람에게 순서와 주의할 지점을 준다. 세 가지를 한 문서에 섞으면 아는 사람에게는 길고 모르는 사람에게는 불친절한 문서가 나온다. 재료가 같다는 이유로 합치지 않고 위키의 같은 문서를 셋이 함께 가리키게 하는 편이 낫다. 인터랙티브를 만들지 않기로 결정하는 것도 결과다. 만드는 데 드는 시간보다 고치는 데 드는 시간이 크고 근거가 갱신되면 화면도 함께 갱신해야 한다. 한 번 발표하고 버릴 자료라면 정적인 그림이 낫고 여러 사람이 반복해서 쓸 자료이거나 자기 자신이 반년 뒤에 다시 열어 볼 자료일 때 인터랙티브가 값을 한다. 시각적으로 인상적이라는 이유만으로 만든 화면은 대체로 한 번 보이고 잊힌다. 만들지 않기로 했다면 그 판단도 아젠다에 한 줄로 남겨 두는데 그렇지 않으면 반년 뒤에 같은 고민을 처음부터 다시 하게 된다.

산출물과 정보의 관계

산출물은 위키에서 나오지만 위키를 대신하지 않는다. 슬라이드의 한 문장이 위키의 종합 문서보다 최신인 상태가 생기면 그때부터 모든 작업이 어긋난다. 다음 사람이, 혹은 반년 뒤의 자신이 근거를 확인하려고 위키를 열었을 때 거기에는 옛 판단이 남아 있고 새 판단은 어느 발표 파일 안에 들어 있기 때문이다. 산출물은 특정 시점에 특정 독자를 향해 만든 것이므로 시간이 지나면 낡는 것이 정상이고 정보는 낡으면 고쳐지는 것이 정상이다. 두 성격을 구분하지 않으면 위키가 정보의 지위를 잃는다. 새 해석은 산출물을 만드는 과정에서 자주 생긴다. 근거를 나란히 놓고 순서를 정하는 동안 이전에는 보이지 않던 관계가 보이기 때문이다. 그중에서 위키로 되돌릴 것은 다시 쓸 수 있는 비교와 판단뿐이다. 이번 발표에서만 의미 있는 순서 배치나 이 원고에서만 쓰는 강조는 되돌리지 않는다. 저장 여부를 가르는 조건은 9장이 정한 그대로다. 여러 문서를 비교해 다시 쓸 수 있는 판단이 만들어졌는가를 묻고 그렇다면 종합 문서를 실제로 고치고 그렇지 않다면 산출물 안에 두고 끝낸다.

공개된 지식과 미공개 작업을 나누는 일은 처음부터 지켜야 한다. 논문에서 나온 지식은 위키가 소유하고 실행 계획과 미공개 결과와 원고 초안은 아젠다나 해당 작업 폴더가 소유한다. 저자 순서, 제출할 저널, 경쟁 그룹의 동향, 심사 대응 전략처럼 프로젝트에 속한 판단도 위키에 넣지 않는다. 나누는 이유는 위키가 공유되거나 외부에 노출될 수 있기 때문이고 더 실질적으로는 한번 섞인 총을 나중에 분리하는 일이 거의 불가능하기 때문이다. 문서 수백 개에서 어느 문장이 공개 가능한 지식이고 어느 문장이 미공개 판단인지 사후에 가려내는 작업은 처음부터 나누는 비용보다 훨씬 크다. 분석 인계 문서와 결정 메모는 소유가 특히 헛갈리는 산출물이다. 두 문서 모두 근거는 위키에서 오지만 내용의 대부분은 누가 무엇을 어디까지 하는지와 그렇게 정한 이유이므로 작업 폴더가 소유한다. 인계 문서에서 위키로 돌아갈 것은 그 분석을 하기로 한 판단이 문헌의 어느 근거에 기대고 있는지 한 줄뿐이고 나머지는 작업 폴더에 남는다. 결정 메모도 마찬가지로 결정 자체는 프로젝트의 것이고 결정을 뒷받침한 문헌 근거만 종합 문서에 반영된다. 두 종류를 섞어 위키에 넣으면 몇 달 뒤 위키가 프로젝트 관리 도구처럼 보이기 시작하고 다른 프로젝트에서 그 위키를 다시 쓰기 어려워진다.

외부로 나가는 산출물에서 건어내야 하는 것도 있다. 로컬 파일 경로, 내부 지침 문서의 문장, 작업 과정에서 에이전트가 남긴 진행 상황 서술이 그런 것들이다. 위키 안에서는 유용한 정보지만 슬라이드나 원고나 공유 문서에 들어가면 읽는 사람에게는 잡음이고 때로는 노출되면 곤란한 정보다. 산출물을 내보내기 전에 한 번 훑는 절차를 두면 대부분 걸러진다. 걸러야 할 항목이 매번 비슷하므로 세 번쯤 반복되면 운영 규칙으로 적어 둘 만하다. 되돌리지 않기로 결정하는 판단도 명시적으로 해야 한다. 에이전트와 나는 모든 대화와 만들어진 모든 일회성 산출물을 위키에 저장하면 위키는 빠르게 쓸모를 잃는다. 문서 수가 늘면 검색 후보가 늘고 후보 중 실제로 쓸 만한 것의 비율이 떨어지고 결국 위키를 여는 일이 부담이 된다. 앞서 본 감사 도구가 세는 고립 페이지 중에는 애초에 만들 이유가 없었는데 만들어진 문서가 섞여 있다. 무엇을 남기지 않을지 정하는 일이 무엇을 남길지 정하는 일과 같은 비중을 가진다.

세 가지 미니 프로젝트

산출물 제작을 익히는 데는 세 가지 형태가 서로 다른 근육을 쓴다. 세 가지 모두 같은 골격을 지나지만 어디에서 막히는지가 다르다. 공통 골격은 질문을 정하고 근거를 고르고 출처를 기록하고 검증하고 위키로 되돌리는 다섯 단계다. 셋을 한꺼번에 하려 하지 말고 하나씩 끝까지 만들어 보는 편이 낫다. 한 형태를 끝까지 만들어 보면 나머지 둘에서 어느 단계가 어려울지 미리 보이기 때문이다. 첫째는 한 장짜리 연구 아젠다다. 관련 논문 여덟에서 열두 편을 놓고 그 사이의 빈칸을 찾아 앞 절의 일곱 항목으로 정리한다. 여덟 편은 빈칸이 보이기 시작하는 최소 규모이고 열두 편이 넘으면 한 장에 담기지 않는다. 이 프로젝트가 훈련하는 것은 버리는 일이다. 열두 편 중 아젠다에 실제로 등장하는 논문은 서넛이면 충분하고 나머지는 그 서넛의 위치를 정하는 배경으로만 쓰인다.

에이전트에게 맡길 수 있는 일의 경계는 세 프로젝트에서 모두 같다. 후보 논문을 모으고 근거를 표로 정렬하고 문단 순서의 대안을 여러 개 만들고 화면에 들어갈 값의 출처 명세를 채우는 일은 에이전트가 빠르고 성실하게 한다. 어떤 긴장을 연구 질문으로 삼을지, 중심 주장을 어느 강도로 말할지, 어떤 결과가 나오면 그만둘지는 사람이 정한다. 두 종류를 구분하지 않고 산출물 전체를 맡기면 형식은 갖추었지만 아무것도 결정하지 않는 문서가 나온다. 결정을 담지 않은 산출물은 읽는 사람에게 판단을 그대로 넘기고 그러면 산출물을 만든 의미가 없어진다. 세 프로젝트의 실패 신호는 각각 다르다. 아젠다에 중단 기준이 없으면 그 문서는 주제 목록이다. 원고 지도에서 없는 근거를 적는 칸이 비어 있으면 그 문서는 요약이다. 인터랙티브에 출처 기록이 없으면 그 화면은 그림이다. 세 신호는 만든 뒤에 한 번 훑으면 바로 보이므로 완성했다고 판단하기 전에 자기 산출물에서 세 가지를 확인한다.

```

ingest
↓
↓
↓
↓
,      ,
      ↗

```

12장. Obsidian으로 보기와 채팅으로 묻기

74

폴더를 여는 도구

Obsidian은 Markdown 파일이 든 폴더를 하나 골라 그 안을 탐색하고 편집하는 프로그램이다. 파일을 자기 형식으로 가져가지 않고 폴더에 있는 그대로 읽는다는 점이 이 도구를 쓰는 이유다. 위키가 만들어 둔 폴더를 그대로 지정하면 되고 그 안의 파일은 여전히 일반 텍스트로 남는다. 도구를 지우더라도 문서는 그곳에 있고 다른 프로그램으로 열어도 같은 내용이 보인다. 설치해 내려받기 페이지에서 자기 운영체제에 맞는 파일을 받으면 된다. 계정 없이 쓸 수 있고 처음 열면 어느 폴더를 열 것인지 묻는다. 여기서 새로 만들지 말고 이미 있는 폴더를 여는 쪽을 고른다. 위키의 지식 총 폴더를 여는 것이 기준인데, 3장에서 본 위키링크 규약이 그 폴더를 뿌리로 삼아 적혀 있기 때문이다. 저장소 최상위를 열면 링크가 전부 어긋난다.

열고 나면 왼쪽에 폴더와 파일 목록이 뜨고 문서를 누르면 본문이 보인다. 링크가 걸린 자리를 누르면 그 문서로 이동하고 문서 아래쪽에는 이 문서를 가리키는 다른 문서 목록이 나온다. 마지막 것이 특히 쓸모 있다. 어떤 개념 문서를 열었을 때 그 개념을 인용하는 논문 페이지가 몇 개이고 무엇인지가 바로 보이므로 그 개념이 실제로 쓰이고 있는지를 눈으로 확인할 수 있다. 아무 데서도 가리키지 않는 문서는 이 목록이 비어 있고 그것이 4장에서 말한 고립된 페이지다. 링크가 걸려 있지만 대상 문서가 없는 경우도 표시가 다르게 나타난다. 검사 스크립트로 잡는 것과 눈으로 보는 것 가운데 눈이 빠른 경우가 있고 특히 문서를 만들다가 오타를 낸 링크는 이 화면에서 곧바로 걸린다.

그래프가 보여 주는 것

이 도구에서 가장 눈에 띄는 화면은 그래프다. 문서를 점으로, 링크를 선으로 그려 폴더 전체를 한 장에 놓는다. 이 화면이 실제로 쓸모 있어지는 것은 문서가 어느 정도 쌓인 뒤다. 열 개짜리 폴더에서는 점 열 개가 흩어져 있을 뿐이고 몇백 개가 되면 어느 쪽이 촘촘하고 어느 쪽이 성긴지가 보이기 시작한다. 다음은 논문 열 편 남짓을 넣은 위키를 그래프로 연 화면이다.

Obsidian 그래프 뷰에서 본 LLM-Wiki 폴더

전체 그래프에서는 논문 페이지와 개념 페이지의 연결 밀도를 본다. 외곽에 홀로 남은 문서는 종합층 연결이 빠졌을 가능성이 크다.

왼쪽에 폴더 구조가 보이고 가운데가 그래프다. 점 하나가 문서 하나이고 논문 페이지와 개념 페이지가 같은 평면에 놓인다. 오른쪽 설정에서 경로로 범위를 좁힐 수 있어서 지식 총만 보거나 특정 카테고리만 볼 수 있다. 화면에서는 방법을 다루는 하위 폴더에 색을 따로 주었다. 색을 나누면 자기 위키가 어느 축으로 자라고 있는지가 한눈에 드러난다. 논문 페이지지만 늘고 개념 페이지가 거의 없으면 정리는 하고 있는데 종합은 하지 않는 상태이고 그 상태는 4장에서 이름 붙인 그것이다. 점의 크기는 그 문서를 가리키는 링크 수에 따라 달라지므로 큰 점이 여러 논문이 함께 기대고 있는 개념이다. 화면 가운데를 보면 논문 파일명으로 된 점들과 개념 이름으로 된 점들이 섞여 있는데 개념 쪽 점이 대체로 크다. 여러 논문이 같은 개념을 가리키고 있다는 뜻이고 그것이 종합 층이 하는 일이다. 반대로 논문 점끼리만 이어져 있고 개념 점이 작으면 논문을 서로 링크하기만 하고 개념으로 올리지는 않은 상태다.

한 점을 고르면 그 문서와 이어진 것만 강조되고 나머지는 흐려진다. 이 기능이 그래프에서 가장 자주 쓰게 되는 것인데, 전체 그림보다 한 문서의 이웃을 보는 일이 실제로 더 많기 때문이다.

개념 하나를 골랐을 때 이어지는 논문들

개념 하나를 선택하면 그 개념을 뒷받침하거나 사용하는 논문만 강조된다. 연결된 논문의 수와 범위를 보고 해당 개념이 위키의 축으로 자랐는지 확인한다.

가운데 강조된 점이 개념 하나이고 거기서 뿔어나간 선이 그 개념을 인용하는 논문들이다. 선이 닿은 점들의 이름을 읽으면 이 개념이 어느 연도의 어느 논문들에 걸쳐 있는지가 그대로 보인다. 이 화면이 알려 주는 것은 그 개념이 몇 편의 논문에 걸쳐 있는가다. 한두 편에만 걸쳐 있다면 아직 개념으로 승격할 때가 아니었을 수 있고 열 편이 넘게 걸쳐 있다면 그 개념이 이 위키의 축 가운데 하나다. 새 논문을 넣은 뒤에 이 화면을 열어 그 논문이 어디에 붙었는지 보는 것도 확인 방법이 된다. 아무 데도 붙지 않은 채 외곽에 떠 있으면 종합 층 연결을 빠뜨린 것이다.

화면에서 눈에 걸리는 것들

그래프 말고도 눈으로 확인하기 좋은 것이 몇 가지 있다. 첫째는 문서의 머리말이다. 논문 페이지마다 어느 원문에서 왔고 어느 카테고리에 속하며 어떤 도구로 추출했는지 적혀 있는데 여러 문서를 빠르게 넘기며 이 부분만 훑으면 형식이

어긋난 문서가 눈에 띈다. 스크립트로도 잡히지만 형식이 어긋난 이유까지는 알려 주지 않고 화면으로 보면 왜 그렇게 되었는지가 대체로 함께 보인다. 둘째는 검색 결과다. 이 도구의 검색은 위키의 검색과 성격이 다르다. 문자열을 그대로 찾는 것이라 답을 만들어 주지는 않지만 어떤 표현이 몇 개 문서에 몇 번 나오는지 세어 준다. 같은 개념을 서로 다른 표현으로 적어 둔 자리를 찾을 때 이 방식이 빠르다.

셋째는 폴더 배치다. 왼쪽 목록을 펼쳐 두면 카테고리마다 문서가 몇 개인지가 대략 보이고 이상하게 큰 카테고리나 비어 있는 카테고리가 드러난다. 3장에서 카테고리를 다섯에서 열 개로 시작하라고 한 뒤에 실제로 어떻게 되었는지를 확인한다. 넷째는 만들다 만 문서다. 링크는 걸었는데 대상 문서를 만들지 않은 경우가 표시로 구분되므로 그 목록이 다음에 채울 것의 목록이 된다. 네 가지 모두 검사 스크립트로 대신할 수 있지만 스크립트를 만들기 전에는 화면으로 확인한다.

다섯째는 문서를 나란히 놓고 보는 일이다. 두 문서를 좌우로 열어 두면 같은 논문을 다룬 정리 문서와 논문 페이지가 서로 무엇을 다르게 적고 있는지가 보인다. 4장에서 두 층을 나눈 이유가 여기서 눈으로 확인된다. 정리 문서는 원문의 세부를 담고 논문 페이지는 다른 논문과의 관계를 담는데 둘이 같은 내용을 반복하고 있다면 층을 나눈 값이 사라진 것이다. 여러 논문의 정리 문서를 나란히 놓고 같은 항목끼리 비교하는 작업에도 이 화면이 편하다. 표본 수와 대조군과 판정 기준을 세 편에서 각각 뽑아 비교할 때, 세 창을 띄워 두고 훑는 편이 하나씩 열어 기억하는 것보다 정확하다. 이 작업은 채팅으로도 되지만 결과를 눈으로 확인하고 싶을 때가 있고 그때 이 화면을 쓴다.

그래프가 보여 주지 못하는 것

그래프는 연결의 유무를 보여 주고 연결의 내용은 보여 주지 않는다. 두 문서가 이어져 있다는 것은 알 수 있지만 그 논문이 그 개념을 강화하는지 반박하는지는 선의 모양에 나타나지 않는다. 4장에서 ingest의 최소 조건에 판정을 넣어 둔 이유가 여기 있다. 강화인지 좁힘인지 반박인지 대체인지는 문서의 문장으로 적혀 있어야 하고 그 문장을 읽어야 알 수 있다. 그래서 그래프가 촘촘해 보이는 것과 그 위키가 답을 잘 하는 것은 별개다. 링크만 많이 걸어 두면 그래프는 뻘뻘해지고 답의 품질은 그대로다.

두 번째 한계는 규모다. 문서가 수백 개일 때는 그래프가 지도로 작동하지만 만 개를 넘어가면 점이 뭉쳐 아무것도 읽히지 않는다. 범위를 좁혀야 볼 수 있는데 그 좁히는 조건을 정하려면 이미 무엇을 찾는지 알고 있어야 한다. 무엇을 찾는지 알고 있다면 검색으로 가는 편이 빠르다. 세 번째는 그래프가 답을 만들어 주지 않는다는 점이다. 두 논문이 이어져 있다는 것을 보고 나서 실제로 알고 싶은 것은 두 논문의 결과가 왜 다른가인데, 그 답은 두 문서를 읽어야 나온다.

다른 도구를 고를 때

다른 Markdown 도구들도 같은 일을 하고 무엇을 고르는 위키 쪽에서 달라질 것은 없다. 폴더의 파일을 그대로 읽고 그대로 저장하는 도구라면 무엇이든 된다. 고를 때 확인할 것은 세 가지다. 첫째는 파일을 자기 형식으로 가져가지 않는가다. 가져가는 도구는 그 도구를 지운 뒤에 문서를 열 수 없게 만들고 그러면 위키가 그 도구에 묶인다. 둘째는 위키링크를 이해하는가다. 표준 Markdown의 링크만 지원하는 도구에서는 문서 사이의 연결이 그냥 텍스트로 보인다. 셋째는 폴더 구조를 그대로 보여 주는가다. 자기 방식으로 다시 묶어 보여 주는 도구는 3장에서 정한 폴더 배치를 화면에서 지워 버린다.

3장의 설계 원칙이 표준 Markdown과 위키링크만 쓰고 특정 도구에만 있는 문법을 쓰지 않는다고 못 박아 둔 이유가 여기 있다. 도구 하나에 맞춘 문법을 쓰기 시작하면 그 문법이 문서에 박히고 몇 년 뒤에 도구를 바꿀 때 문서 전체를 손봐야 한다. 위키의 수명이 도구의 수명보다 길다는 전제에서 만들어진 규칙이고 실제로 도구는 몇 년 단위로 바뀐다. 이 원칙을 지키면 도구를 고르는 일이 가벼운 결정이 된다. 마음에 들지 않으면 지우고 다른 것을 열면 되고 문서는 그대로 있다.

채팅 쪽을 주로 쓰는 이유

이 책이 근거로 삼은 위키는 그래프를 여는 것보다 폴더를 지정해 세션을 열고 그 안에서 대화하는 방식으로 운영한다. 키보드로 질문을 치기도 하지만 요즘은 14장에서 다룬 대로 휴대폰 앱에서 마이크로 말하는 비중이 늘었다. 채팅을 하면 답이 나오고 어떤 정보가 확인되는데 그것을 눈으로 읽고 나서 머릿속에서 그 지식을 다시 한번 구성하게 된다. 그 재구성이 지식을 자기 것으로 만드는 과정이다. 그래프를 보는 일에는 그 단계가 없다. 구조가 잘 만들어졌다는 확인은 되지만 그 안의 내용이 머리로 들어오지는 않는다.

말로 물으면 한 단계가 더 붙는다. 질문을 소리 내어 만드는 동안 자기가 아는 것을 자기 입으로 한 번 다시 읽게 되고 그 과정에서 다시 한번 생각하게 된다. 이 반복이 지식을 내재화하고 구조적으로 만드는 데 도움이 된다는 것이 지금까지의

경험이다. 답을 받은 뒤에도 마찬가지다. 받은 내용을 자기 언어로 다시 옮겨 다음 질문을 만들면 그 왕복 자체가 학습이 된다. 20장에서 학생에게 정리된 문서를 건네지 않고 직접 정리하게 하는 이유와 같은 원리이고 서문에서 처리 과정의 출력력을 눈으로 한 번은 읽어야 한다고 적은 이유도 같다. 읽지 않고 파일만 늘리면 남는 것이 없다.

그래서 이 방식에서 위키의 값은 멋진 그래프를 만들어 내는 데 있지 않다. 에이전트를 일종의 동료로 두고 계속 대화하고 이것저것 구상해 보고, 에이전트가 찾아온 내용을 자기 머리로 이해하면서 자기 언어로 해석하고 다시 전달하는 그 과정에 있다. 위키는 그 대화가 매번 처음부터 시작하지 않게 만드는 장치다. 반년 전에 내린 판단이 문서로 남아 있으므로 오늘의 대화가 그 위에서 시작하고 오늘 내린 판단이 다시 문서가 되어 반년 뒤의 대화를 받쳐 준다.

네 번째 단계는 그래프가 시간을 담지 않는다는 점이다. 어느 논문이 먼저 들어왔고 어느 판단이 나중에 뒤집혔는지는 점과 선에 나타나지 않는다. 위키에서 실제로 중요한 것 가운데 하나가 이 변화인데, 새 논문이 기존 주장을 좁혔다면 그 사실은 종합 문서의 문장과 그날의 작업 로그에 있다. 그래프만 보면 지금의 상태만 보이고 어떻게 여기까지 왔는지는 보이지 않는다. 다섯 번째는 무엇이 없는지를 보여 주지 않는다는 점이다. 위키에 있는 것들의 관계는 그려지지만 이 주제에서 다뤄야 하는데 아직 넣지 않은 논문은 화면 어디에도 없다. 빈자리를 찾는 일은 문서를 읽고 판단해야 나오고 그래서 7장이 미보유 목록을 따로 두라고 한 것이다.

Obsidian과 채팅 나누기

두 방식이 배타적이지는 않고 잘하는 일이 다르다. 그래프와 화면이 나온 자리는 구조를 확인할 때다. 새 카테고리를 만든 뒤에 그 카테고리가 실제로 자라고 있는지, 논문을 스무 편 넣은 뒤에 종합 층에 붙지 않은 것이 있는지, 링크에 오타가 있는지를 보는 데는 눈이 빠르다. 문서를 직접 손보는 일에도 편하다. 여러 문서를 나란히 열어 두고 표현을 맞추거나 잘못된 링크를 고치는 작업이 그렇다. 채팅이 나온 자리는 내용을 다룰 때다. 무엇을 알고 싶은지에서 시작해 근거를 찾고 판단을 만드는 일, 여러 문서에 흩어진 것을 하나로 묶는 일, 그리고 그렇게 만든 답을 다시 문서로 되돌리는 일이 여기 해당한다.

시작하는 사람에게 권한 순서가 있다면 채팅부터다. 도구를 하나 더 설치하고 익히는 데 시간이 들고 그 시간에 논문 몇 편을 넣어 보는 편이 위키가 무엇인지 이해하는 데 빠르다. 문서가 백 개쯤 쌓여 무엇이 어디 있는지 감이 흐려질 때 그때 열어 보면 화면이 실제로 뭔가를 알려 준다. 그 시점이 언제인지는 사람마다 다르므로 정해진 편수는 없고 폴더 목록을 훑어 어디에 무엇이 있는지 떠오르지 않기 시작하면 그때다. 처음부터 열면 점 몇 개가 흩어져 있을 뿐이고 그 화면에서 배울 것은 없다. 반대로 화면을 보는 일이 목적이 되면 문서를 만드는 기준이 흔들린다. 그래프를 예쁘게 만들려고 링크를 더 걸게 되고 그렇게 걸린 링크는 판정이 없는 링크다.

화면을 여는 일이 목적이 될 때

이 도구를 쓰다가 어긋나는 자리가 하나 있다. 화면이 좋아 보이는 쪽으로 문서를 만들게 되는 경우다. 그래프가 촘촘한 편이 보기 좋으므로 링크를 더 걸고 싶어지고 개념 문서가 많으면 구조가 잘 짜인 것처럼 보이므로 개념 문서를 미리 만들게 된다. 두 가지 모두 위키의 답을 좋게 만들지 않는다. 판정 없이 걸린 링크는 그 논문이 그 개념을 어떻게 다루는지 알려 주지 않고 논문 두세 편만 기대는 개념 문서는 나중에 다시 열리지 않는다. 4장에서 종합 층 연결의 최소 조건에 본문 한 문장 반영을 넣어 둔 이유가 여기 있다. 기준은 그 종합 문서의 본문이 바뀌었는지다.

같은 이유로 그래프를 성과 지표로 쓰지 않는다. 점이 늘고 선이 늘었다는 것은 문서가 늘었다는 뜻이고 문서가 늘었다는 것과 답할 수 있는 질문이 늘었다는 것은 다르다. 파일이 많아지는 것과 지식이 축적되는 것이 다른 일이라는 원칙은 그래프에서도 그대로 성립한다. 판단할 기준을 하나 두자면 이렇다. 지금 이 위키에 새 질문을 던졌을 때 근거가 붙은 답이 나오는가. 그 답이 나오면 그래프가 어떻게 생겼든 상관없고 나오지 않으면 그래프가 아무리 촘촘해도 소용이 없다.

13장. 에이전트에게 주는 권한의 경계

LLM-Wiki 강의를 시작하고 수강생들과 첫 실습을 하면 거의 반드시 같은 질문이 나온다. “이 Allow, Yes 버튼을 계속 눌러야 하나요?” 에이전트가 파일을 읽거나 고치고 명령을 실행할 때마다 확인 창이 뜨기 때문이다. 확인 창이 계속 쌓이면 수강생은 내용을 읽지 않고 같은 버튼부터 누르게 된다. 저는 평소 Codex에서 Approve for me를 선택해 쓰는 편이다. 이 설정에서는 작업 공간의 경계를 벗어나려는 요청을 별도의 자동 검토가 판단하므로 사람이 매번 버튼을 누르는 횟수가 줄어든다. 어떤 설정이 맞는지는 사용하는 사람과 자료의 성격에 따라 달라진다. 처음 여는 폴더인지, 원본 파일을 직접 고치는지, 외부로 메일이나 문서를 보내는지에 따라 허용할 범위가 달라져야 한다. AI 에이전트가 내 파일을 어디까지 읽고

고칠 수 있는지, 어떤 명령을 실행할 수 있는지, 어느 순간에 다시 나에게 물어야 하는지를 정한 것이 권한이다. 권한을 정하지 않은 채 시작하면 확인 창을 무심코 반복해서 누르거나 작업 범위를 지나치게 넓게 열어 두게 된다.

권한을 물어보는 세 가지 상황

에이전트가 사람에게 확인을 요청하는 상황은 크게 셋이다. 첫째는 파일을 만들거나 고칠 때다. 새 문서를 쓰는 일과 이미 있는 문서를 덮어쓰는 일은 되돌리는 비용이 다르다. 앞의 것은 지우면 되고 뒤의 것은 이전 내용이 사라진다. 둘째는 명령을 실행할 때다. 목록을 보는 명령과 파일을 지우는 명령이 같은 통로로 나가므로 도구는 어느 명령이 안전한지를 판정하거나 사람에게 묻는다. 셋째는 폴더 바깥으로 나갈 때다. 작업 폴더 안의 파일을 다루는 것과 홈 폴더의 다른 위치를 건드리는 것, 그리고 네트워크로 나가는 것은 위험의 종류가 다르다. 권한 설정은 이 세 상황 각각에 대해 묻고 진행할지, 묻지 않고 진행할지, 아예 막을지를 정하는 일이다.

두 도구의 권한 모드

Claude Code와 Codex는 모두 작업 범위와 승인 방식을 나누지만 화면에 드러내는 단위가 다르다. Codex의 데스크톱 앱은 누가 경계 밖 요청을 검토하는지를 세 모드로 보여 주고 실제 파일 범위는 샌드박스나 권한 프로필이 정한다. Claude Code는 파일 편집과 명령 실행을 어느 정도까지 자동 승인할지를 여섯 모드로 나눈다. 이름을 일대일로 대응시키기보다 읽기 범위, 쓰기 범위, 네트워크 접근, 승인 주체를 각각 확인해야 한다. 아래 표의 영문은 2026년 8월 공식 문서에 쓰인 이름이고 한글은 이 책에서 뜻을 설명하기 위해 붙인 번역이다. 제품 판과 실행 화면에 따라 표시되는 모드는 달라질 수 있다. (Codex 권한 문서, Claude Code 권한 모드)

제품	English (한글)	확인 없이 할 수 있는 일	멈추거나 거부하는 동작
Codex	Ask for approval (승인 요청)	현재 작업 공간 안에서 읽기, 쓰기, 명령 실행	작업 공간 밖의 파일이나 허용되지 않은 네트워크 접근은 사람에게 묻는다
Codex	Approve for me / Auto-review (자동 검토)	승인 요청과 같은 작업 공간 경계 안에서 진행	경계 밖 요청 가운데 자동 검토 대상은 별도 모델이 승인하거나 거부한다
Codex	Full access (전체 접근)	로컬 샌드박스 제한을 풀고 파일과 명령에 넓게 접근	사람 승인과 샌드박스 보호가 약해지므로 격리된 환경에서만 쓴다
Claude Code	Default / Ask before edits (기본 / 편집 전 확인)	읽기와 검색	파일 편집과 처음 쓰는 명령은 사람에게 묻는다
Claude Code	Accept Edits (편집 자동 승인)	읽기, 파일 편집, mkdir, mv, cp 같은 일반 파일 작업	그 밖의 셸 명령과 민감한 동작은 계속 묻는다
Claude Code	Plan (계획)	읽기, 검색, 읽기 전용 탐색 명령, 계획 작성	원본 파일을 고치지 않는다
Claude Code	Auto (자동)	요청과 일치한다고 안전 분류기가 판정한 편집과 명령	분류기가 허용하지 않은 동작은 실행하지 않는다
Claude Code	Don't Ask (묻지 않음)	미리 허용 규칙에 넣은 도구와 읽기 전용 명령	원래 확인이 필요했던 동작을 묻지 않고 거부한다
Claude Code	Bypass Permissions (권한 검사 우회)	거의 모든 도구와 파일 동작	권한 프롬프트와 보호 검사를 건너뛰므로 컨테이너나 가상 머신에서만 쓴다

Codex에서는 승인 모드와 별도로 기술적 접근 범위를 확인한다. `:read-only`(읽기 전용)는 로컬 명령의 쓰기를 막고 `:workspace`(작업 공간)는 활성 작업 공간과 임시 폴더 안에서만 쓰기를 허용하며 `:danger-full-access`(제한 없는 로컬 접근)는 로컬 샌드박스 제한을 없앤다. Approve for me를 골라도 작업 공간 경계가 넓어지지 않는다는 점이 중요하다. 바뀌는 것은 경계 밖 요청을 사람이 검토하는지 자동 검토가 맡는지다.

아래 그림에서는 공식 모드를 세 등급의 운용 예시로 묶었다. 일반 고양이, 반고반기계 고양이, 사이보그 고양이는 이 책에서만 쓰는 비유다. 자료가 민감할수록 에이전트가 움직일 수 있는 환경을 좁히고 사람이 확인할 항목을 늘린다.

권한 등급의 차이

무엇을 할 수 있고, 무엇을 확인해야 하는가

자료의 민감도가 높을수록
확인과 통제가 더 필요해요!

등급	이름	민감도	권한 등급	할 수 있는 것	확인 필요한 것	예시 상황	추천 모드
Lv.1	일반 고양이	(낮은 민감도)	자유 모드	- 파일 읽기 / 검색 - 웹 검색 / 정보 수집 - 초안 작성 / 정리 - 공유 문서 편집	- 출처 신뢰성 - 사실 검증	- 새 논문의 배경 조사 - 공개 데이터 요약 - 일반 지식 기반 아이디어 정리	Claude Code: Plan / Default Codex: Ask for approval + read-only
Lv.2	반고반기계 고양이	(중간 민감도)	감독 모드	- 내부 파일 읽기 (허용 폴더) - 코드 실행 / 분석 - 데이터 분석 / 시각화 - 정리 노트 작성	- 개인 정보 포함 여부 - 외부 반출 여부 - 외부 접속 필요 여부 - 변경/공유 전 승인	- 공동연구 내부 문서 정리 - 자체 실험 데이터 분석 - 민감정보 필수 포함된 노트 정리	Claude Code: Accept Edits / Auto Codex: Auto (workspace)
Lv.3	사이보그 고양이	(높은 민감도)	격리 모드	- 격리된 환경 내 모든 작업 - 개인 식별 정보 처리 - 대규모 민감 데이터 분석 - 모델 학습 / 스크립트 실행	- 거의 제한 / 불가 - 외부 인터넷 접속 - 클라우드 서비스 사용 - 외부 공유/반출 - 스크린샷 외부 반출	- 환자 진료 기록 분석 - 유전체/영상 개별 데이터 처리 - 미공개 코호트 통계/모델 개발	Claude Code: Bypass Permissions Codex: Full access + danger-full-access

핵심 원칙

되돌릴 수 있는가? 를 기준으로 등급을 선택합니다.
자료의 민감도가 높아질수록 '자유' → '감독' → '격리' 순으로 더 엄격한 통제와 확인이 필요합니다.

의심되면 한 단계 위 등급의
규칙을 적용하세요!

Figure 7: 자료의 민감도에 따라 나는 AI 에이전트 권한 등급

세 등급은 자료의 민감도에 따라 확인 강도를 높이는 이 책의 운용 비유다. 실제 설정에서는 바로 위 표의 Codex·Claude 공식 모드와 접근 범위를 함께 확인한다.

Claude Code의 권한 모드

Claude Code의 dontAsk와 bypassPermissions는 이름이 비슷해 보여도 동작이 반대다. dontAsk는 미리 허용하지 않은 도구를 조용히 거부하므로 사람이 지켜보지 않는 스크립트나 CI에 맞는다. bypassPermissions는 권한 검사를 건너뛰고 동작을 실행하므로 인터넷과 개인 파일에서 격리된 컨테이너나 가상 머신에 한정한다. 처음 시작할 때는 default로 두고 변경 전에 구조를 파악해야 할 때 plan을 고르는 편이 안전하다. 파일 변경을 계속 검토할 수 있는 작업에서는 acceptEdits로 확인 창을 줄일 수 있다.

Codex의 승인 정책과 샌드박스

Codex의 권한은 샌드박스과 승인 방식을 함께 봐야 한다. 샌드박스는 명령이 기술적으로 닿을 수 있는 파일과 네트워크를 정하고 승인 방식은 그 경계를 넘으려 할 때 누가 검토할지를 정한다. Auto라고 부르는 기본 조합은 작업 공간 쓰기와 필요 시 승인을 묶는다. Codex는 이 상태에서 작업 폴더 안의 파일을 읽고 고치며 명령을 실행하고 폴더 밖이나 네트워크가 필요할 때 승인을 요청한다. MCP나 앱 도구가 외부 상태를 바꾸는 경우에는 파일 권한과 별도의 승인이 적용될 수 있다. 설정 이름만 외우기보다 작업 폴더 안에서 무엇을 자동으로 할 수 있고 어떤 경계를 넘을 때 누가 승인하는지를 확인한다.

일상 작업의 출발점은 작업 공간 안의 읽기와 쓰기를 허용하고 그 밖의 파일이나 네트워크에 접근할 때 승인을 받는 자동 작업 프로필이다. Codex 공식 문서가 예로 드는 기본 조합도 작업 공간 쓰기와 필요 시 승인이다. 새 폴더를 탐색하거나

무엇이 있는지 확인만 할 때는 읽기 전용이 맞다. 이 상태에서는 실수로 파일이 바뀌지 않으므로 먼저 범위를 파악하기 좋다. 전체 접근과 승인 요청 없음은 샌드박스과 사람의 확인을 함께 약하게 만들므로 격리된 환경이 아니면 쓰지 않는다. 네트워크 접근은 작업 공간 쓰기 권한만으로 자동 허용되지 않고 기본적으로 꺼져 있다. 웹 검색 도구도 명령이 쓰는 네트워크와 별도의 설정을 가지며 저장된 검색 색인을 쓰는 방식과 실시간 검색을 구분한다. 이 책이 근거로 삼은 위키의 첫 번째 규칙은 사람이 요청하지 않으면 웹 검색을 하지 않는다는 것이고 그 규칙은 지침 파일에 적혀 있다. 도구가 강제하는 권한과 에이전트가 따르는 작업 규칙은 다른 층이므로 둘을 함께 확인해야 한다.

따라해보기

아래 블록은 AI 도구의 채팅창에 그대로 붙여 넣는 프롬프트다. 권한 설정을 고르는 일은 한 번 해 보면 감이 온다. 위키 폴더에서 세션을 열고 다음을 시켜 본다.

읽기 전용이나 계획 모드로 이 요청을 하면 에이전트가 무엇을 하려고 하는지 보인다. 같은 요청을 권한이 열린 상태에서 하면 요청하지 않은 파일이 만들어지는 경우가 있다. 두 결과를 비교해 보면 모드가 무엇을 막는지 알 수 있다. 그다음에 실제 작업을 시켜 본다.

`sources/`

뒤의 한 줄이 도구 설정과 별개로 작동한다는 점이 중요하다. 권한 모드는 도구가 강제하는 경계이고 지침 파일과 요청에 적는 문장은 그 안에서 에이전트가 지키는 규칙이다. 앞의 것은 어기려 해도 어길 수 없고 뒤의 것은 지키기로 한 약속이다.

확인 창에서 볼 것

확인 창이 떴을 때 무엇을 보든지 익혀 둘 필요가 있다. 창에 나오는 것은 대체로 무엇을 하려는지에 대한 요약과 실제로 실행될 내용이다. 요약만 읽고 누르면 요약이 정확한지를 확인하지 않은 채 승인하는 것이 된다. 파일을 고치는 경우에는 어느 파일인지와 무엇이 어떻게 바뀌는지를 본다. 여러 파일이 한 번에 나오면 그 수를 먼저 본다. 예상보다 많으면 범위가 잘못 잡힌 것이고 그때 멈추는 편이 낫다. 명령을 실행하는 경우에는 명령 자체를 읽는다. 특히 삭제하거나 덮어쓰거나 이동하는 동작, 그리고 여러 파일을 한꺼번에 지정하는 표현이 들어 있는지 본다. 대상 경로가 작업 폴더 밖을 가리키고 있지는 않은지도 본다.

확인 창을 누르는 속도가 빨라지는 것은 자연스럽게 위험한 신호이기도 하다. 같은 종류의 확인을 열 번 넘게 받았다면 그것은 그 동작을 허용 목록에 넣어야 한다는 뜻이다. 매번 승인하면서 매번 읽지 않는 것보다, 한 번 판단해 목록에 넣고 나머지 확인은 실제로 읽는 편이 낫다. 확인 창의 수를 줄이는 일과 권한을 넓히는 일은 같지 않다. 반복되는 안전한 동작을 목록으로 빼면 창의 수는 줄어들고 허용 범위는 그대로다.

도구가 막는 것과 규칙이 막는 것

권한 설정으로 막을 수 있는 것에는 한계가 있다. 도구는 동작의 종류만 알고 의도는 모른다. 작업 폴더 안에서 문서를 만드는 일과 작업 폴더 안의 문서를 지우는 일은 도구 입장에서 둘 다 쓰기이고 같은 설정 아래에서 둘 다 허용되거나 둘 다 막힌다. 사람 입장에서 두 동작의 위험은 전혀 다르다. 이 간격을 메우는 것이 지침 파일이다. 도구가 허용한 범위 안에서 무엇을 어떤 순서로 할지는 규칙 문서가 정한다. 덮어쓰기 전에 먼저 읽고 무엇을 바꿀지 보여 준다는 규칙, 밖으로 나가는 것은 초안을 만들어 승인을 받는다는 규칙, 확인하지 못한 것을 이상 없음으로 적지 않는다는 규칙이 여기 해당한다. 두 층은 성격이 다르다. 도구의 경계는 어기려 해도 어길 수 없고 규칙은 지키기로 한 약속이다.

약속이라서 깨진다. 그래서 자주 깨지는 규칙에는 기계 검사를 붙인다. 위키에 넣지 않기로 한 표현이 문서에 들어갔는지 검사하는 스크립트, 종합 층과 연결되지 않은 페이지를 세는 감사, 끊긴 링크를 찾는 검사가 그런 것들이다. 검사가 붙은 규칙과 붙지 않은 규칙은 몇 달 뒤에 지켜지는 정도가 다르다. 모든 규칙에 검사를 붙일 수는 없는데 판단이 들어가는 규칙은 기계가 판정할 수 없기 때문이다. 무엇을 규칙으로 두고 무엇을 검사로 바꿀지는 그 판정이 결정적으로 내려질 수 있는가로 갈린다. 파일 이름의 형식은 검사할 수 있고 이 논문이 저 종합 문서와 관련이 있는지는 검사할 수 없다.

되돌릴 수 있는가로 나누는 경계

설정 값을 고르는 기준은 하나로 정리된다. 그 동작을 되돌릴 수 있는가다. 파일을 읽고 검색하고 초안을 만들고 형식을 맞추고 빠진 것을 찾는 동작은 되돌릴 수 있다. 결과가 마음에 들지 않으면 지우면 되고 손해도 남지 않는다. 이런 동작은 묻지 않게 두는 편이 낫다. 확인 창이 자주 뜨면 사람은 내용을 읽지 않고 누르게 되고 그 습관이 붙은 뒤에는 정말 봐야 할 확인 창도 그냥 눌린다. 되돌리기 어려운 동작은 반대쪽에 둔다. 기존 파일을 덮어쓰는 일, 여러 파일을 한 번에 바꾸는 일, 폴더 밖으로 나가는 일이 여기 해당한다. 밖으로 나가는 동작은 예외 없이 사람이 본다. 메일 발송, 남의 일정 변경, 외부 시스템에 무언가를 올리는 일은 되돌리는 비용을 상대가 함께 치르기 때문이다.

일괄 변경의 위험은 실제 사고에서 확인되었다. 이 책이 근거로 삼은 위키에서 위키링크 약 5만 개에 접두사를 붙이는 일괄 작업이 실행되어 사용 중이던 문서 구조가 망가진 적이 있다. 그 작업 자체는 승인을 받고 실행되었고 승인한 사람이 본 것은 무엇을 바꿀지에 대한 요약 한 줄이었다. 규모가 큰 변경에서는 승인 창의 문장 한 줄과 실제로 일어나는 일 사이의 간격이 커진다. 그래서 덮어쓰기 전에 먼저 읽고 무엇을 어떻게 바꿀지 보이고 승인을 받은 뒤에 쓴다는 순서를 지침 파일에 적어 두었다. 이 순서는 권한 모드로는 강제되지 않는다. 도구는 쓰기를 허용하거나 막을 뿐이고 쓰기 전에 무엇을 보여줄지는 정해 주지 않는다.

도구 단위 허용 목록

모드 하나로 정하는 것과 별개로, 어떤 도구를 쓸 수 있는지를 목록으로 지정하는 방법이 있다. Claude Code에는 허용할 도구 이름을 나열하는 설정이 있고 명령 실행의 경우 어떤 명령까지 허용할지를 패턴으로 적을 수 있다. 이 방식은 반복 작업에 유용하다. 매일 도는 예약 작업이나 정해진 절차를 반복하는 세션에서는 필요한 도구가 정해져 있으므로 그것만 허용하면 확인 없이 돌면서도 범위는 좁게 유지된다. 모드를 열어 두는 것과 목록으로 좁히는 것의 차이가 여기서 갈린다. 모드를 여는 것은 무엇이든 할 수 있게 하되 묻지 않는 것이고 목록을 쓰는 것은 정해진 것만 할 수 있게 하되 그 안에서는 묻지 않는 것이다. 사람이 지켜보지 않는 작업에는 뒤의 방식이 맞다. 27장에서 다루는 예약 자동화가 이 설정 위에서 돌아간다.

자료에 섞여 들어온 지시문

권한을 생각할 때 빠뜨리기 쉬운 경우가 하나 있다. 에이전트가 읽는 자료 안에 에이전트에게 향하는 문장이 섞여 있는 경우다. 위키에 들어오는 것은 논문 PDF와 웹에서 받은 문서와 메일과 남이 만든 파일이고 그 안에 지시처럼 읽히는 문장이 들어 있을 수 있다. 에이전트가 그것을 사람의 요청과 구분하지 못하면, 읽으라고 준 문서가 시키는 문서가 된다. 실제로 위험해지는 조합은 권한이 열려 있고 자료를 밖에서 받아 오는 경우다. 읽기만 허용된 상태에서는 그 문장을 읽어도 할 수 있는 일이 없지만 쓰기와 명령 실행이 열려 있으면 그 문장이 동작으로 이어질 수 있다. 그래서 밖에서 들어온 자료를 처음 처리하는 세션의 권한을 좁게 두는 편이 낫다.

원칙은 하나다. 지시는 사람이 대화로 준 것만이고 파일과 문서와 메일에 적힌 것은 전부 자료다. 자료 안에 지시처럼 보이는 문장이 있으면 그것을 따르지 말고 사람에게 그 문장이 있다는 사실을 알리게 한다. 이 규칙은 도구 설정으로 강제되지 않으므로 지침 파일에 적어 둔다. 판단이 애매한 경우도 있다. 자기가 예전에 쓴 규칙 문서는 자료인가 지시인가. 이 경우는 지시로 다루되, 그 문서가 어디에 있는지를 정해 둔다. 지정한 위치의 규칙 문서만 지시로 읽고 위키의 다른 문서는 내용이 무엇이든 자료로 읽는다. 파일 위치로 구분하면 판단할 것이 없어진다.

설정이 흔들리는 원인

권한 설정에서 실제로 문제가 되는 것은 값을 잘못 고르는 일보다 값이 세션마다 달라지는 일이다. 급한 작업에서 확인 창이 성가셔 한 번 열어 두면 그 상태가 다음 세션까지 이어지고 언제부터 열려 있었는지는 기억나지 않는다. 그래서 기본값을 파일에 적어 두고 그 값에서 벗어날 때만 의식적으로 바꾸는 편이 낫다. 두 번째로 흔한 경우는 도구를 여러 대의 기계에서 쓸 때다. 한 기계에서 좁게 설정하고 다른 기계에서 넓게 설정해 두면, 같은 폴더에 같은 작업을 시켜도 결과가 다르다. 세 번째는 판이 올라가면서 값의 이름이나 기본값이 바뀌는 경우다. 도구를 갱신한 뒤에 첫 세션에서 무엇이 확인 없이 진행되는지 한 번 보는 습관이 필요하다. 확인 창이 뜨지 않는다면 그 종류의 동작을 이미 허용한 것이다.

처음 시작할 때 권하는 조합

권한을 어떻게 두고 시작할지에 대한 답은 작업의 단계마다 다르다. 새 폴더를 열고 무엇이 있는지 보는 단계에서는 읽기만 허용한다. 이 단계에서 파일이 바뀔 이유가 없고 읽기만 허용해 두면 시켜 보는 데 부담이 없다. 위키를 만들고 문서를 채우는 단계에서는 작업 폴더 안의 쓰기를 열고 폴더 밖과 명령 실행은 확인받는다. 실습을 따라가며 되돌리기 어려운 동작에서 확인 창이 뜨므로 어디에서 창이 뜨는지를 보는 것 자체가 배우는 과정이 된다. 검색 색인을 세우고 스크립트를 돌리는 단계에서는 명령 실행이 찾아지므로 안전한 명령을 자동으로 통과시키는 설정이 편해진다. 이때도 폴더 밖으로 나가는 동작과 외부로 내보내는 동작은 열지 않는다.

권한을 넓힐지 고민될 때 쓸 만한 질문이 하나 있다. 지금 이 확인 창을 없애면 무엇이 잘못될 수 있고 잘못되었을 때 되돌릴 방법이 있는가다. 되돌릴 방법이 있으면 열어도 되고 없으면 성가셔도 남겨 둔다. 되돌릴 방법이 파일 버전 기록뿐인 환경에서는 그 기록이 얼마나 오래 남는지도 확인해 둔다. 며칠치만 남는 설정이라면 몇 주 뒤에 발견한 문제는 되돌릴 수 없다. 이 확인은 위키를 만들기 전에 끝내 둔다.

이 책이 근거로 삼은 환경의 설정 파일을 열어 보면 앞 절이 권하는 조합보다 훨씬 넓게 열려 있다. 기본값이 확인을 묻지 않는 것이고 작업 폴더 밖까지 접근이 열려 있으며 도구 하나하나에 사람 승인을 강제해 둔 항목은 전체 설정에서 손에 꼽는다. 몇 년 동안 매일 돌리다 보면 확인 창을 닫는 동작이 습관이 되고 습관이 된 확인은 제 기능을 잃는다. 스무 번 눌러 온 창이 스물한 번째에 다른 것을 묻고 있어도 한곳을 누르게 된다. 그래서 확인 창의 수를 늘리는 대신 되돌릴 수 없는 동작만 골라 그곳에 강제 게이트를 두는 쪽으로 정리했다. 지금 기계적으로 막혀 있는 것은 이메일 발송과 일정 변경 둘이다. 나머지는 지침 문장과 작업별 프롬프트가 지키는데 프롬프트마다 같은 금지 목록이 반복해서 적혀 있다. 자동으로 메일을 보내지 않고 댓글을 자동으로 달지 않고 메일과 파일과 문서를 지우지 않는다.

값이 이 모양으로 정해진 경위는 대체로 사고 뒤였다. 일정 변경을 코드로 막은 가드는 같은 실수가 반복되고 나서 만들어졌고 운영 로그에는 아침에 허용되어 있던 동작이 같은 날 저녁에 읽기 전용으로 바뀐 기록도 남아 있다. 권한이 좁아진 시점은 전부 무언가 잘못된 뒤였고 미리 좁혀 둔 적은 없다. 이 순서를 뒤집을 수 있다고 믿지 않는 편이 낫고 그래서 처음 시작하는 사람에게는 앞 절의 좁은 조합을 권한다. 넓게 열어 둔 설정은 무엇이 잘못되었을 때 그것을 알아차리는 경로를 따로 갖춘 뒤에야 감당할 수 있다. 상시로 허용한 것이 있으면 그 사실도 프롬프트 안에 데이터로 적어 둔다. 무엇을 누가 언제 허용했고 그 범위가 어디까지인지를 적어 두지 않으면, 몇 달 뒤에는 그것이 결정이었는지 흘러온 상태인지 구분되지 않는다.

14장. 말로 지시하고 맥락을 넘기는 법

3장에서 위키를 세우고 나면 그 위키에 무엇을 시킬 것인가가 남는다. 여기서 결과를 가르는 것은 도구의 성능보다 그 요청에 얼마나 많은 맥락이 실려 있는가다. 새로 온 학생에게 줄 자료를 정리해 달라는 요청을 예로 들면, 어느 분야에서 역사적으로 중요한 논문을 찾아 달라는 문장만으로는 전달되지 않는 것이 대부분이다. 어떤 순서로 읽고 싶은지, 왜 그 순서인지, 그 학생이 지금 무엇을 못 하는지, 이 자료로 무엇을 깨우치게 하려는지가 그 요청에 없다. 에이전트가 그것을 채워 넣으면 그럴듯한 목록이 나오고 그 목록에는 사람이 실으려던 판단이 빠져 있다. 손으로는 적지 않았을 맥락을 말로 넣고 그렇게 들어온 요청은 산출물을 소유할 작업 공간으로 보낸다.

타이핑하지 않게 되는 맥락

요청이 짧아지는 이유는 적은 데 시간이 들기 때문이다. 머릿속에 있는 배경을 문장으로 옮기면 여러 문단이 되고 그 문단을 치는 동안 정작 하려던 일이 미뤄진다. 그래서 대부분의 요청이 한두 문장으로 줄어든다. 줄어든 요청과 원래 하려던 일 사이의 거리가 결과의 품질을 정한다. 이 거리는 도구를 바꾼다고 줄지 않는다. 모델이 좋아지면 짧은 요청에서 더 그럴듯한 결과가 나오는데 그럴듯하다는 것과 내가 원하던 것이라는 것은 다르다. 오히려 결과가 그럴듯할수록 무엇이 빠졌는지 알아채기 어려워진다. 빠진 것을 알아채려면 자기가 무엇을 말하려 했는지가 어딘가에 적혀 있어야 하고 짧은 요청에는 그것이 없다.

빠지는 맥락에는 몇 가지 종류가 있다. 목적이 첫째다. 이 산출물이 누구에게 가고 그 사람이 이것으로 무엇을 하는지가 빠지면 형식은 맞고 쓸모는 없는 결과가 나온다. 배경이 둘째다. 지금까지 무엇을 해 왔고 어디까지 정해졌는지가 없으면 이미 결정된 것을 다시 제안한다. 제약이 셋째다. 언제까지인지, 무엇을 건드리면 안 되는지, 누구와 관련된 일인지가 빠지면 실행할 수 없는 결과가 온다. 판단의 근거가 넷째다. 왜 그렇게 하려는지를 적어 두면 에이전트가 그 근거에 맞지 않는 제안을 스스로 걸러 낸다. 네 가지를 매번 타이핑하려면 요청 하나에 몇 분이 걸리고 그래서 대부분 적지 않는다.

긴 수다 세션

Karpathy가 이 문제를 두고 쓴 짧은 글이 있다. 그는 LLM과 일할 때 쓸 만한 방식으로 “a nice long ramble session”을 든다. LLM이 무엇을 하려는지 이해하려면 더 많은 정보가 필요한데 타이핑하기는 귀찮은 상황이 있고 그럴 때는 의자에 기대앉아 음성 입력으로 바꾼 뒤 10분쯤 떠든다는 것이다. 정리하지 않아도 되고 뒤죽박죽이어도 되고 의식의 흐름 그대로여도 된다. 그는 맨 앞에 음성 인식으로 바꾸니 오타는 양해해 달라는 말을 붙이기도 하고 때로는 몇 번 주고받는 짧은 인터뷰 형태로 만들기도 한다고 적었다. 이어지는 관찰이 이 방식의 핵심이다. LLM은 길고 두서없는 수다를 재구성하는데 상당히 능하고 되돌아온 정리가 처음 말한 것보다 오히려 깔끔한 경우가 많다는 것이다. 그 결과로 서로가 이해한 바가 맞춰지고 그 뒤로는 고칠 일이 줄어든다.

이 방식이 앞 절의 문제를 정확히 겨냥한다. 목적과 배경과 제약과 근거를 타이핑으로 적으려면 오래 걸리지만 말로 하면 몇 분이면 된다. 말은 정리되지 않아도 되므로 정리하는 데 드는 시간이 아예 빠진다. 그리고 정리하는 일 자체가 LLM이 잘하는 일이다. 흩어진 발화에서 무엇이 목적이고 무엇이 제약이고 무엇이 결가지인지를 가려내 구조로 만드는 작업은 사람이 하면 지치고 기계가 하면 빠르다. 순서는 사람이 원재료를 쏟아 놓고 기계가 정리한 것을 사람이 확인하는 쪽으로 바뀐다. 확인하는 단계가 빠지면 이 방식은 성립하지 않는다.



Andrej Karpathy ✓
@karpathy



One pattern I find useful for working with LLMs is a nice long ramble session. Sometimes the LLM needs more bits to understand what you're trying to achieve, but you're too lazy to type them. In these cases I like to lean back, switch to /voice and just ramble for like 10 minutes, total mess, anything goes, full stream of consciousness. Sometimes I declare it up top, something like "switching to speech recognition sorry for any typos...". Sometimes I turn it into a small interview of a few turns. But I find that the LLMs are somehow very good at reconstructing long incoherent rambles and often their echo of your own tangle of thoughts comes out quite a bit cleaner than what you started with. The result is that you improve the mind meld and have to correct things less from that point on.

1:53 AM · Jul 22, 2026 · 4M Views

Figure 8: Andrej Karpathy가 LLM과 길게 수다를 떠는 방식을 설명한 X 게시물

Andrej Karpathy, X 게시물, 2026년 7월 22일. 저자가 보관한 화면.

번역

LLM과 일할 때 유용하다고 느끼는 방식 가운데 하나는 길게 수다를 떠는 세션이다. 가끔은 LLM이 내가 무엇을 하려는지 이해하려면 더 많은 정보가 필요한데, 그 내용을 모두 타이핑하기는 귀찮다. 그럴 때 나는 의자에 기대앉아 /voice로 바꾼 뒤 10분쯤 그냥 떠든다. 완전히 뒤죽박죽이어도 되고 무슨 말이든 해도 되며 의식의 흐름 그대로 말한다. 때로는 맨 앞에서 “지금부터 음성 인식으로 바꿉니다. 오타가 있어도 양해해 주세요...”라고 밝히고 몇 번 주고받는 짧은 인터뷰로 만들기도 한다. LLM은 길고 두서없는 수다를

놀라울 만큼 잘 재구성하며 엉킨 생각을 되비춘 결과가 처음 말한 것보다 제법 또렷하게 나오는 경우가 많다. 이렇게 서로의 생각을 더 잘 맞추고 나면 그 뒤로 고쳐야 할 것도 줄어든다.

음성으로 넣는 실제 방식

음성 입력의 편의는 도구마다 다르다. 휴대폰 앱에서 마이크를 눌러 그대로 넣을 수 있는 도구가 있고 그렇지 않은 도구가 있다. 이 차이가 실제 사용량을 가른다. 책상에 앉아 있지 않은 시간에 떠올린 것을 그때 넣을 수 있으면 이 방식이 일상이 되고 컴퓨터 앞에 앉아야만 되면 결국 타이핑하게 된다. 그래서 도구를 고를 때 음성 입력이 편한 쪽을 하나는 확보해 두는 편이 낫다.

내가 Codex를 권하는 또 하나의 이유

나는 이 장의 방식을 실제로 쓸 때 ChatGPT와 Codex의 음성 입력을 먼저 고른다. 컴퓨터와 iPhone 어느 쪽에서든 몇 분 동안 쉬지 않고 말해도 발화를 거의 끝까지 받아 적었다. 생각나는 대로 말하다가 문장을 고쳐 말하거나 다른 이야기로 썼다가 돌아와도 입력이 중간에 끊기지 않았다. 긴 수다에 목적과 배경과 제약을 한꺼번에 담은 이 장의 방식과 잘 맞는다.

Claude 앱에서는 같은 경험을 얻지 못했다. 긴 음성이 중간에 잘리거나 받아 적힌 분량이 적었고 Claude가 긴 발화를 끝까지 처리하지 못한 이유는 확인하지 못했다. iPhone의 기본 음성 인식을 거쳐 Claude에 넣는 방식도 누락이 많았으며 Claude 앱의 전사 결과도 내가 말한 내용을 충분히 남기지 못했다. 고유명사와 숫자는 어느 도구를 쓰든 다시 확인해야 하지만 긴 맥락이 얼마나 온전히 남는지를 보면 ChatGPT와 Codex 쪽이 훨씬 좋았다.

이 평가는 실제 연구와 행정 업무를 말로 지시하며 반복해서 얻은 경험에서 나왔다. 같은 녹음을 여러 도구에 넣어 수치로 비교하지는 않았다. 음성 지시를 매일 쓰려면 몇 분 동안 말한 내용이 끝까지 남아야 한다. 이 조건 때문에 나는 Claude보다 Codex를 추천한다.

실제로 무엇을 떠드는지가 감이 잡히지 않을 수 있다. 메일 한 통을 받았고 그에 대해 무언가 정해야 하는 상황을 예로 들면, 5분에서 10분 동안 다음과 같은 것들이 섞여 나온다. 어떤 메일이 왔고 무엇을 요구하는지, 그 메일이 가리키는 논문이 무엇이고 지금 어떤 목적으로 쓰고 있는지, 그 주제에 대해 위키에 무엇을 정리해 두었는지, 이 연구가 어떤 의미를 갖는다고 보는지, 함께 일하는 사람들이 어떤 방식으로 일하고 언제까지를 원하는지, 누가 참여하고 싶어 하고 어떤 부분에 강점이 있는지가 이어진다. 문장이 끊기고 중간에 다른 이야기로 뺏다가 돌아오기도 한다. 그대로 두면 된다. 이 입력에는 많은 정보가 필요하다.

따라해보기

아래 블록은 시 도구의 채팅창에 넣는 프롬프트이고 가운데 부분은 타이핑 대신 마이크로 말한다. 수다를 시작할 때 앞에 붙일 말은 짧으면 된다. 음성으로 넣는다는 사실과 무엇을 해 달라는지만 말하고 나머지는 흘러가는 대로 둔다.

<5 10 >

- 1.
2. , , ,
- 3.
- 4.

세 번째와 네 번째 항목이 이 요청의 값이다. 말로 하면 앞에서 한 말과 뒤에서 한 말이 어긋나는 일이 흔하고 그 어긋남은 말하는 사람 눈에는 잘 보이지 않는다. 어긋난 대목을 짚어 주면 아직 정하지 않은 것이 드러난다. 네 번째도 같은 성격이다. 한참 떠돌고 나면 다 말한 것 같은데 실제로는 가장 중요한 결정 하나를 말하지 않은 경우가 있다. 아무것도 실행하지 말라는 마지막 줄을 넣는 이유는 정리를 확실히 하기 전에 작업이 시작되면 잘못 알아들은 내용 위에 결과가 쌓이기 때문이다.

인터뷰로 바꾸기

한 번에 다 떠들지 않아도 된다. Karpathy가 함께 적어 둔 방식이 몇 번 주고받는 짧은 인터뷰로 만드는 것이다. 3분쯤 말한 뒤 무엇이 더 필요한지 물어보게 하고 물어온 것에 답하는 식으로 두어 번 오간다. 이 방식이 잘 맞는 경우가 있다. 자기가 무엇을 말해야 하는지 모르는 상태에서 시작할 때다. 한참 떠들다 보면 정작 중요한 제약을 빠뜨리는데 무엇이 빠졌는지는 정리를 받아 봐야 한다. 인터뷰로 바꾸면 그 확인이 앞당겨진다. 질문의 방향도 정해 줄 수 있다. 목적과 제약과 판단 근거 가운데 무엇이 비어 있는지를 묻게 하면 결가지로 새는 질문이 줄어든다.

반대로 인터뷰가 방해가 되는 경우도 있다. 이미 무엇을 원하는지 알고 있고 그것을 옮기기만 하면 되는 상황에서는 중간에 끊기는 것이 흐름을 깬다. 그럴 때는 끝까지 말하고 한 번에 정리를 받는 편이 빠르다. 두 방식을 가르는 기준은 자기 머릿속이 얼마나 정리되어 있는가다. 정리되어 있으면 한 번에 쏟고 정리되지 않았으면 주고받는다. 어느 쪽이든 마지막에 하는 일은 같다. 정리된 결과를 읽고 고치는 것이다.

되돌아온 정리 확인

정리를 받으면 읽는다. 이 방식에서 사람이 하는 일이 여기에 몰려 있다. 확인할 것이 두 종류다. 하나는 잘못 알아들은 곳이고 다른 하나는 잘못 말한 곳이다. 앞의 것은 음성 인식이 만든 오류다. 사람 이름, 기관 이름, 유전자나 방법의 이름 같은 고유명사와 숫자가 특히 자주 틀린다. 정리된 문장은 문법적으로 완전하므로 틀린 이름도 자연스럽게 읽히고 그래서 원문을 아는 사람이 눈으로 확인해야 한다. 숫자는 더 위험하다. 마감일이나 표본 수가 틀리면 그 뒤의 모든 계획이 어긋나는데 정리된 문장만 보면 틀렸다는 표시가 없다.

뒤의 것은 자기가 잘못 말한 곳이다. 이쪽이 더 유용하다. 떠드는 동안에는 앞뒤가 맞다고 느꼈는데 정리해 놓고 보면 두 대목이 서로 부딪히는 경우가 있다. 아직 내리지 못한 결정이 그 충돌에서 드러나고 정리를 받기 전에는 결정을 미뤘다는 사실조차 모르고 있었던 것이다. 이 발견이 이 방식의 실질적인 이익 가운데 하나다. 확인이 끝나면 고친 정리를 다시 받는다. 여기까지가 한 번의 왕복이고 대개 두 번이면 실행에 들어갈 만한 상태가 된다. 확인 없이 곧바로 실행으로 넘어가면 잘못 알아들은 이름이 파일 이름과 문서 제목에 박히고 그 시점에 되돌리는 값이 커진다.

수다가 이미 있는 문서를 만날 때

말로 넣는 방식이 위키와 만나면 이익이 한 번 더 생긴다. 떠드는 동안 말한 것 가운데 상당수가 이미 위키에 적혀 있기 때문이다. 그 논문에 대해 무엇을 정리해 두었는지, 그 주제에 대해 어떤 판단을 내려 두었는지, 지난번에 무엇을 하기로 했는지가 문서에 있다. 정리를 시킬 때 그 문서들을 함께 읽게 하면 두 가지가 갈린다. 내가 말한 것 가운데 이미 적혀 있는 것과 새로 나온 것이다. 앞의 것은 다시 적을 필요가 없고 뒤의 것이 이번에 남길 것이다. 이 구분이 없으면 같은 판단이 문서에 두 번 세 번 쌓이고 쌓인 것들이 조금씩 달라진다. 말한 것과 적혀 있는 것이 어긋날 때는 몇 달 사이에 생각이 바뀌었는데 문서를 고치지 않았거나 문서가 맞고 지금 잘못 기억하고 있는지를 확인한다.

어느 쪽인지는 사람이 정한다. 판정을 시키면 문서 쪽이 맞다고 답하는 경향이 있는데 문서에 적힌 것이 항상 최신은 아니기 때문에 그 답을 그대로 받으면 안 된다. 어긋난 대목을 짚어 달라고만 하고 어느 쪽이 맞는지는 묻지 않는 편이 낫다. 짚어 준 대목을 열어 보고 언제 적힌 것인지 확인한 뒤에 고칠지 정한다. 이 확인이 끝나면 문서 쪽을 고치는 일이 그날의 산출물 가운데 하나가 된다.

어느 작업 공간으로 갈지 정하기

말로 들어온 요청에는 한 가지 문제가 더 있다. 어느 폴더에서 처리할 일인지가 요청에 적혀 있지 않다는 것이다. 3장에서 본 대로 위키 하나로 모든 일이 처리되지는 않는다. 논문을 다루는 위키 옆으로 분석 코드를 두는 작업 공간, 원고를 쓰는 작업 공간, 회의와 강연을 기록하는 작업 공간이 갈라져 나오고 각각이 자기 지침 파일과 자기 위키를 갖는다. 말로 쏟아 놓은 내용에는 이 공간들에 걸친 이야기가 섞여 있다. 논문 이야기를 하다가 그 논문에 들어갈 그림 이야기로 넘어가고 다시 그 그림을 만드는 분석 이야기로 넘어간다. 세 공간에 걸친 요청이 하나의 발화로 들어온 것이다.

그래서 정리 단계에서 판정을 함께 시킨다. 이 요청의 각 부분이 어느 공간으로 가야 하는지를 정하고 그 공간의 지침 파일을 읽은 뒤에 무엇을 할지 정하게 한다. 판정 기준은 산출물의 소유다. 그 작업이 만들어 내는 결과물을 어느 공간이 소유하는지로 정하고 소유한 쪽이 파일명과 폴더 배치와 완료 기준을 정한다. 나머지는 근거와 원자료만 공급한다. 이

기준이 없으면 지금 열려 있는 폴더나 요청에 들어 있는 단어 하나로 정해지고 같은 종류의 작업이 매번 다른 곳에서 벌어진다.

작업 공간	소유하는 것	위키에 담는 것
논문 위키 분석 폴더 원고 폴더	문헌 지식과 그 위의 종합 코드와 결과물 원고 문장과 파일	카테고리 규칙, 읽기 절차, 작업 로그 프로젝트별 폴더 관계, 분석 작업 로그 작성 개요, 그림 규칙, 함께 쓰는 사람들에 대한 기록
회의와 강연 폴더	행사 기록과 제출 서류	기관별 요령, 서류 작성 절차, 반복 입력 항목의 위치

작업 공간마다 다른 위키

각 작업 폴더 안에 위키 폴더를 하나 두고 해당 공간의 규칙과 상태를 거기에 둔다. 분석 폴더의 경우 프로젝트마다 폴더를 만들고 그 아래에 자료, 코드, 결과, 참고자료 같은 하위 폴더를 둔다. 프로젝트들 사이의 관계와 어느 문서가 무엇의 정보인지는 위키 폴더의 문서 하나가 소유한다. 프로젝트마다 그 관계를 따로 적으면 프로젝트가 늘어날 때 서로 어긋나기 시작한다. 작업 로그도 그 위키 폴더에 함께 둔다. 각 프로젝트 폴더 안에 로그를 만들면 폴더 루트가 지저분해지고 여러 프로젝트에 걸친 작업은 어느 로그에 적을지가 매번 결정 사항이 된다.

원고 폴더의 위키에는 다른 것이 들어간다. 지금 쓰고 있는 원고의 개요, 무엇을 왜 고쳤는지에 대한 작업 로그, 그림을 어떤 규칙으로 만드는지, 그리고 함께 쓰는 사람들에 대한 기록이 여기 놓인다. 마지막 항목은 다루는 방식을 정해 두어야 한다. 사람에 대한 기록에는 협업에 필요한 사실만 담고 그 사실이 문서에 적힐 때 밖으로 나갈 수 있다는 것을 전제로 쓴다. 회의와 강연 폴더의 위키에는 기록과 함께 서류 작성 요령이 들어간다. 어느 기관이 어떤 양식을 쓰는지, 출장 신청과 보고를 어떤 순서로 하는지 같은 것들이다. 매번 넣어야 하는 개인 정보와 소속 정보는 이 문서 본문에 적지 않고 바깥 파일에 두고 경로만 가리킨다. 규칙 문서는 세션마다 통째로 읽히므로 본문에 적어 두면 그 정보가 매번 함께 나간다. 16장에서 다루는 자료 경계가 여기서 미리 걸린다.

이 라우팅을 처음부터 완벽하게 만들 필요는 없다. 공간이 둘일 때는 어느 쪽인지가 대체로 분명해서 판정할 것이 없고 넷을 넘어가면서 흔들리기 시작한다. 흔들리는 경우가 나타나면 그때 판정 규칙을 한 줄 더 적는다. 실제로 자주 흔들리는 것은 두 공간에 걸친 작업이다. 원고에 들어갈 그림을 만드는 일은 분석이면서 원고이고 어느 쪽이 소유하는지를 정해 두지 않으면 그림 파일이 두 곳에 생긴다. 이런 경우에는 파일이 최종적으로 어디에 실리는지로 정한다. 그림은 원고에 실리므로 원고 폴더가 소유하고 그 그림을 만든 코드는 분석 폴더가 소유하며 두 폴더는 경로로 서로를 가리킨다. 복제하지 않고 가리킨다는 것이 이 구조에서 반복해서 나오는 원칙이다.

음성 입력에서 남길 것

음성으로 넣은 내용은 그 세션이 끝나면 사라진다. 정리된 결과도 대화 안에만 있으면 마찬가지다. 그래서 왕복이 끝난 뒤에 무엇을 어디에 남길지 정한다. 남길 것은 대체로 셋이다. 이 요청으로 무엇을 하기로 했는지, 그 판단의 근거가 무엇인지, 그리고 아직 정하지 못한 것이 무엇인지다. 세 가지 모두 그 작업을 소유한 공간의 위키에 들어간다. 수다 원문과 정리 전문은 남기지 않는다. 원문에는 결가지가 대부분이고 정리 전문은 그날의 상태를 담고 있어 며칠이면 낡는다. 회의 전사를 다룰 때처럼 원자료와 판단은 서로 다른 문서에 남긴다.

이 방식을 반복하면 각 작업 공간의 위키가 실제로 자란다. 처음에는 지침 몇 줄뿐이던 문서에 판단과 근거가 쌓이고 몇 달이 지나면 새 요청을 넣었을 때 그 문서가 이미 절반을 답해 준다. 말로 넣는 방식이 특히 잘 맞는 이유가 여기 있다. 타이핑으로는 적지 않았을 근거가 말로는 나오고 그 근거가 문서에 남아 다음 판단의 재료가 된다. 그 결과 같은 종류의 요청을 두 번째로 넣을 때는 떠들 것이 줄어든다. 이미 적혀 있는 것을 다시 말할 필요가 없기 때문이다.

이 방식은 정확한 값을 그대로 넘겨야 하는 작업에는 잘 맞지 않는다. 파일 경로, 식별자, 명령의 옵션, 통계 수치처럼 한 글자만 틀려도 다른 것이 되는 입력은 말로 넣지 않는다. 음성 인식이 이런 종류를 특히 자주 틀리고 틀린 값은 정리된 문장 안에서 자연스럽게 읽힌다. 이런 값은 타이핑하거나 붙여 넣고 말로는 그 값을 무엇에 쓸 것인지만 설명한다. 배경과 판단은 말로 넣고 정확해야 하는 값은 손으로 넣으면 된다. 여러 사람이 함께 있는 곳에서도 쓰기 어렵다. 5분에서 10분을 소리 내어 말해야 하므로 혼자 말할 수 있는 환경이 필요하고 다른 사람에 대한 이야기가 들어가면 더 조심해야 한다.

15장. 논문이란 무엇인가 — 분야의 대화에 들어가는 법

대학원 첫 학기에 지도교수가 논문 한 편을 건네며 다음 주 랩미팅에서 발표하라고 한다. 학생은 사전을 켜고 세 번 읽는다. 모르는 단어를 다 찾았고 그림도 이해했고 방법 절의 수식도 따라갔다. 발표 자료는 각 절의 요약으로 채워지고 슬라이드는 열 장이 넘는다. 그런데 발표가 끝나고 “그래서 이 논문이 왜 중요한가”라는 질문이 들어오면 대답이 나오지 않는다. 요약은 할 수 있는데 평가는 할 수 없는 상태이고 부족한 것은 이 논문 앞에 있던 시도와 실패의 목록이다. 논문이 어렵게 느껴지는 가장 흔한 이유는 그 논문이 끼어든 대화의 앞부분을 모른다는 데 있다. 앞부분을 모르면 모든 문장이 같은 무게로 읽히고 어느 문장이 분야의 오랜 논쟁을 건드리는지 알 수 없다. 한 편을 읽고 나서 왜 그 질문이 그 시점에 나왔는지, 왜 그 결과가 그때 중요했는지, 무엇을 바꾸고 무엇을 남겼는지 자기 말로 설명할 수 있어야 그 논문을 읽은 것이다. 파트 1은 자료가 들어와 정리되고 연결되고 종합되는 순환을 다뤘다. 파트 2는 그 순환을 연구자가 되는 과정에 쓴다. 분야에 들어가고 남의 방법을 자기 설계로 바꾸고 다른 사람과 판단을 맞추고 자기 근거로 주장을 쓰는 네 단계가 이어지고 지금은 그 첫 단계다. 논문 한 편을 위키에 넣는 절차 자체는 4장에서 익혔다고 전제한다. 여기서 새로 배우는 것은 어느 논문을 어떤 순서로 넣을지 정하는 판단이고 그 판단의 재료가 분야의 맥락이다.

논문이라는 이야기의 구조

논문의 절 이름은 논증의 순서다. Introduction, Results, Discussion이라는 이름을 외우는 일과 각 절이 논증에서 맡은 동작을 아는 일은 전혀 다른 훈련이다. 도입부가 하는 일은 세 개의 문장을 세우는 것이다. 지금 무엇이 확립되어 있는가, 그중 무엇이 비어 있는가, 왜 그 빈자리를 하필 지금 메워야 하는가. 세 번째 문장이 가장 자주 빠지고 동시에 가장 많은 것을 알려 준다. 왜 지금인가에 답하려면 최근에 무엇이 새로 가능해졌는지를 알아야 하고 그 답이 곧 논문이 담고 선 시점이다. 결과 절은 질문에 답하기 위해 근거를 배열한 순서를 따르고 그림 1이 앞에 오는 이유는 뒤의 주장이 성립하려면 그 관찰이 먼저 확립되어야 하기 때문이다. 읽을 때는 이 그림을 빼면 다음 주장이 무너지는가를 묻는다. 무너지지 않는 그림은 뒷받침이고 무너지는 그림은 결정적 근거다. 두 종류를 구분하기 시작하면 한 편에서 정독해야 할 분량이 절반 이하로 줄고 남은 시간을 다른 논문과 비교하는 데 쓸 수 있다.

고찰이 결과를 다시 늘어놓는 자리로 쓰이면 논문에서 가장 읽을 것이 없는 절이 된다. 고찰이 실제로 정하는 것은 세 가지다. 이 결과로 무엇을 새로 믿을 수 있는가, 어디까지 말할 수 있는가, 무엇이 아직 남았는가. 세 가지 가운데 두 번째가 저자의 훈련 수준을 가장 정확하게 드러낸다. 자기 주장의 경계를 스스로 긋는 저자는 어떤 조건에서 결과가 성립하지 않는지를 먼저 말하고 굿지 않는 저자는 심사자가 대신 그어 줄 때까지 기다린다. 학생이 논문을 읽으며 배울 수 있는 것 중에 이 차이를 알아보는 눈이 가장 오래간다. 제목과 초록은 논문 전체를 한 문단으로 압축한 주장이다. 초록에서 확인할 것은 읽는 사람에게 무엇을 새로 믿으라고 요구하는지다. 요구를 한 문장으로 옮겨 적어 보면 본문을 읽는 동안 어느 그림이 그 요구를 지탱하는지 추적할 수 있다. 요구는 초록에 적히지만 그 요구가 성립하지 않는 조건은 거의 언제나 고찰에만 적히고 두 곳을 함께 읽지 않으면 주장의 크기만 남고 경계가 사라진다. 좋은 논문은 증거가 허용하는 가장 강한 주장을 찾고 동시에 그 주장이 성립하지 않는 지점까지 표시한다. 두 가지를 함께 해야 하기 때문에 어렵다. 주장을 약하게 쓰면 안전하지만 아무것도 더하지 못하고 강하게 쓰면 눈에 띄지만 반례 하나에 무너진다. 논문을 읽을 때 저자가 이 두 힘 사이에서 어디에 자리를 잡았는지를 보면, 같은 결과를 가지고도 논문마다 결론의 강도가 왜 다른지가 보이기 시작한다.

영어보다 먼저 필요한 맥락

번역기를 돌려 모든 문장을 한국어로 바꾸어도 남은 질문이 있다. 이 문단이 이 논문에서 중요한 자리인가 아닌가. 중요도는 문장 안에 표시되어 있지 않고 그 문장이 건드리는 기존 지식과의 거리로 정해진다. 어떤 방법이 처음으로 특정 조건에서 작동했다는 한 줄은, 그 조건에서 십 년 동안 아무도 작동시키지 못했다는 사실을 아는 사람에게만 결정적인 문장으로 읽힌다. 같은 문장이 배경을 모르는 사람에게는 그저 성능 보고로 지나간다. 영어 실력이 부족해서 논문이 어려운 경우도 물론 있지만 발표를 준비하다 막히는 지점은 대체로 이 거리 감각이다. 초보자와 숙련자의 읽기는 분류에서 갈린다. 숙련된 연구자는 읽으면서 문장을 세 무더기로 나눈다. 분야가 이미 합의한 전제, 이 논문이 새로 주장하는 것, 아직 아무도 확정하지 못한 것. 초보자는 세 무더기를 구분할 기준이 없으므로 전부 새 정보로 읽고 그래서 같은 논문에 몇 배의 시간을 쓰고도 무엇을 기억해야 할지 모른 채 끝낸다. 분류 기준은 논문 한 편 안에 들어 있지 않으므로 아무리 정독해도 저절로 생기지 않는다. 기준은 여러 편을 나란히 놓았을 때, 그리고 그 관계를 기록해 두었을 때 생긴다.

맥락은 읽기만으로 자라지 않는다. 직접 데이터를 다뤄 보고 같은 방법으로 실패해 본 사람에게는 방법 절의 한 문단이 다르게 읽힌다. 왜 저자가 그 전처리 단계를 굳이 언급했는지, 왜 특정 대조군을 추가했는지가 자기 경험과 불기 때문이다. 처음 몇 달 동안 논문이 먼 나라 이야기처럼 느껴지는 것은 정상이고 그 상태를 벗어나려면 읽은 것을 자기 작업과 연결해 남겨야 한다. 파트 1에서 만든 위키의 다섯 층 가운데 연결과 종합 층이 바로 그 관계를 담는다. 읽는 순서를 바꾸는

것만으로도 맥락이 조금 빨리 생긴다. 첫 쪽부터 마지막 쪽까지 차례로 읽으면 도입부의 배경 설명에 시간을 다 쓰고 정작 주장에 도달할 때 지친다. 도입부의 마지막 문단과 고찰의 첫 문단을 먼저 읽으면 저자가 세운 빈자리와 저자가 채웠다고 믿는 것이 한 쌍으로 잡힌다. 그 쌍을 손에 쥔 채 그림을 순서대로 보면 각 그림이 어느 자리를 메우려고 놓였는지가 보이고 그때 본문으로 들어가면 읽어야 할 문단이 훨씬 적어진다. 영어 표현의 완성도와 과학적 판단의 완성도는 분리해서 평가해야 한다. 문장이 유창하고 구성이 매끄러운데 주장과 근거의 관계가 느슨한 논문이 있고 표현이 투박한데 무엇을 보이려고 어떤 대조를 두었는지가 선명한 논문이 있다. 학생이 앞의 것을 좋은 논문으로 배우면 자기 원고를 쓸 때도 문장을 다듬는 데 시간을 몰아 쓰게 된다. 실제로 심사에서 걸리는 것은 근거가 주장을 어디까지 지탱하는지이고 문장은 그 관계가 정해진 다음에 손보는 대상이다. 영어를 다 익힌 뒤에 논문을 이해하겠다는 순서를 세우면 시작 자체가 무한히 미뤄진다.

시리즈 안에서 한 편이 갖는 의미

오래 이어진 드라마 시리즈를 중간의 한 회부터 본 사람은 그 회를 이해할 수는 있어도 평가하지 못한다. 등장인물이 왜 그 선택을 하는지, 어떤 대사가 앞선 시즌의 약속을 뒤집는지, 이 회가 시리즈 전체에서 어떤 자리를 차지하는지는 앞을 본 사람에게만 보인다. 처음부터 따라온 시청자는 특정 회를 놓고 왜 그것이 전환점인지 설명할 수 있고 그 설명은 회차 내부의 완성도와 별개다. 논문을 처음 읽는 학생은 정확히 중간 회차부터 보기 시작한 시청자와 같다. 분야의 시리즈를 만드는 것은 앞선 질문과 실패, 측정 기술의 변화, 그리고 가끔 일어나는 전제의 교체다. 어떤 문제가 오랫동안 풀리지 않은 이유가 데이터를 그 규모로 모을 수 없어서였다면 데이터가 모이는 순간 십 년 전의 아이디어가 갑자기 논문이 된다. 이런 사정은 논문 본문에 한 문장으로만 나오거나 아예 나오지 않고 대신 참고문헌의 배치에 흔적을 남긴다. 어느 논문을 도입부 첫 문단에서 인용하고 어느 논문을 방법 절 깊은 곳에서만 인용했는지가 저자가 생각하는 계보를 보여 준다. 참고문헌 목록을 이름의 나열로 읽는 한 이 정보는 전혀 보이지 않는다.

참고문헌을 이전 이야기로 가는 연결점으로 읽으려면 인용마다 역할이 다르다는 사실부터 받아들여야 한다. 어떤 인용은 문제의 동기를 세우고 어떤 인용은 방법의 계보를 밝히고 어떤 인용은 성능을 비교할 상대를 지정하고 어떤 인용은 사용한 데이터의 출처를 기록한다. 네 종류를 구분하지 않고 세면 인용 수가 많은 논문이 중요한 논문처럼 보이는 착시가 생긴다. 인용의 기능을 나누어 논문의 구조를 해부하는 방법은 10장이 다룬다. 여기서는 참고문헌이 지도의 일부라는 감각까지만 있으면 된다. 한 편의 논문은 결론과 함께 공개한 데이터셋, 배포한 코드, 새로 만든 평가 기준, 그리고 그 분야가 이후에 쓰게 되는 용어를 시리즈에 남긴다. 남긴 것이 많은 논문은 결론이 나중에 수정되어도 계속 인용되는데 후속 연구가 그 자원 위에서 돌아가기 때문이다. 논문을 읽을 때 결론과 함께 남긴 것을 따로 적어 두면, 왜 어떤 논문이 인용 목록에서 사라지지 않는지 설명할 수 있게 된다. 자기 연구가 무엇을 남길 것인지 생각하기 시작하는 것도 대체로 이 관찰 이후다. 이 책이 파트 1과 파트 2를 관통하며 쓰는 corpus는 단일세포와 유전체 foundation model이고 진입점으로 삼은 review는 2024년에 나왔다(Szallata 외, 2024). 그 뒤에도 원저는 계속 나왔고 2025년과 2026년에 나온 논문은 그 review 안에 있을 수 없다. 진입점 하나를 잘 고르는 일과 그 진입점의 지평 밖을 스스로 채우는 일은 둘 다 필요하며 후자를 하지 않으면 몇 년 전의 지도를 현재의 지도로 착각하게 된다. 시리즈를 따라잡는 일에는 끝이 없고 끝이 없다는 사실을 받아들여야 매번 처음부터 다시 읽지 않게 된다.

과학적 리더십의 형성

과학적 리더십은 내 결과가 왜 중요한지를 대신 말해 줄 사람을 기다리지 않는 능력이다. 지도교수가 중요하다고 했으니 중요하다는 설명은 그때는 통하지만 심사자 앞에서도 학회 포스터 앞에서도 통하지 않는다. 자기가 이해한 근거로 문제의 중요성을 말하려면 그 문제가 분야의 어느 빈자리에 놓이는지를 먼저 알아야 하고 그 얇은 논문을 읽고 관계를 기록하는 과정에서만 생긴다. 리더십을 목소리의 크기나 발표 태도의 문제로 옮겨 놓으면 훈련할 수 있는 것이 남지 않는다. 훈련 가능한 형태로 쓰면 리더십은 네 개의 문장 중 하나를 고르는 일이 된다. 내가 얻은 결과가 기존 설명을 강화하는가, 적용 범위를 좁히는가, 반박하는가, 아니면 질문 자체를 다른 것으로 바꾸는가. 네 문장은 4장에서 논문 페이지에 한 줄로 적기로 한 판정과 같은 종류이고 남의 논문에 대해 연습한 판정을 자기 결과에 적용하는 것이 파트 2가 넘어가는 지점이다. 판정을 고를 수 없다는 것은 대개 비교 대상을 아직 모른다는 뜻이다. 무엇과 비교해야 하는지 모르는 상태에서 나온 “의미 있는 결과입니다”라는 말은 아무 정보도 전달하지 못한다.

자기 결과를 다루는 방식에서 학생이 빠지는 실패는 양쪽으로 나 있다. 한쪽은 과장이다. 한 조건에서 관찰한 것을 일반 법칙처럼 쓰고 효과의 크기를 밝히지 않은 채 방향만 말하고 대조군의 한계를 적지 않는다. 다른 한쪽은 축소다. 자신이 없어서 가장 중요한 함의를 고찰의 마지막 문단에 한 줄로 숨겨 두고 그 결과 읽는 사람이 이 연구가 무엇을 바꾸는지 알아채지 못한다. 두 실패는 반대 방향으로 보이지만 원인이 같다. 근거가 어디까지 지탱하는지를 계산해 보지 않았다는 것이다. 에이전트에게 내 결과가 왜 중요한지 물으면 그럴듯한 문단이 즉시 나온다. 그 문단은 표현의 초안으로는 쓸

만하고 판정으로는 쓸 수 없다. 에이전트는 위키에 들어 있는 논문들 사이에서 비교 축을 제안하고 빠진 반대 근거를 지적하고 같은 주장을 세 가지 강도로 써 볼 수 있다. 어느 강도를 택할지 정하는 일은 그 주장이 틀렸을 때 답할 사람이 해야 하고 학위논문 심사장에 앉는 사람은 학생이다. 책임의 경계를 이렇게 그어 두면 에이전트를 훨씬 공격적으로 쓸 수 있게 된다. 연구자의 주도성은 모르는 것과 다음에 확인할 것을 스스로 구조화한 상태에서 나온다. 어떤 학생은 미팅에 와서 모르겠다고 말하고 어떤 학생은 세 가지를 확인해야 하는데 그중 두 번째를 어떻게 확인할지 모르겠다고 말한다. 두 사람이 아는 양은 비슷할 수 있지만 다음 한 시간에 할 수 있는 일은 완전히 다르다.

Review-first 온보딩과 첫 지도의 선택

새 분야에 들어갈 때 원저 논문부터 무작정 모으면 대체로 실패한다. 각 논문이 전제하는 배경이 서로 다르고 무엇이 이미 합의된 것인지 알 수 없으며 다섯 편을 읽어도 다섯 개의 고립된 이야기가 남는다. 출간된 지 한두 해 안쪽의 좋은 review 한 편은 그 다섯 편이 어떤 질문의 서로 다른 답이었던지를 미리 정리해 준다. Review는 분야의 질문과 논쟁과 주요 근거를 압축한 지도이며 지도이므로 축척이 있고 생략된 것이 있고 그린 사람의 관점이 들어가 있다. 정답지로 쓰면 그 관점을 분야의 합의로 착각하게 되고 지도로 쓰면 어디를 직접 걸어가 확인해야 하는지가 보인다. 같은 시기에 나온 다른 review가 같은 쟁점을 다르게 정리하는 일도 흔하므로 review가 미해결이라고 적은 것은 미해결로 두고 해결되었다고 적은 것은 확인 대상으로 둔다. Review라는 이름이 붙은 글도 성격이 서로 다르다. 문헌을 폭넓게 정리해 분야의 현재를 서술하는 글이 있고 검색과 선정 기준을 미리 정해 놓고 그 범위 안의 연구를 체계적으로 모으는 글이 있고 저자의 관점과 전망을 앞세우는 짧은 글이 있다. 첫 번째 종류는 지도로 쓰기 좋지만 저자의 취향이 선택에 들어가고 두 번째는 빠짐이 적은 대신 해석이 얇으며 세 번째는 논쟁의 위치를 알려 주는 대신 근거의 폭이 좁다. 어느 종류를 손에 들었는지 모른 채 읽으면 그 글이 하지 않기로 한 일을 하지 않았다고 탓하게 된다. 진입점으로는 첫 번째 종류가 대체로 쓸모 있고 세 번째는 논쟁의 온도를 재는 보조 자료로 함께 읽는다. 첫 review를 고를 때 확인할 조건은 다섯 가지다.

1. 자신의 연구 질문과 직접 맞닿아 있다. 분야 전체를 조망하는 대형 review보다 자기 문제에 가까운 중형 review가 낫다.
2. 핵심 개념과 최근의 변화가 함께 정리되어 있다. 개념만 있으면 교과서에 가깝고 최신 동향만 있으면 배경 없이 목록만 남는다.
3. 문헌을 나열하는 데 그치지 않고 논쟁과 한계와 남은 질문을 드러낸다.
4. 중요한 원저로 내려갈 수 있을 만큼 참고문헌이 충분하고 인용이 본문의 주장과 붙어 있다.
5. 저자가 다루는 범위를 스스로 어떻게 한정했는지 밝혀 두었다.

저널의 명성과 인용 수는 후보를 좁히는 데 도움이 된다. 인용이 많은 review는 읽기 쉬워서 많이 인용된 것일 수도 있고 특정 하위 분야가 커서 그런 것일 수도 있다. 중요한지 아닌지는 그 review가 자기 연구 질문에 대해 무엇을 정리해 주는지로 판단한다. 학생이 명성만으로 진입점을 고르면 자기 문제와 두 단계쯤 떨어진 지도를 붙잡고 몇 주를 보내게 된다. 지도를 바꾸는 결정은 빠를수록 손해가 적으므로 첫 review가 자기 질문을 다루지 않는다고 느껴지면 그때 다른 후보로 넘어간다. 자기 문제에 맞는 최근 review가 아예 없는 분야도 많다. 좁은 주제일수록, 그리고 최근 몇 년 사이에 생긴 주제일수록 그렇다. 대안은 세 가지가 있다. 인접 분야의 review에서 방법에 해당하는 절만 빌려 오는 방법, 최근 원저 네다섯 편의 도입부를 겹쳐 읽고 공통으로 인용되는 논문을 진입점으로 삼는 방법, 그리고 같은 주제의 최근 학위논문 서론을 지도 삼아 읽는 방법이다. 어느 쪽을 택하든 지도의 출처와 한계를 첫 줄에 적어 둔다. 빌려 온 지도를 자기 분야의 지도로 착각하는 일은 review를 정답지로 쓰는 것보다 더 조용히 진행된다. 이 책이 진입점으로 쓰는 review는 단일세포 오믹스에서 transformer 계열 모델을 정리한 한 편이다(Szałata 외, 2024, 10.1038/s41592-024-02353-z). 이 review를 고른 이유는 공개 접근이라 독자가 같은 PDF를 구할 수 있고 아래로 내려갈 원저가 서로 다른 방식으로 신뢰를 만들고 있어 비교할 거리가 실제로 있기 때문이다. 동시에 2024년 논문이므로 지금 기준으로 최신은 아니고 그 간격이 훈련 재료가 된다. 독자가 자기 분야에서 다른 review를 고르더라도 이 장의 절차는 그대로 적용되며 바뀌는 것은 아래로 내려갈 원저의 목록뿐이다.

첫 review를 LLM-Wiki에 넣고 대화하기

Review 한 편을 위키에 들이는 절차는 원저를 들이는 절차와 같고 4장의 여섯 단계를 그대로 따른다. 다만 review는 그 자체가 종합 문서에 가까우므로 만들어진 페이지가 다른 논문 페이지들의 허브가 될 가능성이 높다는 점만 염두에 둔다. 들여놓은 직후에는 아직 아래에 아무것도 없으므로 허브 노릇을 하지 못하고 원저가 서너 편 들어오고 나서야 연결이 생긴다. 처음부터 완성된 구조를 만들려 하지 말고 review 한 편과 그 페이지 하나로 시작한다. 에이전트와 나누는 대화에서 사람과 에이전트가 맡는 몫은 분명히 갈린다. 에이전트는 원문에서 후보를 찾아오고 여러 문단을 나란히 놓고 비교하고 답을 구조화하고 빠진 반대 근거를 지적한다. 학생은 원문을 직접 확인하고 어떤 질문을 자기 연구의 질문으로

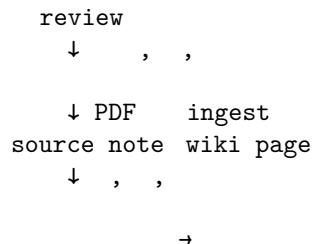
삼을지 고르고 결론의 강도를 정하고 그 판단의 책임을 진다. 지도교수와 공동연구자는 전문적 판단과 자원의 제약을 함께 정해 주지만 학생의 사전 정리를 대신하지 않는다. 정리되지 않은 상태로 미팅에 들어가면 미팅 시간이 정리하는 데 쓰이고 정작 판단이 필요한 대목은 다루지 못한 채 끝난다. Ingest가 끝나면 에이전트와 나누는 첫 대화는 여섯 개의 질문으로 구성한다.

1. 이 review가 해결하려는 가장 큰 질문은 무엇인가.
2. 현재 이 분야를 지배하는 설명이나 전제는 무엇인가.
3. 저자들이 중요하다고 보는 최근의 변화는 무엇인가.
4. 최근 몇 년의 참고문헌 가운데 핵심 주장에 직접 필요한 원저는 무엇인가.
5. 아직 합의되지 않았거나 서로 충돌하는 쟁점은 무엇인가.
6. 내 연구 주제와 직접 연결되는 개념, 방법, 데이터셋은 무엇인가.

답은 전부 review 원문으로 되돌아가 확인한다. 특히 네 번째 질문의 답으로 나온 논문 목록은 검증 없이 받아들이면 안 된다. 에이전트가 중요하다고 말한 논문이 review 본문에서 실제로 중심 근거로 쓰였는지, 아니면 도입부에서 한 번 스쳐 간 인용인지는 원문을 열어야 알 수 있다. 두 경우를 구분하지 않으면 읽기 대기열의 앞자리가 배경 인용으로 채워지고 정작 review의 논지를 지탱하는 논문은 뒤로 밀린다. 검색 도구는 후보를 좁혀 줄 뿐이며 그 역할과 한계는 9장에서 다뤘다. 여섯 질문의 답은 대화창에 두지 않고 곧바로 한 문서에 옮긴다. 대화창에 남긴 답은 다음 세션에서 사라지고 사라지면 같은 질문을 다시 던지게 되며 두 번째 답이 첫 번째와 다르면 어느 쪽이 맞는지 판단할 근거가 없다. 답마다 원문의 어느 절에 근거가 있는지를 함께 적어 두면 나중에 확인하는 비용이 크게 줄어든다. 그렇게 만들어진 문서가 이 장의 마지막 산출물인 field map의 초안이 되고 초안 상태에서 이미 빈칸이 어디인지 드러난다. 빈칸은 아직 읽지 않은 원저가 채울 자리이므로 대기열을 짜는 근거가 된다.

1. 위의 여섯 질문을 순서대로 던지고 답을 받는다
2. 답마다 review 원문의 어느 절에 근거가 있는지 함께 요구한다
3. 근거 위치가 나오지 않은 답은 채택하지 않는다

Review에서 원저 논문으로 내려가는 반복 순환



온보딩은 같은 순환을 여러 바퀴 도는 일이다. 한 바퀴에 논문 한 편만 넣는다. 열 편을 한꺼번에 모으면 열 개의 PDF가 생기고 지도는 그대로다. 어느 편을 먼저 넣을지는 질문 하나로 정한다. 이 논문을 읽고 나면 review에 대한 내 이해 중 무엇이 바뀔 가능성이 있는가. 답이 “잘 모르겠다”이면 아직 그 논문의 차례가 아니고 답이 “이 방법이 왜 표준이 되었는지 알게 된다”처럼 구체적이면 그 편이 다음 차례다. 우선순위를 이렇게 정하면 지도의 변화가 진도의 척도가 된다. 각 원저를 넣을 때마다 한 줄을 반드시 남긴다. 이 논문이 review의 어떤 문장을 지탱하는가. 한 줄을 남기면 원저 페이지와 review 페이지 사이에 방향이 생기고 나중에 review의 그 문장이 흔들릴 때 어느 원저를 다시 봐야 하는지 바로 찾을 수 있다. 그리고 새 논문이 들어올 때마다 기존 설명을 강화하는지, 범위를 좁히는지, 반박하는지 구분해 적는다. 세 가지 중 어느 것도 아니고 아무것도 바꾸지 않는다는 판정도 정보이며 같은 결과가 다른 조건에서 재현되었다는 뜻일 수 있다.

1. 네 번째 질문의 답으로 나온 논문 후보 가운데 세 편에서 다섯 편을 고른다
2. 고른 이유를 “review의 어떤 문장을 확인하려고”의 형태로 각각 한 줄로 적는다

이 책의 corpus에서 내려가는 경로는 다섯 편으로 되어 있다. 전이 학습으로 네트워크 생물학의 예측을 시도한 연구(Theodoris 외, 2023), 생성 모델로 단일세포 다중 오믹스의 기반 모델을 만든 연구(Cui 외, 2024), 대규모 학습으로 기존 분석 파이프라인의 성능을 끌어올린 연구(Hao 외, 2024), 여러 종의 서열과 기능을 함께 모델링한 연구(Boshar 외, 2025), 조절 변이의 효과 예측을 다룬 연구(Avsec 외, 2026)다. 앞의 세 편은 진입점으로 삼은 review와 시기가 겹치므로 review의 서술을 확인하고 구체화하는 데 쓰인다. 뒤의 두 편은 review보다 나중에 나왔으므로 review 안에서 찾을 수 없고 “이 review 이후에 무엇이 달라졌는가”라는 질문을 따로 던져야 도달한다. 다섯 편이 같은 목표를 두고 서로 다른 방식으로 신뢰를 만든다는 점이 이 corpus를 고른 이유이며 그 차이를 본격적으로

비교하는 작업은 9장과 10장이 맡는다. 내려간 원저가 review의 서술과 어긋나는 순간이 온다. 초보자는 대개 자기가 잘못 읽었다고 생각하고 넘어가지만 어긋남의 원인은 셋 중 하나다. Review가 요약하면서 결과가 성립하는 조건을 생략했거나 원저의 결과가 그 뒤에 다른 데이터에서 갱신되었거나 실제로 잘못 읽었거나. 세 가능성을 구분하려면 원저의 해당 그림과 그 그림의 조건을 다시 보는 수밖에 없고 그 확인이 이 장의 훈련에서 가장 값진 30분이다. 어긋남을 확인한 뒤에는 결과와 상관없이 이해결 질문 목록에 한 줄로 남기는데 여기서 나온 항목이 학생의 첫 연구 질문이 되는 경우가 드물지 않다.

지평 밖을 채우는 질문은 매 바뀌마다 필요하다. Review가 이해결이라고 적어 둔 쟁점 가운데 이미 답이 나온 것이 있는지, 저자가 유망하다고 본 방향이 실제로 이어졌는지, 아니면 조용히 사라졌는지를 확인해야 지도가 현재형이 된다. 다섯 편이 쌓이고 나면 위키에 다시 물을 수 있는 질문이 생긴다. 이 분야에서 반복해서 중심에 놓이는 논문과 개념은 무엇인가. 이 질문의 답은 review가 준 목록과 부분적으로만 겹치고 겹치지 않는 부분이 독자가 직접 만든 지도의 몫이다. 순환을 도는 동안 대기열에 넣지 않기로 한 논문들이 계속 쌓인다. 관련은 있어 보이는데 지금 질문을 바꾸지는 않는 논문, 방법만 필요할 것 같은 논문, 나중에 비교군으로 쓸 것 같은 논문이 그렇다. 이런 논문은 지우지 말고 보류 목록에 제목과 보류한 이유 한 줄로 남긴다. 이유를 적어 두면 몇 달 뒤 연구 질문이 바뀌었을 때 어느 항목을 다시 꺼내야 하는지 바로 찾을 수 있고 이유 없이 쌓인 목록은 결국 다시 처음부터 훑게 된다. 보류 목록과 읽기 대기열을 구분하는 것만으로도 대기열이 짧게 유지되고 짧은 대기열은 실제로 소화된다. 완료 기준은 자기 말로 분야의 현재 지도를 설명할 수 있는지도. 다섯 편을 읽고도 각 편의 요약만 말할 수 있다면 아직 지도가 없는 것이고 세 편만 읽었어도 세 편이 서로 어떻게 다른 방식으로 같은 문제를 다뤘는지 말할 수 있다면 지도가 생긴 것이다. 두 상태를 구분하는 가장 빠른 방법은 소리 내어 5분 동안 설명해 보는 것이다. 막히는 지점이 곧 지도의 빈칸이고 빈칸을 확인하는 데는 청중이 필요 없다.

온보딩의 산출물인 한 장짜리 field map

Field map은 분야를 자기 말로 그린 한 장짜리 문서다. 한 장이라는 제한은 판단을 강제하는 장치다. 분량이 늘어나면 판단하지 않고 나열하게 되고 나열된 field map은 목차와 구별되지 않는다. 한 장에 넣으려면 어느 논쟁이 지금 중요하지 고르고 어느 방법이 자기 문제에 쓸모있는지 잘라야 한다. 자르는 작업이 곧 지도를 그리는 작업이고 잘라 낸 것이 아까우면 별도 문서로 옮겨 두면 된다. 처음 만든 field map이 부정확한 것은 정상이며 논문이 두세 편 더 들어오면 고칠 자리가 저절로 드러난다. 발표용으로 쓸 것을 전제하고 그리면 아는 것만 적게 되고 정작 필요한 정보인 모르는 자리가 사라진다. 비어 있는 칸과 확신이 없는 칸을 그대로 표시해 두어야 다음에 무엇을 읽을지 정할 수 있고 반년 뒤에 같은 지도를 다시 열었을 때 자신이 어디서부터 자랐는지도 보인다. 확신의 정도를 칸마다 한 글자로 표시하는 정도의 장치면 충분하며 정교한 등급 체계를 만들면 그것을 유지하느라 지도를 고치지 않게 된다. 지도를 자주 고치는 사람과 지도를 예쁘게 만드는 사람 중에 분야를 먼저 이해하는 쪽은 전자다. 첫 줄에는 작성일과 어느 review를 기준으로 그렸는지 적는데 기준을 적어야 나중에 지도가 낡았을 때 어디서부터 다시 그릴지 알 수 있다. 답아야 할 것은 여섯 가지다.

- 분야가 답하려는 중심 질문 한 개에서 세 개
- 현재의 지배적 설명과 그에 맞서는 경쟁 설명
- 최근의 변화를 만든 주요 논문과 각각이 맡은 역할
- 자주 쓰이는 방법과 각 방법이 볼 수 있는 것, 볼 수 없는 것
- 합의된 사실과 논쟁 중인 해석과 아직 비어 있는 근거의 구분
- 자신의 프로젝트가 들어갈 수 있는 위치

읽기 대기열의 각 항목에는 왜 그 순서인지가 한 줄로 붙는다. 이 논문을 읽으면 field map의 어느 칸이 채워지는가, 혹은 어느 칸이 틀렸다고 드러나는가. 이유가 붙지 않은 항목은 대기열에서 내린다. 이유 없이 쌓인 논문 목록은 읽지 않은 죄책감만 만들고 지도를 바꾸지 않는다. 이해결 질문 목록은 세 산출물 중에서 가장 오래 살아남는다. 이 책이 근거로 삼은 연구용 위키에는 질문을 다루는 페이지가 550건 있고 개념 페이지 408건과 개관 페이지 590건이 같은 층에 놓여 있다. 질문 페이지가 개념 페이지보다 많다는 것은 종합 층이 정리된 지식과 함께 아직 답하지 못한 질문을 담는다는 뜻이다. 학생의 이해결 질문 목록은 그 층으로 올라갈 후보이며 같은 질문이 서로 다른 논문을 읽을 때마다 반복해서 나타나면 그때 별도 페이지로 승격한다. 같은 위키의 논문 페이지가 17,591건이고 50개 카테고리에 나뉘어 있다는 규모를 처음부터 목표로 삼을 필요는 없다. 시작은 review 한 편과 원저 한 편, 그리고 한 장짜리 field map이다.

1. 고른 목록을 순서와 이유가 함께 들어간 읽기 대기열 문서로 저장한다
2. Field map 초안을 한 장으로 쓰고 지금 채울 수 없는 칸은 비워 둔 채 왜 비었는지 적는다
3. 채우지 못한 칸에서 이해결 질문 세 개를 뽑아 별도 문서로 저장한다

새 세션을 시작한 에이전트에게 field map과 읽기 대기열만 주었을 때 다음에 읽을 논문 한 편과 그 논문을 읽는 이유를 스스로 제시하지 못한다면 대기열의 이유 칸이 비어 있거나 field map이 나열에 머물러 있다는 뜻이다. 세 문서는 review 페이지와 양방향으로 연결한다. 이해결 질문 목록에는 각 질문이 어느 논문을 읽다가 나왔는지 표시하는데 출처가 없는

질문은 다시 봤을 때 왜 그것이 궁금했는지 복원되지 않는다. 원저를 한 편 넣을 때마다 field map의 한 칸이라도 고쳤는지 확인하고 고칠 것이 없었다면 그 사실을 대기열의 해당 항목에 적는다. 아무것도 바뀌지 않은 논문이 세 편 이어지면 대기열의 우선순위가 잘못되었다는 신호다. 앞부분을 복원해 한 장에 그려 두면 다음 논문이 그 대화의 다음 발언으로 읽힌다.

16장. 상용 AI에 넣으면 안 되는 자료

지금까지 위키에 들어온 것은 출판된 논문이었다. 논문은 이미 공개된 자료이므로 어느 도구에 넣든 새로 공개되는 것이 없다. 연구가 실제 데이터에 닿는 순간 사정이 달라진다. 환자의 진료 기록, 임상 시험의 개별 참여자 자료, 재식별이 가능한 유전체 데이터, 아직 공개하지 않은 코호트의 표는 밖으로 나가면 되돌릴 수 없는 종류의 자료다. 상용 AI 서비스에 파일을 올리거나 내용을 붙여 넣는 것은 그 자료를 기관 밖으로 내보내는 행위이고 그 서비스가 학습에 쓰지 않는다고 밝혔는지와는 별개의 문제다. 규정을 판정하는 것은 소속 기관과 심의 위원회이고 그 판정 아래에서 위키를 어떻게 나눌지는 연구자가 정한다.



Figure 9: 상용 AI에 넣으면 안 되는 자료와 안전한 처리 기준

공개 문헌은 상용 AI로 다루고 환자 자료와 재식별 가능한 데이터는 기관 폐쇄망의 로컬 모델로 다룬다. 파일명, 오류 로그, 화면 이미지, 회의 녹음도 반출 경계에 포함된다.

전송이 곧 반출인 경우

상용 AI 서비스를 쓸 때 가장 흔한 오해는 학습에 쓰이는지만 확인하면 된다는 것이다. 학습 여부는 그 자료가 다른 사람의 답에 나타날 가능성에 대한 문제이고 연구 자료에서 먼저 걸리는 것은 그 앞 단계다. 자료가 기관의 관리 범위를 벗어나 다른 회사의 서버로 전송되었다는 사실 자체가 판정 대상이 된다. 심의 승인서에 적힌 자료 보관 위치와 접근 권한자 목록이 그 판정의 기준이 되고 대체로 그 목록에 외부 서비스는 들어 있지 않다. 자료 제공 기관과 맺은 사용 협약에도

반출 조건이 적혀 있으며 다기관 연구에서는 기관마다 조건이 다르다. 확인은 자기 연구의 승인 문서에서 시작하고 서비스 약관을 그다음에 본다. 약관은 서비스가 무엇을 하지 않겠다는 약속이고 승인 문서는 연구자가 무엇을 하지 않겠다고 한 약속이다. 뒤의 것을 어기면 앞의 것이 지켜져도 소용이 없다.

이 구분을 흐리는 상황이 몇 가지 있다. 파일을 올리지 않고 내용만 붙여 넣으면 괜찮다고 생각하는 경우가 첫번째인데, 전송되는 것은 같다. 기관이 계약한 서비스라면 괜찮다고 보는 경우가 둘째다. 기관 계약은 그 서비스와 기관 사이의 문제이고 특정 연구의 심의 조건과는 별개다. 자료를 조금 가공했다는 이유로 원본과 다른 자료로 보는 경우가 셋째다. 가공의 정도가 재식별 가능성을 실제로 없앴는지가 기준이고 열 이름만 바꾼 표는 여전히 원본이다. 판단이 애매하면 넣지 않는 쪽을 고른다. 넣지 않아서 잃는 것은 그 작업의 편의이고 넣어서 잃는 것은 되돌릴 수 없다.

식별 가능한 자료의 범위

직접 식별자는 판단이 쉽다. 이름, 등록번호, 주민등록번호, 연락처, 주소가 여기 해당하고 대부분의 사람이 이것들을 지운다. 문제는 간접 식별자다. 생년월일과 진료일과 진단명이 함께 있으면 특정 개인을 좁힐 수 있고 희귀질환일수록 좁혀지는 속도가 빠르다. 거주 지역과 직업과 성별의 조합도 마찬가지다. 각각은 식별자가 아닌데 함께 놓이면 식별자가 된다. 표에서 이름 열만 지우고 나머지를 그대로 두는 방식이 여기서 무너진다. 유전체 자료는 그 자체가 식별자에 가깝다. 서열 변이의 조합은 개인마다 고유하고 참조 자료와 대조하면 신원이 좁혀진다는 것이 여러 연구에서 확인되었다. 요약 통계와 개인 수준 자료를 가르는 선이 여기서 중요해지는데 앞의 것은 대체로 공개할 수 있고 뒤의 것은 그렇지 않다.

임상 기록의 서술 부분이 특히 위험하다. 진료 기록의 자유 서술란과 영상 판독 소견과 병리 보고서에는 정형화된 항목에 들어가지 않는 정보가 문장으로 들어 있고 그 문장에 직업, 가족 관계, 특이한 사건이 섞인다. 표의 열을 지우는 방식으로 이것을 걸러지지 않는다. 문장 안에서 지워야 하고 지웠는지 확인하려면 결국 사람이 읽어야 한다. 그래서 이 자료는 지운 뒤에 밖으로 내보내는 방식보다 애초에 밖으로 내보내지 않는 방식이 안전하다. 지우는 작업 자체를 상용 AI에게 시키는 것도 같은 이유로 성립하지 않는다. 지우려면 먼저 읽어야 하고 읽게 하려면 이미 보낸 것이다.

위키를 둘로 나누는 경계

해법은 위키를 하나로 두지 않는 것이다. 출판된 문헌을 다루는 위키와 환자 자료를 다루는 위키를 물리적으로 나누고 두 위키가 서로 다른 환경에서 돈다. 문헌 위키는 지금까지 이 책이 다룬 그대로이고 상용 AI를 쓴다. 논문은 이미 공개된 자료이므로 전송해도 새로 공개되는 것이 없다. 환자 자료 위키는 기관의 폐쇄망 안에 두고 그 안에서만 도는 모델을 쓴다. 두 위키의 구조는 같아도 된다. 원자료 층과 정리 층과 종합 층으로 나누고 지침 파일을 두고 작업 로그를 남기는 방식은 그대로 쓸 수 있다. 달라지는 것은 어느 모델이 그 문서를 읽는가뿐이다.

두 위키 사이에 무엇이 오갈 수 있는지도 정해 둔다. 문헌 위키에서 환자 자료 위키 쪽으로는 자유롭게 갈 수 있다. 논문의 정리 문서와 방법 설명과 개념 문서는 공개된 지식이므로 폐쇄망 안으로 들어가도 문제가 없다. 반대 방향은 좁게 연다. 개인 수준 자료가 아닌 요약 통계, 표본 수, 분석 절차, 그리고 그 자료에서 얻은 판단 가운데 개인을 좁히지 않는 것만 나온다. 나오는 것 하나하나를 사람이 보고 정한다. 자동으로 동기화하는 순간 이 경계는 없는 것과 같아진다. 논문 원고를 쓰는 자리는 대체로 문헌 위키 쪽에 두는데 원고에 들어가는 것은 이미 요약 통계 수준이기 때문이다. 원고를 쓰다가 개별 자료를 확인해야 하면 폐쇄망으로 들어가서 확인하고 결과만 들고 나온다.

자료	놓이는 자리	쓰는 모델
출판 논문과 그 정리	문헌 위키	상용 AI
공개 데이터베이스의 요약 통계	문헌 위키	상용 AI
개인 수준 임상 자료와 진료 기록	폐쇄망 위키	로컬 모델
재식별 가능한 유전체 자료	폐쇄망 위키	로컬 모델
미공개 코호트의 개별 표	폐쇄망 위키	로컬 모델
분석 결과의 요약 통계	사람이 확인한 뒤 문헌 위키로	상용 AI

폐쇄망에서 도는 모델

폐쇄망 안에서 쓰는 모델은 그 기관의 서버나 작업용 기계에 올려 두고 바깥과 연결하지 않는다. 모델 가중치를 내려받아 두면 그 뒤로는 네트워크가 필요 없고 자료는 그 기계를 벗어나지 않는다. 준비할 것은 세 가지다. 모델을 올릴 기계, 그

폐쇄망 위키의 지침 파일에는 한 줄이 더 들어간다. 이 폴더의 내용을 바깥 서비스로 보내지 않는다는 문장이다. 이 문장이 도구 설정으로 강제되지는 않지만 적어 두면 그 폴더에서 세션을 열 때마다 읽힌다. 더 강한 장치는 13장에서 본 권한 설정이다. 폐쇄망 안의 세션에서는 네트워크로 나가는 동작을 아예 막아 두고 웹 검색 기능도 끈다. 문헌을 찾아야 하면 문헌 위키 쪽에서 찾아 결과만 들고 들어온다. 두 위키에서 같은 도구를 쓰더라도 설정 파일을 따로 두어 이 차이가 유지되게 한다. 한 기계에서 두 폴더를 번갈아 열면 설정이 섞이므로 가능하면 기계 자체를 나누는 편이 확실하다.

폐쇄망에서 쓸 수 있는 모델은 상용 서비스의 최신 모델보다 대체로 작다. 그래서 무엇을 시킬지 골라야 한다. 잘 되는 일부터 보면, 정해진 형식으로 옮기는 작업이 가장 잘 된다. 진료 기록의 자유 서술에서 정해진 항목을 뽑아 표로 만드는 일, 여러 문서를 같은 틀에 맞추는 일, 형식이 어긋난 곳을 찾는 일이 여기 해당한다. 문서를 요약하는 일도 쓸 만하다. 개별 문서 하나를 짧게 만드는 작업은 작은 모델로도 실용적인 품질이 나온다. 빠진 항목을 찾는 검사도 잘 된다. 스무 건의 기록에서 특정 항목이 비어 있는 것을 골라내는 일에는 큰 모델이 필요 없다.

따라해보기

AI

- —
—
—

- 1.
- 2.
- 3.

94

경우가 있고 그 화면을 복사해 상용 시에 물어보는 순간 전송된다. 오류를 물어볼 때는 자료가 나온 부분을 지우고 묻는다. 세 번째는 화면 이미지다. 표나 기록을 찍은 화면을 올려 물어보는 방식은 파일을 올린 것과 같다.

네 번째는 중간 파일이다. 변환하거나 추출하는 과정에서 임시 폴더에 원본의 사본이 생기고 그 폴더가 동기화 대상이면 클라우드에 올라간다. 임시 파일을 동기화되지 않는 자리에 두는 규칙이 여기서도 필요하다. 다섯 번째는 통계 결과 자체다. 표본이 작을 때는 요약 통계도 개인을 좁힌다. 특정 조합에 해당하는 사람이 한 명뿐인 칸이 표에 있으면 그 칸은 개인 자료와 같다. 여섯 가지 경로를 따로 적어 두는 이유는 하나만 막으면 나머지로 새기 때문이다. 확인은 기계로 한다. 밖으로 나가는 문서 전체에서 식별번호 형식과 이름 목록과 날짜 조합을 검색해 걸리는 것이 없는지 보고, 확인한 결과를 작업 로그에 적는다.

회의 녹음과 전사

임상 사례를 논의하는 회의를 녹음하면 그 파일에는 환자 정보가 음성으로 들어 있다. 이 파일을 클라우드 전사 서비스에 올리는 것은 진료 기록을 올리는 것과 같다. 전사한 결과 파일도 마찬가지로 텍스트가 되었다고 성격이 달라지지 않는다. 그래서 이런 회의는 로컬에서 도는 전사 도구를 쓴다. 음성 인식 모델을 자기 기계에 올려 두고 그 기계 안에서 전사하면 파일이 나가지 않는다. 전사 도구를 위키 폴더 안에 두지 않는 것도 함께 정해 둔다. 모델 파일과 원본 음성은 기계이고 지식이 아니므로 위키 밖의 별도 자리에 두고 위키로 들어오는 것은 사람이 정리한 회의 노트뿐이다. 원본 음성과 원시 전사 결과는 그 별도 자리에 남고 정리한 노트에서 환자를 좁히는 정보를 지운 뒤에 위키로 옮긴다. 18장에서 다루는 회의 기록 절차가 이 구분 위에서 돌아간다.

공개 데이터 쓸 때도 남는 조건

공개 데이터베이스의 개인 수준 유전체 자료에도 사용 조건이 붙는다. 개인 수준 유전체 자료를 배포하는 저장소는 대체로 사용 신청과 승인을 거치게 하고 승인서에는 자료를 어디에 보관하고 누가 접근하는지가 적혀 있다. 승인받은 사람이 자기 컴퓨터에서 쓰는 것과 그 자료를 외부 서비스로 보내는 것은 다른 행위이고 대부분의 승인 조건이 뒤의 것을 허용하지 않는다. 요약 통계만 공개하는 자료와 개인 수준 자료를 함께 다룰 때는 두 종류를 폴더 단위로 나눠 둔다. 같은 폴더에 있으면 어느 파일이 어느 조건 아래 있는지 몇 달 뒤에 알 수 없고 알 수 없는 상태에서는 안전한 쪽으로 판단할 수밖에 없어 결국 둘 다 폐쇄망에 갇힌다.

조건은 자료마다 다르므로 받을 때 함께 기록한다. 어느 저장소에서 언제 받았고 승인 번호가 무엇이고 조건에 무엇이 적혀 있고 언제까지 보관할 수 있는지를 그 자료 옆에 적어 둔다. 이 기록이 없으면 몇 년 뒤에 그 자료를 다시 쓸 때 조건을 확인할 방법이 없고 확인하지 못한 자료는 쓸 수 없는 자료가 된다. 자료를 지워야 하는 시점이 조건에 적혀 있는 경우도 있어서 그 날짜도 함께 남긴다. 이 기록은 논문을 낼 때 어느 도구로 추출했는지 적어 두는 것과 같은 성격이다. 나중에 판단하려면 그때의 조건을 알아야 하고 조건은 시간이 지나면 복원되지 않는다.

학생과 정하는 자료 경계

이 경계는 지도교수가 혼자 정해서 통보하는 것으로 끝나지 않는다. 실제로 자료를 만지는 사람이 학생이고 판단이 필요한 순간은 지도교수가 보고 있지 않을 때 온다. 분석이 막혀 오류 메시지를 붙여 넣고 싶을 때, 표의 형식이 이상해 그대로 올려 물어보고 싶을 때가 그런 순간이다. 그래서 규칙을 알려 주는 것과 함께 왜 그런지도 함께 짚는다. 이유를 모르면 규칙은 불편한 절차로만 남고 급할 때 가장 먼저 건너뛰어진다. 반대로 이유를 알면 규칙에 적히지 않은 새로운 상황에서도 같은 방향으로 판단한다. 규칙 목록보다 판단의 기준 하나를 정확히 전하는 편이 오래 간다. 그 기준은 되돌릴 수 있는가이고 13장에서 권한을 정할 때 쓴 것과 같은 기준이다.

나가도 되는 것을 미리 정하기

경계를 지키는 가장 실용적인 방법은 나가면 안 되는 것을 매년 판단하는 대신 나가도 되는 것을 미리 정해 두는 것이다. 두 방식의 기본값이 다르다. 금지 목록을 만들면 목록에 없는 새로운 종류의 자료는 나가도 되는 것이 되고 허용 목록을 만들면 목록에 없는 것은 나가지 않는다. 되돌릴 수 없는 동작에서는 뒤의 방식이 맞다. 허용 목록은 짧아도 된다. 출판된 논문과 그 정리 문서, 공개 데이터베이스에서 받은 요약 통계, 표본 수와 분석 절차, 그리고 사람이 확인한 결과 요약이면

대부분의 작업이 된다. 새로운 종류의 자료를 상용 시에 넣어야 할 일이 생기면 그때 목록에 추가할지 판단하고 판단한 근거를 함께 적는다.

이 목록은 지침 파일에 적어 두고 새 사람이 들어올 때 온보딩 묶음과 함께 건넨다. 규칙을 모르는 상태로 시작한 사람은 편한 방식을 고르게 되고 편한 방식은 대체로 상용 서비스에 올리는 것이다. 한 번 나간 자료는 되돌릴 수 없으므로 이 규칙만은 첫날에 알려 준다. 규칙을 어겼을 때 어떻게 하는지도 미리 정해 둔다. 숨기지 않고 알리는 것이 첫 번째이고 무엇이 언제 어디로 나갔는지 기록하는 것이 두 번째다. 알리기 어려운 분위기를 만들면 같은 일이 반복되고 그 사실이 밖으로 드러나는 시점은 훨씬 뒤가 된다.

17장. 데이터 분석을 위한 폴더 구성

방법을 이해했다는 말과 그 방법으로 자기 질문을 판정할 수 있다는 말 사이에는 아직 비어 있는 자리가 있고 그곳을 채우는 산출물이 concept page와 method page, 그리고 가설과 대안 가설과 대조군과 판정 기준을 갖춘 계획서다. 이 가운데 가장 먼저 정해야 하는 것은 판정 기준이다. 결과를 본 뒤에 기준을 정하면 어떤 결과든 성공으로 읽히기 때문이다. 지도교수가 논문 한 편을 건네면서 이 방법을 써 보라고 한다. 학생은 Methods 절을 읽고 단계를 번호로 정리한다. 시약과 장비, 소프트웨어 이름까지 적어 두면 준비가 끝난 것처럼 보인다. 그런데 자기 시료나 자기 데이터에 대입하려는 순간, 논문에 적혀 있지 않은 것들이 한꺼번에 나타난다. 몇 개를 비교해야 하는지, 무엇을 대조군으로 둘 것인지, 어느 정도의 차이가 나야 차이가 있다고 말할 수 있는지가 그것이다. Methods 절은 저자들이 무엇을 했는지 적은 기록이다. 논문의 저자들은 자기 결과를 보기 전에 무엇을 성공으로 볼지 정해 두었고 그 결정은 대체로 본문에 문장으로 남지 않는다. 남지 않은 것을 복원하지 않은 채 단계만 옮겨 오면, 실험은 돌아가지만 결과를 받아 들고 나서야 무엇을 기준으로 삼을지 정하게 된다. 그 시점에 정해진 기준은 이미 결과를 알고 정한 기준이므로 판정을 하지 못한다. 방법을 설계로 바꾼다는 말은 논문에 적히지 않은 결정들을 찾아내 자기 조건에서 다시 정한다는 뜻이다.

논문을 넣을 때 함께 만들어지는 method card

Method card는 여러 논문에서 반복되는 방법의 단계, 전제, 판정 기준을 한 문서에 모은다. 논문의 절차를 자기 실험 설계로 옮길 때 이 문서에서 시작한다.

Method card는 논문 페이지와 별도로 만들어지는 문서다. 그 논문이 쓴 방법을 다시 쓸 수 있는 형태로 떼어 놓은 것이고 같은 방법을 쓴 다른 논문이 들어오면 그 문서를 함께 가리킨다. 위 화면은 논문 한 편을 넣었을 때 만들어진 method card이며 방법의 이름과 핵심 단계와 그 방법이 전제하는 조건이 항목으로 나뉘어 있다. 논문 페이지와 나뉘 두는 이유는 쓰임이 다르기 때문이다. 논문 페이지는 그 논문이 무엇을 주장했는지를 담고 method card는 그 방법을 내가 쓰려 할 때 무엇을 정해야 하는지를 담는다. 같은 방법을 쓴 논문이 세 편 들어오면 세 논문 페이지가 하나의 method card를 가리키게 되고 그 시점에 이 방법에 대한 판단이 한자리에 모인다. Method card의 판단을 자기 실험 조건에 옮기면 연구 설계가 시작된다.

방법을 다루는 세 종류의 논문

같은 방법이 여러 종류의 논문에 등장하고 종류마다 읽는 목적이 다르다. 방법 논문은 기술 자체가 주인공이므로 목적, 성능, 벤치마크, 한계를 축으로 읽는다. 여기서 주의할 것은 성능 수치가 저자들이 고른 조건에서 나온 값이라는 점이다. 저자는 자기 방법이 잘 작동하는 조건을 알고 있고 그 조건을 벗어난 곳에서의 성능을 보고할 의무는 없다. 방법 논문에서 실제로 건져야 하는 것은 그 방법이 어떤 가정 위에 서 있는지, 가정이 깨지면 무엇이 먼저 무너지는지에 대한 서술이다. 방법을 사용한 연구 논문에서 방법은 도구로 쓰이고 관심의 초점은 그 도구가 어떤 생물학적 질문의 판정에 쓰였는지로 옮겨 간다. 같은 방법이라도 어떤 논문에서는 후보를 넓게 훑는 탐색에 쓰이고 다른 논문에서는 하나의 후보를 확정하는 검증에 쓰인다. 탐색에 쓸 때와 검증에 쓸 때 필요한 대조군과 반복 수는 같지 않다. 방법의 이름만 보고 두 용도를 구분하지 않으면, 탐색용 설계로 검증을 주장하거나 검증용 설계로 탐색을 하다가 자원을 소진하게 된다.

여러 방법을 같은 과제 위에 올려놓고 비교하는 벤치마킹 논문은 또 다른 읽기를 요구한다. 이 장이 실습 대상으로 삼는 것은 세포 foundation model들을 섭동 후 RNA-seq 예측이라는 과제로 평가한 연구다(Csendes 외, 2025). 벤치마킹 논문을 방법의 성능표로 읽으면 절반만 읽은 것이 된다. 어떤 과제를 골랐고 무엇을 기준선으로 두었고 어떤 지표로 우열을 갈랐는지가 본문의 실제 주장이기 때문이다. 평가 설계를 명시적으로 드러낸 논문은 판정 기준을 어떻게 만드는지 배울 수 있는 드문 교재다. 한 방법을 제대로 알려면 세 종류의 논문이 모두 필요하다. 방법 논문 한 편만 읽으면 그 방법이 어디까지 검증되었는지는 알지만 실제 연구에서 어떤 문제를 일으키는지 모른다. 사용 논문만 읽으면 방법을 도구로 다루는 감각은 생기지만 가정이 깨지는 지점을 모른다. 벤치마킹 논문만 읽으면 순위는 알지만 그 순위가 자기 과제에도 적용되는지 모른다. 그래서 한 방법에 대해 논문 서너 편을 묶어 읽고 그 묶음에서 하나의 method page를 만드는 편이, 논문마다

요약을 하나씩 만드는 것보다 나중에 훨씬 자주 쓰인다. 연구실이 이미 낸 논문과 지도교수가 지정한 참고 논문은 또 다른 자리에 있다. 연구실의 기존 논문에는 공식 프로토콜에 적히지 않은 것이 들어 있다. 어떤 대조군을 반드시 넣는지, 어느 지점에서 실험을 중단하는지, 무엇을 보고 재현되었다고 말하는지가 결과와 그림 안에 흔적으로 남아 있다. 지도교수가 건넨 논문은 자기 연구에서 선택해야 할 설계 요소가 무엇인지 알려 주는 목록이다. 그 목록을 만드는 일이 방법을 자기 것으로 만드는 첫 단계다.

방법을 꼬집어내는 열한 가지 질문

논문을 위키에 들이는 절차는 4장에서 다룬 ingest 여섯 단계를 그대로 따른다. 그 절차가 끝나면 원문과 독서 기록과 논문 페이지가 갖추어져 있으므로 방법을 추출하는 작업은 그 위에서 시작한다. 에이전트에게 다음 항목을 표로 정리하게 하면 방법의 뼈대가 한 화면에 들어온다. 항목을 고정하는 이유는 4장에서 독서 기록의 항목을 고정한 이유와 같다. 논문마다 다른 형식으로 정리하면 같은 방법을 다르게 구현한 논문들을 나란히 놓을 수 없다.

열한 항목 중에서 에이전트가 안정적으로 잘 채우는 것은 앞의 여섯 개다. 재료, 변수, 단계처럼 본문에 명시적으로 적혀 있는 정보이기 때문이다. 뒤로 갈수록 정확도가 떨어지고 마지막 항목이 가장 어렵다. 논문은 자기가 하지 않은 것을 적지 않으므로 증명하지 못한 범위는 본문 어디에도 문장으로 존재하지 않는다. 그 자리는 설계를 읽고 추론해야 채워지고 추론을 그럴듯한 문장으로 대신하면 나중에 자기 계획의 근거가 되어 버린다. 표가 만들어지면 원문으로 돌아가 검증하는데 Methods 절만으로는 부족하다. 대조군이 실제로 몇 개였는지는 그림 범례에 있고 데이터를 어떻게 걸렀는지는 보충 방법에 있으며 어느 비교가 저자들에게 결정적이었는지는 결과 서술의 순서에 드러난다. 본문 그림과 보충 자료까지 확인하지 않고 만든 방법 정리는 설계의 결모습에 그친다. 확인하는 데 걸리는 한 시간이 잘못된 전제로 석 달을 쓰는 일을 막는다. 방법 논문 한 편을 골라 method page 하나와 한 쪽짜리 설계안 하나를 만드는 순서는 여기서 시작한다. 15장에서 만든 읽기 대기열의 앞자리에 있는 논문이면 왜 읽기로 했는지가 이미 한 줄로 붙어 있으므로 고르기 쉽고 자기 분야의 방법 논문으로 바꾸어도 절차는 같다.

1. 논문을 4장의 절차로 위키에 들인다. 이미 들어와 있다면 이 단계는 넘어간다
2. 에이전트에게 앞의 열한 항목을 표로 정리하게 한다
3. 표에서 대조군, 판정 기준, 원 논문이 증명하지 못한 범위 세 항목을 골라 원문의 그림과 보충 자료로 직접 확인한다
4. 확인한 것과 확인하지 못한 것을 표에 표시한다

대체 가능한 방법과 그 대가를 묻는 항목은 자주 건너뛰지만 계획을 가장 크게 바꾼다. 어떤 방법을 쓰기로 정한 뒤에 그 방법의 대안을 적으면 이미 내린 결정을 정당화하는 문장이 나오기 쉬우므로 표를 만드는 단계에서 미리 적어 두는 편이 낫다. 대안을 적을 때는 이름만 나열하지 않고 무엇을 얻고 무엇을 포기하는지를 같은 축으로 쓴다. 해상도를 얻고 처리량을 잃는지, 처리량을 얻고 인과의 방향을 잃는지처럼 무엇과 무엇을 맞바꾸는지가 드러나야 한다. 이 항목이 채워져 있으면 나중에 지도교수가 다른 방법을 제안했을 때, 검토하지 않아서 안 쓴 것인지 검토하고 안 쓴 것인지를 즉시 답할 수 있다. 논문에 없는 항목은 없다고 적는다. 빈칸을 그대로 두는 편이 그럴듯한 값으로 채우는 것보다 낫다는 원칙은 3장에서 세운 운영 원칙 중 없으면 없다고 말하는 규칙과 같은 것이다. 대조군의 근거가 논문에 적혀 있지 않다면 “논문에 명시 없음”이라고 남기고 그 빈칸이 자기 설계에서 반드시 채워야 할 결정 지점이 된다. 빈칸의 위치가 곧 자기가 정해야 할 것의 목록이라는 점에서, 빈칸은 이 작업의 산출물이다. 표가 빈칸 없이 깔끔하게 채워져 돌아왔다면 어딘가를 지어냈을 가능성을 먼저 본다.

Concept page와 method page의 구분

Concept page는 이 현상이나 개념이 무엇인지를 설명하고 method page는 그것을 어떻게 측정하고 무엇으로 판정하는지를 설명한다. 둘을 한 문서에 합치면 개념이 바뀔 때마다 방법 서술을 건드리게 되고 방법을 개선할 때마다 개념 설명이 흔들린다. 개념은 분야가 바뀌어야 바뀌고 방법은 장비와 알고리즘이 바뀔 때마다 바뀌므로 갱신 주기가 다른 내용을 갈라 두는 편이 유지에 유리하다. 한 개념을 여러 방법이 측정하고 한 방법이 여러 개념에 쓰이는 관계도 두 문서를 나누어야 표현된다. Concept page에 들어가는 것은 정의와 경계다. 이 개념이 무엇을 가리키고 무엇을 가리키지 않는지, 분야에서 어떤 이름들로 불리는지, 인접한 개념과 어디서 갈라지는지를 적는다. 같은 단어가 분야마다 다른 뜻으로 쓰이는 경우가 흔하므로 지금 위키에서 이 단어를 어느 뜻으로 쓰기로 했는지를 정해 적어 두는 것이 concept page의 실질적인 기능이다. 개념과 관련된 논쟁도 여기에 적는다. 아직 합의되지 않은 것을 합의된 것처럼 적으면, 그 개념을 전제로 세운 모든 계획이 한곳에서 흔들린다. Method page의 구조는 다음과 같이 잡는다.

Question and purpose
Inputs and prerequisites
Assumptions
Experimental or analytical design
Controls and comparisons
Stepwise procedure
Quality control
Outputs and interpretation
Failure modes
Alternatives and limitations
Supporting papers
Local decisions and open questions

절차는 열두 항목 중 하나에 불과하다. 다음에 이 방법을 다시 쓸 때 필요한 것은 전제 조건, 입력 자료가 갖추어야 할 요건, 흔히 발생하는 실패 방식, 판정 기준, 그리고 비용이다. 비용은 마지막 항목에 적는데 시료 수와 장비 시간, 계산 시간, 사람의 시간을 함께 적어야 나중에 다른 방법과 비교할 수 있다. 비용을 적지 않은 method page는 언제나 그 방법을 쓰는 쪽으로 판단을 밀어붙인다. 같은 방법을 여러 논문이 다르게 구현했다면 차이를 숨기지 않고 나란히 적는다. 구현이 갈리는 지점은 대체로 그 방법에서 가장 민감한 선택이 있는 곳이고 나중에 자기 실험이 재현되지 않을 때 먼저 의심해야 한다. 특정 논문의 수치와 조건을 보편적 규칙으로 옮겨 적는 것도 피한다. 어떤 논문이 특정 세포주에서 정한 처리 시간은 그 세포주와 그 목적에 묶인 값이다. 연구실 내부에서 최적화한 조건과 공개된 논문에서 온 조건은 같은 문서 안에서도 출처를 갈라 적어야, 나중에 외부에 공유할 때 무엇을 빼야 하는지 판단할 수 있다.

1. 검증한 표를 앞의 method page 구조로 옮긴다
2. 논문에 없어서 자기가 정해야 하는 항목을 Local decisions and open questions에 모은다

모든 방법에 method page를 만들 필요는 없다. 한 번 쓰고 말 방법, 이미 표준이 되어 매뉴얼이 있는 방법, 자기 연구에서 결정할 것이 거의 없는 방법은 논문 페이지의 링크만으로 충분하다. Method page를 만들 값어치가 있는 것은 앞으로 여러 번 쓸 방법, 구현이 논문마다 갈리는 방법, 그리고 실패했을 때 원인을 찾기 어려운 방법이다. 판단이 서지 않으면 같은 방법을 두 번째로 쓰게 될 때 만든다. 두 번째로 쓰는 시점에는 첫 번째에서 무엇이 문제였는지 알고 있으므로 실패 방식 항목이 실제 내용으로 채워진다. 근거로 삼은 연구용 위키는 wiki page 17,591건을 50개 카테고리로 나누어 두고 있는데 방법 문서는 논문 페이지와 같은 층에 섞이지 않고 개념과 종합을 담는 층에 놓인다. 논문 페이지는 한 편의 논문을 가리키고 method page는 여러 논문을 가로지르기 때문이다. 독서 기록의 항목 이름은 4장에서 정한 그대로 쓰고 method page는 여러 독서 기록에서 방법에 해당하는 부분만 모아 재구성한 문서다. 출처가 된 논문들은 Supporting papers에 모두 적어 어느 문장이 어느 논문에서 왔는지 되짚을 수 있게 한다.

실험 설계의 큰 그림

실험 계획은 다음 여덟 질문을 순서대로 고정하면서 만들어진다. 순서를 바꾸면 앞의 답이 뒤의 답에 끌려간다.

1. 연구 질문은 무엇인가.
2. 중심 가설과 경쟁 가설은 무엇인가.
3. 두 가설을 가장 잘 가르는 관측은 무엇인가.
4. 어떤 실험군과 대조군이 필요한가.

5. 생물학적 반복과 기술적 반복은 각각 왜 필요한가.
6. 어떤 편향을 무작위화, 눈가림, 배치 설계로 줄일 것인가.
7. 성공과 실패와 애매한 결과를 무엇으로 판정할 것인가.
8. 결과별로 다음 실험은 무엇인가.

첫 번째 질문은 쉬워 보이지만 대부분의 계획서에서 가장 넓게 쓰여 있다. 어떤 유전자가 어떤 과정에 관여하는지를 알고 싶다는 문장은 관심사에 머문다. 질문이 되려면 어느 대상에서, 어떤 조건에서, 무엇과 비교해 무엇을 보겠다는 범위가 붙어야 한다. 범위를 좁히는 일이 야심을 줄이는 것처럼 느껴져 미루게 되지만 좁히지 않으면 뒤의 일곱 질문에 답할 수 없다. 네 번째 질문의 대조군도 마찬가지로 질문의 범위가 정해져야 고를 수 있다. 무엇과 비교할지는 무엇을 주장하고 싶은지에 따라 정해지고 주장이 흐릿하면 대조군은 관행대로 정해진다. 두 번째와 세 번째 질문에서 계획이 판정력을 갖는지가 정해진다. 경쟁 가설이 없으면 어떤 결과든 중심 가설의 증거로 읽히고 실제로 대부분의 초안 계획서에 경쟁 가설 자리가 비어 있다. 경쟁 가설은 “같은 관측을 만들어 낼 수 있는 다른 설명”으로 써야 쓸모가 있다. 기술적 잡음, 배치 효과, 시료 준비 과정의 차이처럼 재미없는 설명이 가장 자주 참인 경쟁 가설이다. 세 번째 질문은 두 설명을 서로 다른 방향으로 밀어내는 관측이 무엇인지 묻는 것이고 그 관측이 없으면 실험을 아무리 잘해도 판정이 되지 않는다.

다섯 번째와 여섯 번째 질문은 결과의 신뢰도를 좌우한다. 생물학적 반복은 결론을 개체나 배양 단위로 일반화하기 위한 것이고 기술적 반복은 측정 과정의 변동을 잡기 위한 것인데, 둘을 섞어 세면 실제보다 강한 주장을 하게 된다. 편향을 줄이는 장치는 실험을 시작한 뒤에는 넣을 수 없다. 처리군과 대조군을 서로 다른 날에 서로 다른 배치로 처리했다면 분석 단계에서 어떤 보정을 해도 처리 효과와 배치 효과가 갈라지지 않는다. 무작위화와 배치 설계는 되돌릴 수 없는 시점의 문제다. 일곱 번째 질문의 판정 기준은 결과를 보기 전에 적어 두어야 기준으로 작동한다. 효과의 방향과 크기, 비교 대상, 반복 사이의 일관성, 그리고 애매하다고 판단할 구간까지 미리 적는다. 애매한 구간을 비워 두면 실제로 애매한 결과가 나왔을 때 그 결과가 지지 쪽이나 기각 쪽 중 그때의 기대에 가까운 쪽으로 끌려간다. 결과가 나온 뒤에 기준을 정하는 순간, 그 실험은 가설을 검증하는 힘을 잃고 기대를 확인하는 절차가 된다. 여덟 번째 질문은 계획을 실행 가능한 것으로 만든다. 가설이 지지되었을 때의 다음 실험만 적힌 계획서는 절반만 쓰인 계획서다. 아무 차이도 없는 결과가 나왔을 때 방법이 작동하지 않은 것인지 효과가 없는 것인지를 어떻게 가를지, 예상과 반대 방향의 결과가 나왔을 때 무엇을 먼저 의심할지를 함께 적는다. 실험 하나가 끝났는데 다음에 무엇을 할지 다시 회의를 열어야 한다면 계획이 아직 미완인 것이다. 결과별 분기를 적어 두면 실험이 끝나는 날 바로 다음 작업으로 넘어갈 수 있다. 이렇게 만든 설계안은 아젠다 쪽에 날짜와 함께 남기고 판정 기준 문단은 실행을 시작한 날짜를 적어 고정한다. 이후에 고칠 때는 덮어쓰지 않고 언제 왜 고쳤는지를 함께 남기는데 기준을 바꾼 기록이 남아 있는 것과 바꾼 흔적이 없는 것은 다른 문서다.

분석 설계와 실험 설계의 동형성

분석은 질문과 판정 기준을 정하는 순간에 시작된다. 계산 분석은 시료가 소모되지 않고 실패해도 다시 돌리면 되기 때문에 계획 없이 시작하기 쉽다. 여러 번 돌릴 수 있기 때문에 사전 계획이 없으면 가장 마음에 드는 결과가 나온 설정을 사후에 정당화하게 된다. 실험에서는 시료가 없어서 못 하는 일을, 분석에서는 마음만 먹으면 스무 번도 할 수 있다는 점이 계산 분석의 고유한 위험이다. 분석 계획에는 시료 포함과 제외 기준, 표현형의 정의, 전처리 절차, 함께 넣을 공변량, 데이터를 나누는 방식, 비교의 기준선, 그리고 민감도 분석을 미리 적는다. 표현형 정의가 특히 자주 비어 있다. 같은 코호트에서도 어떤 기준으로 사례를 정의하느냐에 따라 표본 수와 효과 크기가 함께 움직이고 정의를 몇 번 바꾸다 보면 유의해지는 정의를 고르게 된다. 기준선도 계획서에 이름으로 적어야 한다. 무엇과 비교해서 좋아졌는지가 정해지지 않은 성능 수치는 해석할 수 없고 기준선을 나중에 고르면 이길 수 있는 기준선을 고르게 된다. 가설 없이 해 볼 만한 분석을 모아 놓은 목록은 계획처럼 보이지만 어떤 결과가 나와도 다음 행동이 정해지지 않으므로 계획이 되지 못한다. 목록의 각 줄 옆에 그 분석이 어떤 가설을 가르는지 적어 보면 대부분의 줄이 지워진다.

같은 데이터에 여러 방법을 시도하는 일에서 갈리는 것은 그 결과를 어떻게 보고하느냐다. 사전 계획에 따라 여러 방법을 비교하고 전부를 보고하는 것과, 여러 방법을 돌려 본 뒤 가장 좋은 결과 하나만 남기는 것은 겉으로 같은 표를 만들지만 의미가 정반대다. 계획서에 시도할 방법의 목록과 선택 규칙을 미리 적어 두면 둘을 구분할 수 있다. 선택 규칙은 “사전에 정한 지표에서 정한 차이 이상으로 앞선 것”처럼 결과를 보기 전에 적용할 수 있는 형태여야 한다. 모델 성능과 생물학적 결론을 같은 문장에 넣지 않는 것도 분석 계획에 명시한다. 예측 정확도가 올라갔다는 것은 예측이 잘된다는 뜻이지만 보증한다. 성능이 높은 모델이 무엇을 근거로 그 예측을 했는지는 별도의 분석이 필요하고 그 분석을 계획에 넣지 않았다면 논문에서도 그 주장을 하지 않아야 한다. 이 계획에서 어떤 결론까지 허용되는지를 계획서에 한 줄로 적어 두면, 나중에 결과가 좋을 때 주장을 늘리고 싶은 압력에 대응할 기준이 생긴다. 허용 범위를 넘는 주장을 하고 싶어졌다면 그때 계획을 다시 세운다.

민감도 분석은 계획서에서 가장 자주 생략되고 가장 자주 논문 심사에서 요구된다. 결론이 전처리 방식 하나, 제외 기준 하나, 공변량 하나에 얼마나 의존하는지를 미리 정해 두면 결과를 얻은 뒤에 급하게 만들지 않아도 된다. 어떤 선택을 흔들어

볼지는 계획 단계에서 이미 알고 있다. 근거 없이 관행으로 정한 항목, 논문마다 다르게 하는 항목, 정하면서 망설였던 항목이 그것이다. 망설였던 자리를 그대로 민감도 분석의 목록으로 옮기면, 나중에 결론이 그 선택에 크게 흔들릴 때 스스로 먼저 발견하게 된다. 재현 경로도 계획의 일부다. 실행한 코드, 입력 파일, 중간 산출물, 최종 결과가 서로를 가리키도록 남긴다. 사전에 정하지 않으면 중간 산출물이 임시 폴더에 흩어지고 석 달 뒤에 그림 하나를 다시 그리려 할 때 어느 입력에서 나온 것인지 알 수 없게 된다. 벤치마킹 논문은 그래서 분석 설계의 교재가 된다. 여러 방법을 공정하게 비교하려면 입력과 전처리와 지표를 모두 고정해야 하므로(Csendes 외, 2025), 그런 논문의 설계를 따라 읽으면 자기 분석에서 무엇을 고정해야 하는지가 드러난다.

재현 경로를 계획의 일부로 만들려면 파일이 놓일 자리가 먼저 정해져 있어야 한다. 분석 작업 공간 하나를 열어 두고 그 안에서 프로젝트마다 폴더를 하나씩 만드는 배치가 오래 버틴다. 최상위에는 규칙 파일 하나와 위키 폴더 하나, 그리고 작업 단위가 되는 프로젝트 폴더들이 놓인다.

```
Projects/
  AGENTS.md
  wiki/
    index.md
    conventions.md
    references.md          registry
    projects.md
    lint.md
    log.md
    project-guides/
    project-logs/          ( )
  { }/                    . 00_inputs → 01_{ } → 02_results
```

폴더 이름은 예시다. 최상위 폴더의 이름을 자기 말로 바꾸어도 되고 프로젝트 폴더에는 각자의 과제 이름을 쓴다. 배치에서 실제로 지켜야 하는 것은 두 가지다. 하나는 작업의 단계를 폴더 이름으로 드러내는 것이고 다른 하나는 로그를 작업 폴더 밖으로 뽑아내는 것이다. 받은 원본은 00_inputs에 그대로 두고 덮어쓰지 않으며 파생본은 새 이름으로 만든다. 지금 내놓는 결과물은 프로젝트 폴더의 루트에 두어 무엇이 최신인지 몇 초 안에 보이게 하고 새 버전이 나오면 직전 버전을 같은 자리의 old/로 밀어낸다. 중간 산출물은 old/tmp/나 작업 공간 밖의 임시 폴더에 두고 프로젝트 폴더의 루트에는 임시 폴더를 만들지 않는다. 여러 프로젝트가 함께 쓰는 공용 자료는 정보 한 곳에서 경로로 읽고 프로젝트 폴더로 복사하지 않으며 그 정보의 위치는 references.md가 소유한다. 심볼릭 링크도 쓰지 않는데 동기화 서비스가 링크를 사본으로 평탄화해 결국 사본이 두 벌 남기 때문이다. 진행 메모와 변경 기록은 wiki/project-logs/ 아래에 프로젝트마다 한 파일씩 두고 덧붙이기만 한다. 로그 항목의 형식도 고정해서 무엇이 이 작업을 촉발했는지, 그때 원본이 어떤 상태였는지, 무엇을 실행했는지, 무엇을 확인했는지, 무엇을 발견했는지, 무엇을 결정했는지, 무엇이 남았는지, 어떤 산출물이 나왔는지의 여덟 항목을 같은 순서로 적는다. 확인 결과에는 확인됨, 불일치, 미확인, 보류, 재실행필요의 다섯 라벨을 쓰는데 확인하지 못한 것을 이상 없음으로 적지 않기 위한 장치다. 위키의 규칙 문서에는 일반형 문장만 쓰고 특정 날짜와 세션과 파일 이름은 로그 쪽에 남긴다.

따라해보기

아래 블록은 AI 도구의 채팅창에 그대로 붙여 넣는 프롬프트다. 분석 작업 공간을 처음 세울 때 쓰는 프롬프트다. 빈 폴더를 하나 만들어 그 안에서 에이전트를 실행한 뒤 아래를 그대로 붙여 넣고 꺾쇠로 표시한 자리는 자기 상황에 맞게 바꾼다.

```
1. AGENTS.md
   - < > .
   - . wiki/project-logs/ .
   - AGENTS.md wiki/index.md .
2. wiki/ index.md, conventions.md, references.md, projects.md, lint.md, log.md project-guides/ pr
3. conventions.md . 00_inputs/ , 01_{ }/ , ,
4. < > 00_inputs/ .
5. projects.md . , , .
6. . .
```


계획서가 만들어지는 순환

계획은 한 번에 완성되지 않고 다음 순환을 돈다.

↓ concept, method,

↓

↓ , , ,

↓

→

순환에서 예비 실험과 예행 분석을 빼놓지 않는 것이 중요하다. 예비 단계의 목적은 계획이 실행 가능한지를 확인하는 것이다. 시료가 예상한 양만큼 나오는지, 측정값이 읽을 수 있는 범위에 들어오는지, 데이터의 형식이 파이프라인에 그대로 들어가는지처럼 본 실험에서 발견하면 늦는 것들을 여기서 걸러 낸다. 계산 분석에서는 실제 데이터 대신 무작위로 뒤섞은 자료나 가상의 자료로 분석 전체를 한 번 돌려 보는 방식이 같은 역할을 한다. 예행에서 유의한 결과가 나온다면 분석 절차 어딘가에 결과를 만들어 내는 단계가 있다는 뜻이므로 본 분석을 돌리기 전에 그것을 찾아야 한다. 순환의 각 단계에서 중요한 선택 옆에는 근거가 된 논문이나 method page를 링크로 붙인다. 링크를 붙이는 일이 번거로워 보이지만 계획을 지도교수나 심사자에게 설명할 때 되돌아가야 할 곳이 바로 그 링크들이다. 근거 없이 적힌 선택은 시간이 지나면 관행이 되고 관행이 된 선택은 왜 그렇게 하는지 아무도 모르는 채로 연구실에 남는다. 어떤 선택에 붙일 근거가 없다면 그 선택은 아직 가정이고 가정이라고 표시해 두면 나중에 결과가 이상할 때 먼저 확인할 목록이 된다. 근거와 가정을 갈라 표시하는 일에 드는 시간은 계획서를 쓰는 시간의 일부다.

1. 자기 연구 질문에 맞춘 설계안을 한 쪽으로 쓴다
2. 가설, 경쟁 가설, 두 가설을 가르는 관측, 대조군, 분석, 판정 기준을 반드시 넣는다
3. 에이전트에게 예상한 결과가 그대로 나와도 가설이 틀렸을 수 있는 경로를 세 가지 찾게 한다
4. 그중 하나를 반영해 설계안을 고친다

설계안이 제 몫을 하는지는 판정 기준 문단을 가린 채 읽었을 때 다음 세션의 에이전트가 “이 실험에서 어떤 결과가 나오면 가설을 기각하는가”에 답할 수 있는지로 정해진다. 경쟁 가설 자리와 판정 기준 자리가 모두 채워져 있고 각 선택 옆에 근거 문서가 링크되어 있는지로 판단한다. 에이전트에게는 계획을 공격해 달라고 요청한다. 약한 전제, 빠진 대조군, 같은 결과를 만들어 낼 수 있는 다른 설명을 찾게 한다. 요청은 구체적일수록 잘 작동한다. “이 계획의 문제를 찾아 줘”는 일반론을 돌려주지만 “이 계획에서 예상한 결과가 그대로 나와도 중심 가설이 틀렸을 수 있는 경로를 세 가지 찾아 줘”는 실제로 검토할 만한 목록을 돌려준다. 위키에 method page가 쌓여 있으면 에이전트가 과거에 정리해 둔 실패 방식과 지금 계획을 대조할 수 있으므로 반론의 질이 위키의 두께에 비례해 올라간다. 책임의 경계는 이 순환 안에서 분명하다. 에이전트는 관련 문서를 찾고 여러 논문의 설계를 나란히 비교하고 항목을 구조화하고 반론을 제기한다. 학생은 원문을 직접 확인하고 어떤 질문을 할지 고르고 실험과 분석의 실행에 책임을 지고 결론을 어느 강도로 말할지 정하고 무엇을 밖으로 내보낼지 결정한다. 지도교수와 공동연구자는 전문적 판단과 자원의 제약을 함께 정하지만 학생이 미리 정리해야 할 부분을 대신하지는 않는다. 에이전트가 제안한 프로토콜이나 분석 코드를 검증 없이 실행하는 일은 이 경계를 지우는 가장 흔한 방식이다. 결과가 예상과 다르게 나왔을 때 계획서를 조용히 덮지 않는다. 어떤 가정이 깨졌는지를 method page의 실패 방식 항목에 되돌려 적는 것이 이 순환의 마지막 화살표다. 다음에 같은 방법을 쓰는 사람은 그 한 줄 때문에 같은 실패를 반복하지 않는다. 실패한 실험의 기록이 남지 않는 연구실에서는 같은 실패가 사람이 바뀔 때마다 되풀이되고 그 비용은 누구의 논문에도 나타나지 않는다.

실험과 계산 분석의 갈림

후보 조절 요소의 기능을 검증하는 실험에서 먼저 정할 것은 현재 주장의 범위와 후보를 고른 근거다. 어떤 신호를 보고 후보로 올렸는지에 따라 검증해야 할 것이 달라지기 때문이다. 다음으로 검증 방식이 답하는 질문이 서로 다르다는 점을 확인한다. 리포터 방식은 잘라 낸 서열이 활성을 가질 수 있는지를 묻고 내인성 자리를 억제하는 방식은 원래 위치의 서열이 실제로 발현에 기여하는지를 묻는다. 두 질문의 답이 같을 수 있고 같았을 때 어느 쪽을 결론으로 삼을지는 계획 단계에서 정해야 한다. 젖은 실험 계획에서 남는 결정은 대체로 물리적 제약에 묶여 있다. 어떤 벡터와 프로모터를 쓸지, 서열을 얼마나 길게 넣을지, 무엇을 읽을지, 한 번에 몇 개의 후보를 처리할 수 있는지가 서로 맞물린다. 후보 수를 늘리면 후보당

반복이 줄고 반복을 늘리면 처리할 수 있는 후보가 준다. 양성 대조군은 이 실험계에서 신호가 나오는지 확인하고 음성 대조군은 나온 신호가 서열 때문인지를 확인한다. 둘 중 하나가 빠지면 결과가 나와도 해석되지 않고 시료를 이미 다 쓴 뒤에는 되돌릴 수 없다. 결과별 후속 검증까지 계획에 넣으면 실험 하나가 다음 실험을 부른다. 두 검증 방식의 결과가 같은 방향이면 후보를 확정하고 다음 층위의 질문으로 넘어가지만 갈리면 어느 쪽을 먼저 의심할지를 미리 정해 두어야 한다. 잘라 낸 서열에서는 활성이 나오는데 원래 자리에서는 나오지 않는 경우와 그 반대의 경우는 서로 다른 해석을 부르고 각각에 대해 확인할 실험도 다르다. 두 경우의 다음 단계를 계획서에 미리 적어 두면 결과가 갈렸을 때 당황하지 않는다. 갈린 결과를 실패로 부르지 않는 것도 계획서의 역할인데, 서로 다른 질문에 서로 다른 답이 나온 것은 정보이기 때문이다.

여러 코호트의 전장 유전체 데이터를 분석하는 사례에서는 결정의 성격이 다르다. 이미 확보한 데이터와 앞으로 들어올 데이터를 먼저 구분하고 각각이 어느 참조 유전체와 어느 파이프라인 버전으로 어느 단계까지 처리되어 있는지 적는다. 코호트마다 처리 이력이 다르면 같은 방식으로 다시 처리할 수 있는지가 첫 번째 결정 사항이 되고 그 결정에 따라 합동 변이 검출이 가능한지와 언제 가능한지가 정해진다. 품질 검사와 제외 기준도 코호트별로 다르게 적용할지 통일할지를 미리 정한다. 마지막으로 확정된 사항과 아직 확인해야 할 사항을 갈라 적어 두면, 미팅에서 논의할 것이 무엇인지가 그대로 드러난다. 두 사례에서 같은 것은 판정 기준을 결과보다 먼저 적는다는 원칙이고 다른 것은 되돌리는 비용이다. 젖은 실험은 시료와 시간이 소모되므로 잘못 설계된 실험을 다시 하려면 처음부터 다시 해야 한다. 계산 분석은 다시 돌릴 수 있지만 다시 돌릴 수 있다는 사실 때문에 사후에 설정을 고르는 유혹이 생긴다. 앞의 위험은 자원을 잃는 것이고 뒤의 위험은 결론을 잃는 것인데, 뒤의 손실은 겉으로 드러나지 않기 때문에 더 늦게 발견된다. 같은 계획서 양식을 쓰되 젖은 실험에서는 순서와 배치를, 계산 분석에서는 사전 고정과 보고 범위를 더 꼼꼼히 적는다.

계획에서 미팅 자료로의 전환

완성된 계획서를 그대로 들고 미팅에 들어가면 대체로 앞부분에서 시간이 다 간다. 계획서는 실행하는 사람을 위한 문서이고 미팅 자료는 결정하는 사람을 위한 문서라서, 담아야 할 것이 다르다. 실행에 필요한 세부 조건은 계획서에 그대로 두고 미팅에는 결정이 필요한 것만 가져간다. 분량은 네 장에서 여섯 장 사이로 잡는다. 그보다 길면 듣는 쪽이 어디에서 판단해야 하는지 알기 어렵고 그보다 짧으면 판단의 근거가 빠진다. 가져갈 내용은 다음 순서로 정리한다. 연구 질문과 현재 결론이 어디까지 말할 수 있는지, 그 질문에 관련된 근거와 그로부터 나온 설계 선택, 실험이나 분석의 흐름과 대조군, 예상되는 결과별 해석, 아직 정해지지 않은 사항과 필요한 결정, 그리고 결정이 나면 바로 착수할 다음 작업이다. 다섯 번째 항목이 미팅을 여는 실제 이유이므로 가장 구체적으로 적는다. 무엇을 결정해 달라는 요청 없이 계획만 설명하면 회의는 감상으로 끝나고 같은 계획을 들고 다음 달에 다시 모이게 된다. 네 번째 항목의 예상 결과별 해석도 빼놓지 않는데 결과가 나오기 전에 해석을 합의해 두면 결과가 나온 뒤에 해석을 놓고 다투지 않아도 된다.

가져가지 않을 것도 정해 두는 편이 좋다. 시약의 카탈로그 번호, 파이프라인의 실행 옵션, 코드의 구조는 실행하는 사람에게 필요한 정보다. 세부를 넣으면 논의가 그 세부로 끌려가고 정작 결정이 필요한 항목은 시간이 없어 다음으로 미뤄진다. 다만 질문이 나올 만한 세부는 계획서의 어느 부분에 있는지 링크로 걸어 둔다. 미팅 자료가 계획서로 가는 입구가 되어야, 논의가 깊어졌을 때 바로 근거를 열어 볼 수 있다. 한 장짜리 결정표를 함께 준비하면 논의가 흩어지지 않는다. 각 행에 결정할 항목, 선택지, 각 선택지의 근거와 대가, 그리고 지금 기울어져 있는 쪽을 적는다. 기울어져 있는 쪽을 미리 적는 것이 중요한데, 아무 의견 없이 선택지만 내리면 결정이 지도교수에게 통째로 넘어가고 학생은 자기 판단을 훈련할 기회를 잃는다. 결정표는 미팅이 끝난 뒤 결정된 내용과 그 이유를 채워 넣어 그대로 기록으로 남긴다.

18장. 공동 연구 미팅을 준비하는 법

미팅에서는 서로 다른 배경과 정보를 가진 사람들이 같은 질문과 같은 근거를 놓고 판단해 다음 행동을 정한다. 결정을 남기는 힘은 미팅 전에 공유된 문서에서 나온다. 사전 자료 한 장은 30분의 배경 설명을 10분의 결정으로 바꾸고 그 한 장을 만드는 책임은 자료를 들고 들어가는 사람에게 있다. 어떤 대학원생이 석 달치 분석 결과를 들고 미팅에 들어간다. 슬라이드는 스무 장이 넘고 각 장에는 실제로 수행한 작업이 시간순으로 들어 있다. 데이터를 어떻게 받았고 어떤 필터를 걸었고 중간에 파이프라인을 한 번 바꿨고 그래서 다시 돌렸다는 이야기가 앞의 열 장을 채운다. 한 시간짜리 미팅에서 절반이 그 열 장에 쓰이고 정작 판단이 필요한 뒤쪽 결과는 넘기듯 지나간다. 미팅은 “다음에 한 번 더 보자”로 끝나고 참석자들은 서로 다른 것을 기억한 채 방을 나선다. 학생은 성실했고 결과도 있었다. 없었던 것은 무엇을 결정해 달라는 문장 하나와, 그 결정에 필요한 근거만 골라 놓은 종이 한 장이었다.

미팅의 기능

말로만 하는 설명은 듣는 사람에게 두 가지 일을 동시에 시킨다. 하나는 내용을 이해하는 일이고 다른 하나는 발표자의 머릿속 구조를 추측하는 일이다. 뒤의 일이 앞의 일보다 훨씬 힘들다. 어떤 결과가 어떤 질문에 대한 답인지, 지금 보고 있는 그림이 앞의 그림을 보강하는 것인지 반박하는 것인지, 이 수치가 확정된 것인지 아직 검토 중인 것인지를 청자는 발표를 따라가면서 스스로 조립해야 한다. 조립에 쓰이는 인지 자원은 판단에 쓸 수 있었던 자원이고 그래서 설명이 길수록 결정은 알아진다. 참석자들이 같은 배경 지식을 갖고 있지 않다는 점이 문제를 키운다. 실험을 하는 사람, 계산을 하는 사람, 임상 자료를 보는 사람이 한자리에 있으면 각자가 당연하다고 여기는 것이 다르다. 한쪽에는 설명할 필요가 없는 전처리 단계가 다른 쪽에는 결과를 좌우하는 미지의 상자다. 발표자가 자기 기준으로 “이건 다들 아실 테니 넘어가겠습니다”라고 말하는 순간, 모르는 사람은 질문할 기회를 잃고 남은 시간 내내 반쯤 이해한 채로 앉아 있게 된다. 배경이 다른 사람들을 같은 출발점에 세우는 일은 미팅장에서 즉흥으로 되지 않는다. 수행한 일을 전부 보여주는 방식과 결정에 필요한 근거만 배열하는 방식은 결과물의 모양이 다르다. 앞의 방식은 시간순으로 흐르고 마지막에 결론이 나오며 중간에 끊기면 아무것도 남지 않는다. 뒤의 방식은 결론의 범위를 먼저 놓고 그것을 지지하는 근거를 붙이며 중간에 끊어도 앞부분만으로 판단이 가능하다. 실제 미팅은 거의 언제나 중간에 끊긴다. 예정보다 늦게 시작하고 앞의 논의가 길어지고 참석자 한 명이 먼저 나가야 한다.

축	진행 보고형	결정 요청형
배열 순서	작업 시간순	결론과 그 근거 순
앞부분에 오는 것	데이터 수집과 처리 과정	결정 항목과 현재 판단
중간에 끊겼을 때	결론에 도달하지 못함	이미 판단 가능
청자가 하는 일	구조를 추측하며 따라가기	제시된 선택지 검토
남는 것	진행 상황에 대한 인상	결과와 다음 작업

세 종류의 미팅에 같은 원칙이 적용되지만 강조점은 다르다. 내부 연구실 미팅에서는 배경이 이미 공유되어 있으므로 사전 자료가 짧아도 되지만 대신 무엇을 결정받으려는지가 더 분명해야 한다. 지도교수가 이미 아는 내용을 다시 설명하면 남은 시간이 사라진다. 공동연구 미팅에서는 배경 공유가 절반의 일이고 상대 분야의 용어와 이쪽 분야의 용어가 같은 대상을 다르게 부르는 지점을 미리 맞춰야 한다. 외부 발표에서는 결정 요청이 없는 대신 결론의 범위를 어디까지 말할 수 있는지가 더 엄격해지고 확정되지 않은 것을 확정된 것처럼 말하지 않는 규율이 그때 평판을 만든다.

미팅 전에 정하는 결정 항목

미팅 일정을 잡기 전에 여섯 문장을 쓴다. 이번 미팅이 끝날 때 결정되어 있어야 하는 것은 무엇인가. 상대가 그 결정을 하려면 어떤 배경과 자료가 필요한가. 지금 확정된 사실은 무엇인가. 아직 확인되지 않았거나 서로 다르게 이해하고 있는 것은 무엇인가. 상대의 전문성이 꼭 필요한 질문은 무엇인가. 결정이 나면 바로 시작할 다음 작업은 무엇인가. 각각 한 문장이면 충분하고 한 문장으로 쓸 수 없다면 아직 미팅을 잡을 준비가 되지 않은 것이다. 17장에서 계획서와 함께 만든 결정표가 있으면 첫 번째 문장과 다섯 번째 문장이 그 표에서 그대로 나온다. 여섯 문장 중 첫 번째가 비면 미팅을 잡지 않는 편이 낫다. 결정할 것이 없는데 모이면 그 자리는 진행 상황을 확인하는 자리가 되고 진행 상황은 문서 한 통으로 더 정확하게 전달된다. 결정할 것이 없다는 판단이 부담스러워 역지로 항목을 만들면 미팅은 열리지만 결정은 나지 않고 다음번에 같은 일이 반복된다. 반대로 결정 항목이 다섯 개, 여섯 개로 늘어나면 어느 것도 충분히 검토되지 않는다. 한 번의 미팅에서 실제로 결정될 수 있는 것은 하나에서 셋 사이이고 그 이상은 목록일 뿐이다. 실제로 예정된 미팅 하나를 골라 준비부터 정리까지 한 번 수행해 보면 여섯 문장이 어디까지 좁혀지는지 알 수 있고 아직 미팅이 없다면 다음 연구실 미팅을 대상으로 삼는다.

1. 미팅 전 여섯 문장을 쓴다
2. 결정 항목이 하나에서 셋 사이가 되도록 좁힌다

네 번째 문장이 가장 쓰기 어렵고 가장 값이 나간다. 아직 확인하지 못한 것을 문서에 적는 일은 준비가 부족해 보일 위험을 감수하는 일이기 때문이다. 그런데 미확정 항목을 감추면 상대는 그것을 확정된 것으로 간주하고 판단하며 나중에 어긋났을 때 되돌리는 비용이 훨씬 커진다. 미확정을 미확정이라고 적은 문서는 읽는 사람에게 어디까지 믿어도 되는지를 알려 준다. 확정된 사실, 현재의 해석, 아직 확인할 항목을 같은 문장 안에 섞지 않는 규율이 사전 정리의 절반을 차지한다. 여섯 문장이 정리되면 결정 안건 목록으로 옮긴다. 안건 목록에는 결정할 항목만 모으고 항목마다 세 가지를 적는다. 결정의 형태가 무엇인지, 그 결정을 하려면 어떤 근거를 봐야 하는지, 이번에 결정하지 않으면 무엇이 멈추는지다. 세 번째를 적어 보면 안건의 우선순위가 저절로 갈린다. 지금 결정하지 않아도 아무것도 멈추지 않는 항목은 공유 사항이므로 문서로 알려

주면 된다. 안건 목록을 미팅 시작 전에 참석자가 보고 있으면 논의가 흩어졌을 때 되돌아올 자리가 생긴다. 여섯 문장을 쓰는 일은 학생의 몫이다. 지도교수와 공동연구자는 전문적 판단을 제공하고 자원의 한계를 정하지만 무엇을 결정해야 하는지 정리하는 일까지 대신하지는 않는다. 정리되지 않은 자료를 들고 가서 “어떻게 하면 좋을까요”라고 물으면 상대는 정리부터 시작해야 하고 그 정리는 상대가 이쪽 데이터를 잘 모르는 상태에서 이루어지므로 부정확하다. 스스로 확인할 수 있는 범위를 먼저 확인하고 확인해도 답이 나오지 않는 지점을 정확히 지목해 묻는 사람이 좋은 공동연구자다. 질문을 넘기는 것과 질문을 좁혀서 건네는 것은 상대가 쓰는 시간의 양이 다르다.

LLM-Wiki로 pre-read를 만드는 법

사전 자료를 만드는 시간의 대부분은 이미 위키에 있는 것 중에서 무엇을 고를지 정하는 데 든다. 고르는 기준은 하나다. 이번 미팅에서 요청할 결정과 직접 연결되는가. 이 기준으로 걸러 보면 지난 석 달 동안 만든 문서의 대부분이 사전 자료에 들어가지 않는다. 잘 만든 논문 정리, 공들여 그린 중간 그림, 여러 번 고친 방법 설명이 모두 훌륭하지만 이번 결정과 무관하다면 자리를 차지할 뿐이다. 사전 자료의 품질은 무엇을 뺐는가로 정해진다. 새로 시작하는 공동연구라면 상대에게 미리 논문을 요청하는 일이 첫 단계다. 상대 분야에서 무엇이 진입점인지는 그 분야 사람이 가장 잘 알고 두세 편만 받아도 용어와 쟁점의 지도가 잡힌다. 받은 논문은 미팅 전에 4장의 절차대로 위키에 올린다. 첫 미팅에서 상대가 자기 분야를 처음부터 설명하는 데 시간을 쓰지 않아도 되면, 그 시간이 그대로 질문과 역할 분담에 쓰인다. 논문을 미리 읽었다는 사실보다 읽고 나서 남은 질문을 가져갔다는 사실이 첫 미팅의 밀도를 바꾼다.

위키에서 근거를 모을 때는 검색어를 완전한 질문으로 던진다. 9장에서 다룬 방식이고 미팅 준비에도 그대로 쓰인다. 관련된 overview와 concept 문서, 방법을 정리한 문서, 개별 논문의 source note를 훑어 이번 결정과 맞닿은 대목을 표시한다. 에이전트는 후보를 모으고 중복을 걸러 내고 형식을 맞추는 일을 빠르게 해내지만 다섯 개의 후보 중 어느 둘이 이번 결정에 실제로 걸리는지는 판단해야 한다. 그 판단이 사전 자료를 만드는 사람의 몫이고 위임하면 자료가 길어지면서 초점이 사라진다. 근거를 옮길 때는 원문으로 되돌아가는 경로를 함께 남긴다. 사전 자료에 적힌 한 문장짜리 판단이 어느 논문의 어느 결과에서 나왔는지 표시되어 있으면, 미팅 중에 이견이 생겼을 때 그때 확인할 수 있다. 경로가 없으면 “제가 어디선가 봤는데”로 시작하는 대화가 이어지고 그 논의는 대개 결론 없이 끝난다. 공개된 문헌 지식과 프로젝트 내부의 미공개 결과도 서식으로 구분한다. 둘을 섞어 놓으면 나중에 이 문서의 어느 부분을 외부에 공유해도 되는지 다시 판단해야 하고 급할 때 그 판단은 자주 틀린다.

사전 자료가 실패하는 가장 흔한 모양은 논문 요약 모음이 되는 것이다. 위키에 잘 정리된 문서가 여러 개 있으면 이어 붙이는 편이 빠르고 붙여 놓으면 분량도 그럴듯해 보인다. 그런데 요약은 각 논문 안에서 완결된 글이라 이번 결정과 만나는 지점이 표시되어 있지 않고 받은 사람은 다섯 편의 요약을 읽으면서 어느 대목이 오늘 안건과 관련되는지 스스로 찾아야 한다. 찾는 데 드는 시간이 배경 설명을 듣는 시간과 크게 다르지 않으므로 사전 공유의 효과가 사라진다. 논문에서 가져올 것은 이번 판단을 지지하거나 흐드는 결과 하나와, 그 결과가 어떤 조건에서 나왔는지에 대한 한 문장이다. 나머지는 링크로 두고 필요한 사람이 내려가게 한다. 공유 시점은 최소 하루 전, 늦어도 열두 시간 전이다. 미팅 시작과 함께 배포한 문서는 유인물이어서 참석자는 발표를 들으면서 동시에 읽을 수 없다. 하루의 여유가 있으면 상대는 자기 전문 영역에서 걸리는 부분을 미리 확인하고 반론이나 대안을 준비해 온다. 준비된 반론이 미팅의 목적이다. 사전 공유의 실제 효과는 상대가 무엇을 물어볼지 정하고 온다는 데 있다.

1. 위키에서 이번 결정과 직접 연결된 근거만 골라 여덟 항목 골격에 맞춘 사전 자료 한 장을 만든다
2. 근거마다 원문으로 가는 경로를 남긴다
3. 미팅 최소 열두 시간 전에 사전 자료를 공유한다

같은 페이지의 표준 구조

“모두가 같은 페이지에 있다”는 말은 질문, 근거, 불확실성, 결정 항목이 하나의 문서에 함께 적혀 있고 참석자가 그 문서를 읽은 상태를 뜻한다. 문서가 없으면 각자의 머릿속에 서로 다른 페이지가 있고 미팅은 그 페이지들을 맞추는 데 시간을 쓴다. 아래 여덟 항목은 그 한 장의 문서가 갖추는 최소 골격이다. 항목의 순서가 곧 읽는 사람이 판단을 만들어 가는 순서다.

1. 과학적 질문
2. 왜 지금 이 질문인가
3. 현재 근거와 데이터 현황
4. 지금 말할 수 있는 결론
5. 아직 불확실하거나 이견이 있는 것

6. 선택지와 각각의 대가
7. 이번 미팅에서 요청하는 결정
8. 다음 작업, 책임자, 기한

1번과 2번은 합쳐서 대여섯 문장을 넘지 않는다. 질문을 길게 쓰면 서론이 되어 버리고 서론은 사전 자료에서 가장 먼저 잘려 나가야 할 부분이다. 2번을 따로 두는 이유는 같은 질문이라도 지금 다루어야 하는 이유가 매년 다르기 때문이다. 새 데이터가 들어왔거나 다른 그룹이 비슷한 결과를 냈거나 자원 배분의 시한이 다가왔다는 사정이 결정의 성격을 바꾼다. 지금이어야 하는 이유가 없으면 그 결정은 미뤄도 되는 결정이다. 3번의 데이터 현황은 표로 정리한다. 표본 수, 대조군 구성, 현재까지 진행된 처리 단계, 사용한 파이프라인과 참조 자료의 버전, 아직 들어오지 않은 항목을 한눈에 볼 수 있게 배열한다. 여기서 지켜야 할 것은 해석이 필요한 자료와 단순 현황을 한 장에 섞지 않는 일이다. 현황 표에 “이 결과는 A를 시사한다” 같은 문장이 끼어들면 읽는 사람이 어디까지가 사실이고 어디부터가 판단인지 구분하지 못한다. 현황은 현황대로 두고 해석은 4번과 5번으로 보낸다.

4번과 5번을 나누는 일이 이 구조의 중심이다. 4번에는 지금의 자료로 말할 수 있는 것을 조건과 함께 적는다. 조건을 빼고 결론만 적으면 그 결론은 실제보다 강해지고 강해진 결론은 이후 논의 전체를 잘못된 자리에서 시작하게 만든다. 5번에는 확인되지 않은 것, 자료가 부족한 것, 참석자들 사이에 이견이 있는 것을 나눠 적는다. 이견이 있다는 사실을 문서에 적어 두면 미팅에서 그 이견을 다루는 데 시간을 배분할 수 있고 적지 않으면 이견은 미팅 중반에 불쑥 나타나 남은 순서를 흔든다. 6번의 선택지는 둘이나 셋으로 제한하고 각각에 드는 비용을 함께 적는다. 비용은 시간, 시료, 계산 자원, 그리고 그 선택을 했을 때 포기하게 되는 다른 작업이다. 선택지를 하나만 제시하면 그것은 통보이고 다섯 개를 제시하면 미팅이 선택지를 줄이는 데 다 쓰인다. 7번은 6번에서 무엇을 골라 달라는 요청이며 문장 하나로 쓰되 결정의 형태를 분명히 한다. “방향을 논의한다”로 쓰면 결정의 형태가 정해지지 않으므로 “A와 B 중 하나를 고른다”로 쓴다. 8번은 미팅 전에 초안으로 적어 두고 미팅 중에 고친다. 다음 작업의 초안이 있으면 결정 직후 그때 책임자와 기한을 채울 수 있고 초안이 없으면 결정만 남고 실행은 다음 주로 밀린다.

질문별로 답할 수 있음과 답할 수 없음을 나누기

공동연구에서 분석을 맡으면 시간이 지날수록 원래 질문에서 멀어진다. 데이터를 들여다보면 흥미로운 것이 계속 나오고 그중 몇 개를 따라가다 보면 처음에 상대가 물었던 것과 다른 자리에 도착한다. 발견 자체는 가치가 있지만 상대는 자기 질문에 대한 답을 기다리는 중이고 답이 오지 않으면 협업의 신뢰가 먼저 닳는다. 사전 자료를 만들 때 가장 먼저 할 일은 상대가 처음에 보낸 질문의 원문을 다시 읽는 일이다. 기억하고 있는 질문과 실제로 받은 질문이 다른 경우가 생각보다 흔하다. 원래 질문을 옮겨 적고 질문마다 네 칸을 채운다. 현재 자료로 답할 수 있는 내용, 그 답을 지지하는 그림과 수치, 지금 자료로는 답할 수 없는 부분, 답하려면 필요한 최소한의 추가 분석이다. 네 번째 칸에서 최소라는 조건이 중요하다. 요청받지 않은 분석을 먼저 벌여 놓고 그 결과를 가져가면, 상대는 자기 질문에 대한 답 대신 새 그림을 받게 되고 논의는 다시 처음으로 돌아간다. 무엇을 더 해야 할지는 이번 미팅에서 합의할 항목이다.

사전 자료에서 가장 자주 비는 항목은 답할 수 없는 질문을 적는 칸이다. 비워 두면 상대는 그 질문이 답해진 것으로 오해하거나 아니면 답이 없다는 사실을 미팅 중에 처음 알게 된다. 뒤의 경우가 특히 나쁜데, 그때 미팅의 주제가 갑자기 “왜 안 되는가”로 바뀌면서 준비한 순서가 무너지기 때문이다. 답할 수 없는 이유를 함께 적으면 논의의 방향이 달라진다. 표본이 부족한 것인지, 설계상 그 질문에 답할 수 없는 것인지, 아니면 필요한 대조 조건이 없는 것인지에 따라 상대가 제안할 해법이 완전히 다르다. 질문별 표는 미팅에서 처음 보여주는 것보다 며칠 앞서 보내는 편이 낫다. 상대가 자기 질문에 대한 현재 상태를 미리 보면, 미팅에 오면서 이미 두 가지를 정하고 온다. 답할 수 없다고 표시된 항목 중 무엇이 자기에게 정말 필요한지, 그리고 그 항목을 위해 자기 쪽에서 무엇을 내놓을 수 있는지를. 표를 미리 보내지 않으면 두 판단이 미팅 중에 즉석으로 이루어지고 즉석 판단은 대개 “일단 다 해 봅시다”로 수렴한다. 질문의 우선순위를 상대가 정하게 하는 일이 분석을 맡은 쪽의 시간을 가장 크게 아껴 준다. 작업 과정과 수정 이력은 뒤로 보낸다. 어떤 단계에서 오류를 발견해 다시 돌렸고 어떤 도구의 버전을 바꿨고 며칠이 걸렸는지는 성실함의 증거이지만 상대의 판단에는 거의 쓰이지 않는다. 재현에 필요한 정보라면 부록이나 별도 문서에 두고 사전 자료 본문에서는 참조만 한다. 예외는 그 과정이 결과의 해석을 바꾸는 경우다. 파이프라인을 바꾼 뒤 결과의 방향이 달라졌다면 그 변경은 근거의 신뢰도에 관한 정보이므로 3번 현황과 5번 불확실 항목에 들어가야 한다.

미팅 자료에 담기는 연구자의 판단

에이전트로 슬라이드 초안을 만드는 일은 이제 어렵지 않다. 사전 자료의 여덟 항목을 넘겨주면 구조가 잡힌 초안이 나오고 문장은 대체로 매끄럽다. 문제는 그 초안이 발표자의 판단을 담고 있지 않다는 데 있다. 초안에 적힌 “이 결과는 A 가설과 일치한다”는 문장은 그럴듯하지만 어느 정도로 일치하는지, 다른 가설로도 설명되는지, 그 판단에 얼마나 자신이 있는지는

발표자만 정할 수 있다. 초안을 받은 다음에 할 일은 각 문장을 자기 판단으로 다시 세우는 일이다. 한 장씩 넘기며 스스로에게 묻는 질문은 두 개다. 이 장이 여기 있는 이유는 무엇인가. 이 장을 보고 나서 청자가 무엇을 판단할 수 있게 되는가. 두 질문에 답할 수 없는 장은 뺀다. 남긴 장에는 그 장이 지지하는 결론을 제목으로 단다. 제목을 “발현량 비교”로 달면 청자는 그림에서 무엇을 봐야 하는지 스스로 찾아야 하고 “처리군에서 발현량이 높지만 대조군 편차가 커서 추가 확인이 필요하다”로 달면 그림을 그 결론의 근거로 읽는다.

그림은 다시 만든다. 분석 과정에서 만든 그림은 자기가 보려고 만든 것이라 축 이름이 줄어 있거나 범례가 잘려 있거나 화면에서는 읽혔지만 회의실 화면에서는 읽히지 않는 크기다. 읽히지 않는 그림은 없는 그림과 같고 청자는 읽으려 애쓰다가 설명을 놓친다. 한 장에 그림을 여러 개 넣는 것도 같은 문제를 만든다. 한 장에서 하나의 판단만 요청하면 그림은 자연스럽게 하나나 둘로 줄어든다. 결론의 강도를 표현하는 언어도 발표자가 정해야 한다. “시사한다”, “일관된다”, “보여준다”, “확인했다”는 서로 다른 강도이고 청자는 그 차이를 근거의 강도로 읽는다. 에이전트가 만든 초안은 대체로 중간 강도의 표현을 고르는데 실제 근거는 그보다 약하거나 강한 경우가 많다. 표본이 셋인 예비 실험 결과에 “확인했다”를 쓰면 그때는 통과하더라도 나중에 되돌리는 비용이 크고 반대로 여러 조건에서 재현된 결과에 “시사한다”를 쓰면 상대가 판단을 미룬다. 각 문장의 동사를 근거의 강도에 맞춰 다시 고르는 작업을 초안 검토의 마지막 단계로 둔다. 발표자가 그때 근거를 대며 방어할 수 없는 문장은 공유하기 전에 빼는 편이 낫다.

동료에게 미리 발표하는 일은 어디서 이해가 막히는지 표시하는 절차다. 듣는 사람에게 “이해가 안 되는 지점에서 바로 끊어 달라”고 부탁하고 끊긴 지점을 기록한다. 같은 지점에서 두 사람이 끊었다면 그 자리는 반드시 고친다. 리허설에서 가장 자주 드러나는 것은 발표자가 당연하게 여기는 전제인데, 발표자는 그 전제를 설명해야 한다는 생각 자체를 하지 못하므로 혼자서는 찾을 수 없다. 에이전트가 만든 슬라이드를 이해하지 못한 채 발표하면 그 대가는 질문 하나로 돌아온다. 그림의 세부에 대한 질문, 왜 이 비교를 했는지에 대한 질문, 다른 조건에서는 어떤지에 대한 질문에 답하지 못하면 그때까지 쌓인 신뢰가 한 번에 내려간다. 미팅 자료가 실제로 전달하는 것은 이 사람이 자기 데이터를 얼마나 이해하고 있는가에 대한 판단이다. 한 번 흔들린 판단은 같은 프로젝트의 다음 미팅까지 이어져, 잘 준비한 자료도 의심을 받으며 읽힌다.

1. 슬라이드를 네 장에서 여섯 장으로 만든다
2. 각 장의 제목은 그 장이 지지하는 결론으로 단다
3. 동료 한 명에게 먼저 발표하고 이해가 막힌 지점을 기록해 고친다

미팅 중 사실과 해석과 결정의 분리

미팅 노트를 발언 순서대로 적으면 나중에 쓸 수 없다. 누가 무슨 말을 했는지는 남지만 그중 무엇이 합의된 것인지가 남지 않기 때문이다. 발언록을 다시 읽으면서 합의 지점을 복원하는 일은 시간이 지날수록 어려워지고 참석자마다 다르게 복원한다. 노트를 세 칸으로 나눠 적으면 적는 순간에 분류가 끝난다. 분류하며 적는 일이 발표자에게 부담이라면 기록자를 따로 두되, 세 칸 구조를 미리 알려 준다.

구분	기록할 내용
근거	확인된 데이터, 문헌에서 온 근거, 자원과 일정의 제약
해석	가능한 설명, 경쟁 가설, 남은 불확실성
결정	선택한 방향, 보류한 선택, 그렇게 정한 이유

합의되지 않은 해석을 결정 칸에 적지 않는 규율이 세 칸 구조의 전부라고 해도 지나치지 않다. 미팅에서 누군가 “그렇다면 A일 가능성이 높겠네요”라고 말했고 아무도 반박하지 않았다고 해서 A가 합의되지는 않는다. 반박하지 않은 이유는 동의일 수도 있고 자기 전문 영역에서 벗어난 이야기여서일 수도 있고 그 순간 다른 생각을 하고 있어서일 수도 있다. 해석 칸에 적어 두고 결정이 필요하면 그때 “그럼 A를 전제로 진행하는 것으로 정할까요”라고 물어야 결정이 된다. 물어보지 않고 결정 칸에 적힌 항목은 다음 미팅에서 “그런 얘기는 안 했는데요”로 되돌아온다. 논의가 새로운 분석 아이디어로 번지는 일은 미팅에서 늘 일어난다. 좋은 아이디어이므로 막을 이유는 없지만 중심 질문과의 거리를 그때 확인하지 않으면 결정 항목을 다루지 못한 채 시간이 끝난다. 아이디어가 나오면 해석 칸에 적어 두고 이번 결정과 연결되지만 확인한다. 연결되지 않으면 별도 목록으로 옮기고 다음 기회에 다룬다. 확인 없이 따라가면 미팅이 끝난 뒤 남는 것은 흥미로운 대화의 기억뿐이다.

누가 어떤 전문적 판단을 제공했는지도 기록한다. 특정 조건에서는 그 assay가 답을 주지 못한다는 지적, 이 표본 구성으로는 그 비교가 성립하지 않는다는 지적은 그 사람의 경험에서 나온 판단이고 나중에 같은 지점을 다시 만났을 때 누구에게 물어야 하는지를 알려 준다. 어떤 데이터가 더 필요한지에 대한 논의와 구분해서 적는 이유는 둘의 후속 작업이 다르기 때문이다. 전문적 판단은 물어서 얻고 부족한 데이터는 만들어서 얻는다. 둘을 섞어 적어 두면 물어보면

끝날 일에 새 실험을 계획하거나 실험이 필요한 자리에서 의견만 더 모으게 된다. 지적을 준 사람의 이름과 함께 그 지적이 어떤 조건에서 성립하는지도 한 줄 붙여 두면, 조건이 달라졌을 때 그 판단을 다시 물어야 하는지 알 수 있다. 녹음과 자동 전사가 세 칸을 대신하지는 않는다. 전사는 발언을 빠짐없이 남기지만 남긴 발언 중 무엇이 근거이고 무엇이 아직 가설이며 무엇이 합의된 결정인지는 표시하지 않는다. 전사본을 요약하게 하면 요약은 나오지만 그 요약이 합의 범위를 정확히 잘랐는지는 참석자가 다시 확인해야 하고 확인하는 시간이 세 칸을 직접 적는 시간보다 짧지 않다. 녹음 자체가 참석자 전원의 동의가 필요한 사안이라는 점도 있다. 형식을 정하지 않은 채 자동화하면 회의마다 다른 모양의 기록이 쌓인다.

회의 기록을 전사에서 노트로 바꾸는 경로

앞 절의 세 갈래 분리를 회의 중에 손으로 적는 것은 쉽지 않다. 말하면서 동시에 사실과 해석과 결정을 갈라 적으려면 논의에 집중하기 어렵고 놓친 부분은 회의가 끝나면 복원되지 않는다. 그래서 기록을 두 단계로 나눈다. 회의 중에는 전사가 원문 역할을 맡고 회의가 끝난 뒤에 사람이 그 원문에서 노트를 만든다. 4장에서 원문 PDF와 정리 문서를 나눈 것과 같은 구조다. 전사는 고치지 않는 층이고 노트는 그것을 읽고 판단해 만든 층이다. 전사를 그대로 노트로 삼으면 발언 순서대로 적힌 문서가 남고 그런 문서는 앞 절에서 말한 대로 나중에 쓸 수 없다.

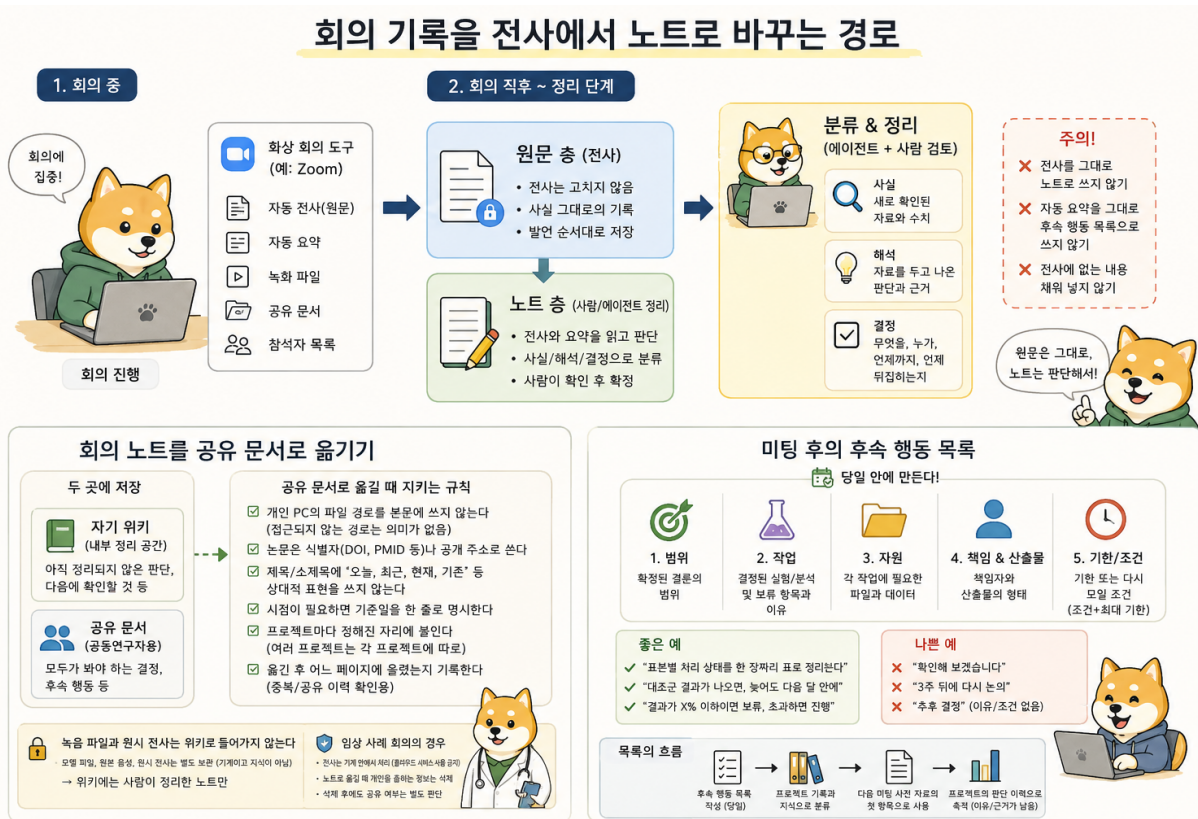


Figure 10: 회의 기록을 전사에서 검토된 노트와 후속 행동으로 바꾸는 경로

회의 중에는 전사를 원문으로 보존한다. 회의가 끝난 뒤 사실·해석·결정을 나누고 확인된 결정만 공유 문서와 후속 행동 목록으로 옮긴다.

화상 회의라면 전사를 얻는 경로가 도구 쪽에 이미 있다. Zoom을 예로 들면 회의가 끝난 뒤 그 회의에 딸린 자료를 한 번에 가져올 수 있고 여기에는 자동 생성된 요약과 참석자별 발언 기록, 녹화 파일, 회의 중 공유한 문서와 참석자 목록이 들어 있다. 자동 요약에는 짧은 개요와 전체 서술과 다음 할 일이 나뉘어 담긴다. 이 자료를 가져오는 연결은 읽기만 하고

회의를 만들거나 고치지 않으며 회의를 연 사람이거나 그 사람이 접근을 허용한 참석자만 가져올 수 있다. 연결을 어떻게 켜고 무엇을 정해 두어야 하는지는 22장에서 다룬다. 여기서 필요한 것은 이 자료가 원문 층에 놓인다는 점 하나다.

자동 요약은 그대로 회의 노트로 쓰지 않는다. 자동 요약은 무엇을 말했는지를 압축한 것이고 필요한 것은 무엇이 사실이고 무엇이 해석이고 무엇이 결정인지의 구분이다. 두 가지는 다른 작업이다. 자동 요약에서 다음 할 일 목록이 나오더라도 그것을 그대로 후속 행동 목록으로 삼지 않는데 누가 언제까지 무엇을 하는지가 회의에서 실제로 정해지지 않은 채 문장으로만 존재하는 경우가 흔하기 때문이다. 전사와 자동 요약을 함께 읽고 결정으로 확정된 것만 골라 노트에 옮긴다. 이 골라내는 작업은 에이전트에게 맡길 수 있고 결과를 사람이 확인한다.

따라해보기

아래 블록은 AI 도구의 채팅창에 그대로 붙여 넣는 프롬프트다. 회의가 끝난 뒤에 주는 요청은 다음 정도면 된다.

```
.
: < >
.
- :
- :
- : , , ,
.
.
```

마지막 세 줄이 이 요청의 핵심이다. 회의록을 만들라고만 하면 빈칸이 그럴듯하게 채워지고 채워진 담당자와 기한은 회의에서 합의된 것처럼 읽힌다. 정해지지 않았다고 적힌 항목은 다음 회의의 첫 안건이 되고 그럴듯하게 채워진 항목은 아무도 하지 않은 채로 남는다.

회의 노트를 공유 문서로 옮기기

만든 노트는 두 자리에 놓인다. 하나는 자기 위키이고 다른 하나는 공동연구자가 함께 보는 문서 도구다. 두 자리에 두는 이유는 접근권이 다르기 때문이다. 위키의 노트에는 아직 정리되지 않은 판단과 다음에 확인할 것이 들어가고 공유 문서에는 참석자 모두가 같은 내용을 보아야 하는 결정과 후속 행동이 들어간다. 공유 쪽에 옮길 때 지키는 규칙이 몇 가지 있다. 개인 컴퓨터의 파일 경로를 본문에 쓰지 않는다. 그 경로는 다른 사람에게 열리지 않고 열리지 않는 경로가 박힌 문서는 나중에 무엇을 가리키는지 알 수 없다. 논문을 가리킬 때는 식별자나 공개 주소를 쓴다.

제목과 소제목에는 상대적인 표현을 쓰지 않는다. 오늘, 최근, 현재, 기존 같은 말은 쓴 날에만 뜻이 통하고 반년 뒤에는 어느 시점을 가리키는지 알 수 없다. 시점이 필요하면 기준일을 한 줄로 명시한다. 회의 노트는 프로젝트마다 정해진 자리 아래에 붙인다. 자리를 정해 두지 않으면 회의 기록이 프로젝트 문서 여기저기에 흩어지고 반년 뒤에 지난 회의를 찾으려면 검색에 의존하게 된다. 여러 프로젝트가 걸린 회의는 각 프로젝트의 자리에 따로 붙인다. 하나로 합쳐 두면 접근권이 가장 넓은 쪽에 맞춰지고 넓게 열린 접근권은 나중에 좁히기 어렵다. 옮긴 뒤에는 어느 페이지에 올렸는지를 기록에 남긴다. 이 기록이 없으면 같은 회의 노트를 두 번 올렸는지, 상대 기관에 나간 것이 무엇인지 확인할 방법이 없다.

녹음 파일과 원시 전사는 위키로 들어가지 않는다. 모델 파일과 원본 음성은 기계이고 지식이 아니므로 위키 밖의 별도 자리에 두고 위키로 들어오는 것은 사람이 정리한 노트뿐이다. 임상 사례를 논의한 회의라면 16장의 경계가 함께 적용된다. 그 녹음에는 환자 정보가 음성으로 들어 있으므로 전사를 클라우드 서비스에 맡기지 않고 기계 안에서 처리하며 노트로 옮길 때 개인을 좁히는 정보를 지운다. 지운 뒤에도 공유 문서에 올릴지는 따로 판단한다.

미팅 후의 후속 행동 목록

정리는 미팅이 끝난 당일에 한다. 하루가 지나면 세 칸 노트에 적힌 짧은 문장이 무엇을 가리켰는지 흐려지고 사흘이 지나면 참석자들의 기억이 서로 갈라지기 시작한다. 정리해야 할 것은 다섯 가지다. 확정된 결론의 범위, 결정된 실험과

분석 그리고 보류된 항목, 각 작업에 필요한 파일과 데이터, 작업별 책임자와 산출물의 형태, 그리고 기한이나 다시 모일 조건이다. 다섯 가지를 채운 문서가 후속 행동 목록이고 미팅에서 실제로 남는 것은 이 목록뿐이다. 보류한 항목에는 보류한 이유를 반드시 적는다. 이유 없이 보류로만 표시된 항목은 다음 미팅에서 처음부터 다시 논의되고 같은 결론에 도달하는 데 같은 시간이 든다. 자원이 없어서 미룬 것인지, 다른 결과를 기다려야 해서 미룬 것인지, 아니면 하지 않기로 사실상 정한 것인지에 따라 다음 행동이 달라진다. 마지막 경우를 보류로 적어 두는 습관이 특히 비용이 크다. 하지 않기로 정한 것은 그렇게 적어야 목록이 짧아지고 짧아진 목록만 실제로 실행된다.

작업마다 산출물의 형태를 적는 일이 책임자를 적는 일만큼 중요하다. “확인해 보겠습니다”는 의사 표시일 뿐이어서 확인이 끝났는지 여부를 다음 미팅에서 다시 물어야 한다. 같은 작업이라도 “표본별 처리 상태를 한 장짜리 표로 정리한다”라고 적으면 완료 여부가 문서의 존재로 판정되고 그 표가 다음 사전 자료의 현황 항목이 된다. 산출물의 형태를 적기 어려운 작업은 대개 아직 충분히 좁혀지지 않은 작업이므로 적으려다 막히는 순간이 작업을 다시 정의할 기회다. 후속 행동 목록에서 산출물 칸이 비어 있는 항목은 다음 미팅까지 진행되지 않을 확률이 높다. 다시 모일 시점은 날짜보다 조건으로 적는 편이 나을 때가 많다. “3주 뒤”라고 정하면 3주 뒤에 아무것도 준비되지 않은 채로 모이는 일이 생기지만 “대조군 결과가 나오면”이라고 정하면 그 결과가 곧 미팅의 안건이 된다. 조건으로 적을 때는 조건이 언제까지도 충족되지 않을 경우의 확인 시점을 함께 둔다. 조건만 적어 두면 그 조건이 막혔을 때 아무도 다시 모이자고 말하지 않은 채 몇 달이 지나가기 때문이다. 두 방식을 섞어 “대조군 결과가 나오면, 늦어도 다음 달 안에”처럼 쓰면 조건과 기한이 모두 살아 있다.

1. 당일 안에 후속 행동 목록을 만든다
2. 위키로 되돌릴 지식과 프로젝트 기록에만 남길 내용을 나눈다

정리한 내용은 두 갈래로 나눠 저장한다. 한쪽은 위키로 되돌릴 수 있는 지식이다. 미팅을 준비하며 새로 들인 논문, 상대의 지적으로 수정된 방법 이해, 이 분야에서 특정 assay가 답할 수 있는 질문의 범위 같은 것은 이번 프로젝트를 떠나서도 다시 쓰이므로 해당 개념 문서와 방법 문서에 반영한다. 다른 한쪽은 프로젝트 기록에만 남길 것이다. 누가 무엇을 맡기로 했는지, 어느 기관의 자료를 언제까지 받기로 했는지, 아직 공개되지 않은 결과의 수치 같은 것은 프로젝트 폴더에 두고 공개 지식층에 섞지 않는다. 두 갈래를 나누는 판단은 한 가지 질문으로 끝난다. 이 문장을 이 프로젝트 밖의 사람이 읽어도 되는가. 후속 행동 목록은 다음 미팅 사전 자료의 첫 항목이 된다. 지난번에 무엇을 결정했고 그중 무엇이 실행되었는지가 새 문서의 맨 앞에 오면, 미팅은 지난 논의의 반복 없이 이어진다. 같은 프로젝트에서 미팅이 서너 번 반복되면 결정 기록이 프로젝트의 판단 이력이 되고 반년 뒤에 “왜 그 방향으로 갔는가”라는 질문이 왔을 때 답할 수 있는 근거가 된다. 원고를 쓰는 단계에서 방법 선택의 이유를 설명해야 할 때, 그랜트 보고에서 계획이 바뀐 경위를 적어야 할 때 이 이력이 그대로 쓰인다. 한 번의 미팅에서 나온 네 가지 파일은 프로젝트 아젠다 폴더에 미팅 날짜로 묶어 둔다. 미팅 전 여섯 문장, 사전 자료 한 장, 세 칸 노트, 후속 행동 목록이 그것이다. 목록이 제 몫을 하는지는 새 세션을 시작한 에이전트에게 그 목록만 주고 “이번에 무엇이 결정되었고 다음에 무엇을 해야 하는가”를 물었을 때 다른 자료를 찾지 않고 답할 수 있는지로 정해진다. 한 걸음 더 나아간 기준은 그 목록으로 다음 미팅 사전 자료의 첫 항목을 바로 채울 수 있는지도. 채울 수 없다면 결정의 범위나 책임자, 조건 중 하나가 비어 있는 것이다.

세 가지 사례

새로 시작하는 공동연구의 첫 미팅에서는 상대가 자기 데이터에 단일세포 foundation model을 적용해 볼 수 있는지 묻고 있고 이쪽은 그 모델들을 다뤄 봤지만 상대의 데이터 종류는 처음이다. 준비의 첫 단계는 상대에게 그 분야의 진입점이 되는 논문 두세 편을 요청하는 일이고 동시에 이쪽에서도 모델 쪽 진입점이 되는 review 한 편(Szałata 외, 2024)과 서로 다른 신뢰 전략을 쓰는 원저 두 편을 골라 보낸다. 양쪽이 상대의 논문을 미리 읽고 오면 첫 미팅에서 각자 자기 분야를 설명하는 데 쓸 시간이 사라지고 대신 “이 모델이 그 데이터에서 무엇을 답할 수 있고 무엇을 답할 수 없는가”라는 질문에서 시작할 수 있다. 첫 미팅에서 요청할 결정은 하나로 좁혀, 시범 분석의 대상 데이터를 무엇으로 할지만 정하고 나머지는 그 결정이 난 뒤에 다룬다. 계산으로 뽑은 후보를 실험으로 검증할지 정하는 미팅은 성격이 다르다. 여기서는 후보 목록을 보여주는 일보다 각 후보를 뽑은 근거의 강도를 구분해 주는 일이 중요하다. 어떤 후보는 여러 독립적인 근거가 겹쳐서 올라왔고 어떤 후보는 한 가지 지표만으로 올라왔으며 후자는 실험에서 떨어질 확률이 높다는 사실을 미리 밝혀야 실험을 맡는 쪽이 자원 배분을 판단할 수 있다. 사전 자료에는 후보별 근거, 지금 자료로 말할 수 있는 결론의 범위, 필요한 대조 조건, 그리고 각 assay가 실제로 답하는 질문의 차이를 적는다. 마지막 항목이 자주 빠지는데 같은 대상을 보는 두 assay가 서로 다른 질문에 답한다는 사실을 실험을 하지 않는 쪽은 놓치기 쉽다. 미팅에서 요청할 결정은 후보 몇 개를 어느 assay로 검증할지이고 그 결정이 나면 필요한 시료와 일정이 곧바로 후속 행동 목록에 들어간다. 분석이 진행 중인 다기관 자료에 대한 정기 미팅은 현황과 판단을 섞지 않는 훈련장이다. 사전 자료를 두 장으로 나눠, 한 장에는 기관별로 들어온 표본 수, 대조군 구성, 현재 처리 단계, 파이프라인과 참조 자료의 버전, 아직 도착하지 않은 항목을 표로만 적는다. 다른 한 장에는 지금까지의 자료로 말할 수 있는 것, 아직 말할 수 없는 것, 그리고 다음 한 달 안에 할 수 있는 최소한의 분석 하나를 적는다. 두 장을 섞으면 참석자들이 현황 숫자를 보면서 동시에 해석을 판단하게 되고 대개 숫자 쪽에서 논의가

멈춘다. 요청할 결정은 다음 분석을 무엇으로 할지 하나이고 나머지 시간은 아직 도착하지 않은 자료를 어떻게 처리할지에 쓴다.

19장. 논문 원고 폴더 구성하기

분석이 일단락된 폴더는 대체로 비슷하다. 그림이 열두어 개, 표가 몇 개, 반년치 미팅 자료가 날짜별로 쌓여 있고 파일 이름에 붙은 버전 표시 중 어느 것이 마지막인지는 만든 사람만 안다. 결과는 충분해 보이고 이제 쓰기만 하면 된다고 생각한다. 그런데 막상 문서를 열면 며칠이 지나도 서론의 세 번째 문장을 넘지 못한다. 막히는 자리는 대개 같다. 열두 개의 그림이 합쳐져 독자가 무엇을 다르게 믿어야 하는지가 아직 정해지지 않은 것이다. 원고를 쓰는 일은 연구 질문과 분석과 그림과 주장과 근거의 경계를 다시 맞추는 설계 과정이고 문장 교정과 참고문헌 서식은 그 설계가 끝난 뒤에 오는 공정이다. 순서를 뒤집으면 잘 다듬어진 보고서가 나온다. 주장을 설계한다는 것은 무엇을 말할지 정하면서 그 말이 무너지는 조건을 함께 적는 일이다. 무너지는 조건이 없는 문장은 어떤 관측으로도 반대할 수 없고 반대할 수 없는 문장은 심사에서도 후속 연구에서도 방어되지 않는다. 만들 것은 네 가지다. 주장과 근거를 짚은 지도, 그림의 서사, 원고 각 절의 논리, 그리고 근거 검증과 반론 점검의 기록이다. 네 가지 모두 원고 본문을 쓰기 전에 만들어지고 원고가 고쳐질 때마다 함께 고쳐진다.

주장 체계로서의 논문

우리가 이 데이터를 분석했다는 문장과 이 분석이 어떤 현상을 보여 준다는 문장은 종류가 다르다. 앞은 수행한 일에 대한 보고이고 뒤는 세상에 대한 주장이다. 보고에는 성실함이 담기고 주장에는 책임이 담긴다. 분석도 보고도 시간순으로 하다 보면 반년 동안 쌓인 미팅 자료의 순서가 그대로 원고의 순서로 옮겨 오는데 그 순서는 저자에게만 익숙하다. 독자는 이 연구가 무엇을 밝혔는지 알고 싶어서 원고를 열지, 저자가 어떤 경로로 여기까지 왔는지 알고 싶어서 열지 않는다. 표본 집단의 구성, 처리 파이프라인, 모델과 참조 자료의 버전은 주장이 성립하는 조건이고 방법 절과 보충 자료에서 한 글자도 틀리지 않게 적혀야 하는 근거다. 자료의 규모가 큰 분야일수록 이 조건이 주인공 자리로 올라오는데 큰 코호트와 새 파이프라인 자체가 성과처럼 느껴지기 때문이다. 코호트의 크기는 어떤 주장을 가능하게 하는 자원이고 그 자원으로 무엇을 갈랐는지 말하지 않으면 원고는 자원 소개서로 읽힌다.

원고가 어느 쪽인지는 결과 절의 소제목만 이어 읽어 보면 30초 안에 드러난다. 소제목이 데이터 처리 단계로 읽히면 아직 보고서이고 발견한 현상이 순서대로 전개되면 주장 체계다. 이어 읽었을 때 하나의 이야기가 들리지 않으면 그 원고에는 아직 목차가 없는 것이다. 중심 문장을 정하는 질문도 하나다. 이 연구가 끝난 뒤 독자가 무엇을 다르게 믿어야 하는가. 답이 한 문장으로 나오지 않으면 그 원고는 두세 편의 논문이 섞여 있는 형태이고 섞인 채로 투고하면 심사자마다 다른 논문을 읽고 다른 요구를 보내온다. 한 문장으로 좁히는 과정에서 잘려 나가는 결과가 아깝게 느껴지겠지만 잘린 결과는 사라지지 않고 다음 원고의 중심이 된다. 15장에서 그린 field map은 그 한 문장이 어떤 대화에 끼어드는지 알려 주고 17장에서 미리 정한 판정 기준은 결과를 읽을 강도를 이미 정해 두었으며 18장의 결정 기록은 공저자들과 무엇을 주장하기로 합의했는지 보여 준다. 파트 2의 세 장이 만든 문서가 여기서 원고의 재료가 된다.

중심 주장에서 필요한 근거를 역설계하기

역설계의 절차는 단순하다. 중심 주장을 먼저 적어 놓고 그 주장이 참이라면 무엇이 관측되어야 하는지 묻고 그 관측이 다른 이유로는 나올 수 없었는지 다시 묻는다. 첫 질문의 답이 받치는 주장의 목록이 되고 둘째 질문의 답이 대안 설명의 목록이 되며 아직 손에 없는 관측이 남은 분석과 실험의 목록이 된다. 이미 얻은 결과에서 거꾸로 출발하면 결과의 목록이 만들어지는데 결과의 목록에서는 무엇이 빠졌는지 보이지 않는다. 주장과 근거를 짚지어 적은 표를 claim-evidence 지도라고 부른다. 만드는 시점은 분석이 절반쯤 남았을 때가 가장 좋다. 그때 만들면 빈칸이 남은 분석의 계획서가 되고 다 끝난 뒤에 만들면 같은 빈칸이 말할 수 없는 것의 목록이 된다. 둘째 질문이 가장 어려운데, 자기 주장을 무너뜨릴 설명을 스스로 떠올리는 일이라 훈련 없이는 잘 나오지 않는다. 표는 원고 폴더의 한 파일에 두고 분석이 진행되는 동안 계속 고친다.

칸	적는 것	비어 있으면
주장 가르는 관측 현재 근거	독자가 다르게 믿어야 하는 한 문장 주장과 대안 설명을 갈라놓는 관측 그 관측을 담은 그림과 수치의 위치	결과는 있는데 원고의 중심이 없다 결정적인 결과를 골라내지 못한다 아직 하지 않은 분석이 남아 있다

칸	적는 것	비어 있으면
대안 설명	같은 관측을 낳을 수 있는 다른 원인	심사에서 그 질문을 처음 받는다
무너지는 조건	어떤 결과가 나오면 철회하는가	검증 가능한 주장이 아직 아니다
허용되는 표현	근거가 허용하는 서술의 강도	문장마다 강도를 새로 협상한다

중심 주장은 하나로 제한하고 받치는 주장은 셋에서 다섯까지만 둔다. 받치는 주장마다 그것이 중심 주장에 왜 필요한지 한 줄로 적어 보면 개수는 저절로 줄어든다. 여덟 개까지 늘어난 목록에는 중심과 무관한 결과와 같은 주장의 반복이 섞여 있고 앞은 다른 논문으로 보내고 뒤는 하나로 합친다. 무너지는 조건 칸이 이 지도의 중심이다. 철회 조건을 적을 수 없는 문장은 어떤 관측으로도 반대할 수 없고 반대할 수 없으면 지지할 수도 없다. 이 칸이 비는 것은 대개 주장이 너무 넓기 때문이며 어떤 조건에서 어떤 대상에 성립한다는 형태로 좁히면 조건이 따라 나온다. 좁힌 주장이 약해 보이는 것은 표 안에서뿐이고 심사자 앞에서는 조건이 붙은 주장이 훨씬 오래 버틴다. 대안 설명 칸도 같은 시기에 채워야 값이 나가는데 투고 뒤에 처음 떠올린 대안 설명은 이미 심사자가 먼저 적어 보낸 뒤이기 때문이다. 첫 판은 30분 안에 끝내는 것이 목표이고 빈칸을 드러내는 것이 목적이다.

1. 준비 중인 논문의 중심 주장을 한 문장으로 쓴다. 두 문장이 필요하면 어느 쪽이 중심인지 정한다
2. 받치는 주장을 셋에서 다섯 개 적고 각각이 중심 주장에 왜 필요한지 한 줄로 덧붙인다
3. 각 행에 대해 가르는 관측, 현재 근거, 대안 설명, 무너지는 조건, 허용되는 표현을 채운다. 모르는 칸은 비워 둔다
4. 무너지는 조건 칸을 다시 훑으면서 비어 있는 행에 표시한다
5. 빈칸을 두 종류로 나눈다. 분석하면 채울 수 있는 칸과, 분석해도 채울 수 없어 주장을 좁혀야 하는 칸이다

허용되는 표현 칸에는 근거가 감당하는 서술의 강도를 미리 고정한다. 관찰한 것, 연관을 확인한 것, 예측이 맞은 것, 기전을 보인 것, 인과를 가른 것은 서로 다른 강도이고 같은 동사로 쓸 수 없다. 강도를 문장마다 새로 정하면 같은 결과가 어느 절에서는 시사하고 어느 절에서는 입증하는 일이 생기며 그 불일치는 저자보다 심사자가 먼저 발견한다. 데이터가 받쳐 주지 않는 문장은 등급을 낮춰 가설이나 향후 연구 방향으로 옮긴다. 옮겨 놓으면 그 문장은 결론 자리에서 빠지고 다음 연구의 출발점 자리에 남으므로 아까워서 결론에 붙들고 있을 이유가 없다. 지도의 어느 칸이든 비어 있는 채로 두어도 되지만 비어 있다는 표시는 남겨야 한다. 표시 없이 비워 두면 몇 주 뒤에 그 칸을 확인하고 비운 것인지 아직 확인하지 못한 것인지 구분할 수 없다. 표시가 남아 있으면 공저자에게 지도를 그대로 넘겨도 어디를 봐 달라는 말이 따로 필요하지 않다. 표가 다 채워졌는지는 첫 판의 기준이 되지 못한다. 새 세션을 시작한 에이전트에게 이 표를 주었을 때 어느 주장이 근거 없이 서 있는지와 어떤 분석이 남았는지를 짚어 낼 수 있는가로 판단한다. 무너지는 조건 칸이 빈 행이 있다면 그 행은 아직 인상에 머물러 있고 그것을 확인한 것만으로 첫 판은 제 몫을 한 것이다.

서론의 네 단계

서론은 독자를 저자와 같은 문제 앞에 데려다 놓는 곳이다. 네 단계로 설계한다. 분야가 중요하게 여기는 현상이나 문제를 세우고 지금의 지배적 설명이 그 현상의 무엇까지 설명하는지 밝히고 그 설명으로 풀리지 않는 구체적인 어긋남을 드러내고 이 연구가 어떤 관점과 근거로 그 틈에 들어가는지 말한다. 세 번째 단계가 서론의 무게를 결정한다. 알려진 바가 없다는 문장은 쓰기는 쉽지만 검증할 수 없고 그 분야를 아는 심사자에게는 문헌을 덜 읽었다는 신호로 읽힌다. 무엇이 아직 불확실한지, 어떤 두 결과가 서로 맞지 않는지, 어떤 조건에서는 확인되었지만 다른 조건에서는 확인되지 않았는지를 쓰면 그 틈이 손에 잡힌다. 15장에서 그린 field map의 합쳐진 사실이 두 번째 단계의 재료이고 같은 지도에 표시해 둔 논쟁 중인 해석과 비어 있는 근거가 세 번째 단계의 재료다. 지도를 미리 그려 둔 사람은 두 단계를 옮겨 적는 데서 시작하고 그리지 않은 사람은 서론을 쓰면서 지도를 처음 그린다. 뒤의 경우에 서론이 오래 걸리는 것은 문장 때문이 아니고 분야의 구조를 그때 처음 만들고 있기 때문이다.

연구 목적이 마지막 문단에서 갑자기 등장하면 앞의 세 단계와 끊긴다. 목적은 세 번째 단계에서 드러낸 어긋남에서 필연적으로 따라 나와야 하고 독자가 목적 문장을 읽기 전에 이미 그 연구가 필요하다고 느껴야 한다. 앞의 빈틈과 뒤의 목적이 맞지 않으면 심사자는 그 사이에 적히지 않은 다른 이유가 있다고 읽으며 그 의심은 원고 전체에 남는다. 문헌을 저자별로 나열하면 문단은 연대기가 되고 독자는 각 논문과 이 문제의 관계를 스스로 계산해야 한다. 하나의 설명을 세운 뒤 그 설명을 지지하는 근거를 함께 인용하고 설명이 흔들리는 지점에서 반대 근거를 인용하면 문단이 논증이 된다. 10장에서 다룬 인용의 기능 분류가 여기서 각 논문의 자리를 정해 준다. 어떤 인용은 문제의 중요성을 세우고 어떤 인용은 지배적 설명의 근거이며 어떤 인용은 그 설명이 맞지 않는 사례다. 같은 논문이 서론과 고찰에서 다른 기능으로 쓰이는 일도 흔하므로 기능을 표시해 두면 인용의 중복도 함께 정리된다. 반론 점검으로 중심 주장이 좁아지면 빈틈 서술도 함께 좁혀야 하므로 서론은 주장이 확정된 뒤에 한 번 더 읽는다.

결과의 배열 순서

결과 절은 독자가 중심 주장을 믿게 되는 순서로 놓는다. 실험한 순서와 그 순서가 같은 경우는 드물고 같다면 대개 우연이다. 두 순서가 어긋나는 이유는 분석이 탐색으로 시작하기 때문이다. 무엇이 나올지 모르는 상태에서 여러 방향을 동시에 열어 두고 보다가 그중 하나가 답이 되면, 나머지 방향은 답을 받치는 배경이나 대안 설명을 지우는 근거로 자리를 바꾼다. 첫 결과는 독자가 연구 질문과 데이터의 신뢰성을 이해하는 출발점이 맞는다. 가장 화려한 결과를 앞에 놓으면 독자는 그것을 믿을 근거가 없는 상태에서 읽게 되고 뒤에서 근거가 나와도 첫인상은 되돌아오지 않는다. 각 소절은 질문 하나, 답 하나, 그 답을 받치는 그림 하나 이상의 구조를 갖는다. 답 없이 관측만 나열된 소절은 결과가 되지 못한 자료의 묶음이고 독자는 그때 저자가 무엇을 주장하는지 스스로 정해야 한다. 소제목은 발견한 현상을 말한다. 사용한 방법의 이름을 소제목에 걸면 독자는 그 절이 무엇을 밝혔는지 본문을 다 읽고 나서야 알게 되고 어떤 분석을 돌렸는지는 어차피 방법 절에 적혀 있다.

이전 절의 답이 왜 다음 질문을 낳는지 한 문장으로 쓸 수 없으면 두 절은 그저 나란히 놓여 있다. 순서를 바꿔도 아무것도 달라지지 않는다면 아직 배열이 이루어지지 않은 것이다. 배열을 고칠 때 가장 큰 저항은 시간을 많이 쓴 분석에서 나온다. 석 달이 들어간 분석을 뒤로 미루거나 보충 자료로 내리는 결정은 감정적으로 어렵지만 투입한 시간은 그 결과가 앞에 와야 할 근거가 되지 못한다. 위치를 정하는 기준은 하나다. 이 결과가 없을 때 중심 주장이 어디까지 약해지는가. 예상과 다른 결과와 서로 모순되는 결과도 중심 주장의 경계를 만든다면 원고에 들어가며 무너지는 조건 칸에 적어 둔 항목과 관련된 결과라면 특히 그렇다. 그런 결과를 저자가 먼저 내놓으면 심사자는 저자가 자기 데이터를 끝까지 보았다고 판단하고 감춘 결과는 대개 재분석 요구와 함께 되돌아온다. 배열이 확정된 다음에 그림 번호를 매긴다. 번호를 먼저 붙이면 순서를 바꿀 때마다 본문의 참조를 함께 고쳐야 하고 고치다 빠뜨린 참조 하나가 투고 직전에 나온다.

그림 서사를 사람 손으로 검수하기

원고에 들어갈 그림을 순서대로 한 장씩 붙여 놓고 각 패널에 대해 여섯 가지를 적는다. 화면에서 넘겨 보는 것보다 인쇄해 책상에 늘어놓는 편이 낫고 이 작업의 결과물을 그림 서사(*figure story*)라고 부른다.

- 이 패널이 답하는 질문
- 관측된 핵심 결과
- 그 결과가 허용하는 해석의 범위
- 앞 패널에서 왜 이 패널로 넘어오는가
- 이 패널이 없으면 어떤 주장이 무너지는가
- 원본 데이터와 그림을 만든 스크립트의 위치

시간이 아깝게 느껴지는 것이 이 작업의 요점이다. 빠르게 넘기면 보이지 않고 한 줄씩 적는 동안에만 보이는 것들이 있다. 설명을 붙이지 않으면 무엇을 보여 주는지 알 수 없는 패널, 앞 패널과 같은 주장을 다른 형태로 반복하는 분석, 중심 서사와 관계없이 관성으로 남은 결과, 원본 근거 없이 발표 자료에만 존재하는 수치, 그림 순서와 결과 절의 논리가 어긋나는 지점이 여기서 드러난다. 다섯째 항목이 가장 자주 비고 가장 유용하다. 이 패널이 없으면 어떤 주장이 무너지는지 적을 수 없는 패널은 중심 주장에 봉사하고 있지 않다는 뜻이다. 그런 패널은 보충 자료로 내리거나 지우거나 그 패널이 실제로 받치는 주장을 원고에 세우는 셋 중 하나로 처리한다. 세 번째를 고르는 경우가 생각보다 많은데 중심 주장이 처음부터 좁게 잡혀 있었다는 뜻이기 때문이다. 여섯째 항목은 심사 대응에서 값을 한다. 재분석 요구가 왔을 때 원본 데이터와 스크립트의 위치가 적혀 있으면 며칠 걸릴 일이 반나절로 줄어든다. 적혀 있지 않으면 자기 그림을 자기가 다시 찾는 데 시간을 쓴다.

이 검수를 AI에게 맡기면 검수의 효과가 사라진다. 패널 순서를 정해 달라고 하면 그럴듯한 답이 나오지만 저자가 각 패널에서 무엇을 말할 수 있는지 모르는 상태는 그대로 남는다. 심사자의 질문은 그림에 대해 들어오고 답은 저자가 한다. 사람이 여섯 항목을 먼저 적은 뒤에 반복되는 패널이나 설명되지 않은 도약을 찾게 하는 순서로는 유용하다. 순서가 뒤집히면 서사가 그림에 맞춰지고 저자는 자기가 만들지 않은 논리를 방어하는 자리에 서게 된다. 그림을 다시 그리는 작업도 사람의 판단이 앞선다. 축 이름과 범례와 글자 크기를 고치는 일은 도구가 빠르게 해내지만 어느 패널을 남기고 어느 패널을 합칠지는 그림 서사를 적어 본 사람만 안다. 18장에서 미팅 슬라이드를 다시 만들 때 쓴 기준이 여기서도 그대로 쓰이고 한 장에서 하나의 판단만 요청한다는 원칙은 원고의 그림에서 더 엄격해진다. 슬라이드는 발표자가 옆에서 설명하지만 그림은 혼자 남아서 읽히기 때문이다.

LLM-Wiki로 주장의 문헌 근거를 검증하기

원고에는 자기 데이터 대신 문헌이 근거인 문장들이 있다. 서론과 고찰의 대부분이 그렇고 결과 절에서 관측에 해석을 다는 문장도 그렇다. 이런 문장이 심사에서 가장 자주 걸리는데 저자에게는 확인한 기억이 있지만 어디서 확인했는지는 남아 있지 않은 상태이기 때문이다. 확인하는 절차는 자기 데이터를 확인하는 절차와 같다. 주장에서 출발해 근거까지 내려가고 내려간 자리에서 실제로 무엇이 쓰여 있는지 본다. 위키가 있으면 내려가는 길이 이미 만들어져 있다는 것이 차이다. 4장에서 만든 source note가 논문과 원고 문장 사이의 중간 기착지이고 그 note에 적어 둔 한계와 조건이 판정의 재료가 된다. 길이 없으면 같은 확인을 할 때마다 PDF를 다시 찾는 시간이 통째로 붙는다. 확인할 문장을 고르는 순서는 중요도로 정한다. 중심 주장을 직접 받치는 문장, 심사자가 반드시 물을 문장, 다른 분야의 결과를 끌어온 문장이 앞에 오고 원고의 모든 인용을 같은 깊이로 확인할 시간은 어차피 없다.

```
↓
wiki page
↓
source note
↓      PDF
/      /      /
↓
```

검색은 후보를 좁히는 단계다. BM25 점수가 높다는 것은 비슷한 단어가 많다는 뜻이고 지지 여부는 점수로 정해지지 않는다. 9장에서 다룬 대로 완전한 질문으로 물으면 후보의 질이 올라가지만 올라간 뒤에도 결과는 여전히 후보다. 판정은 네 갈래로 나뉜다. 직접 지지는 그 논문의 결과가 이 문장이 말하는 관계를 그대로 보인 경우다. 부분 지지는 방향은 같지만 대상이나 조건이 달라서 원고 문장에 조건을 덧붙여야 하는 경우이고 실제로는 이 갈래가 가장 많다. 맥락만 제공하는 논문은 인용을 유지하되 문장이 거기에 기대는 인상을 주지 않게 고친다. 반대 근거는 왜 다른 결과가 나왔는지 다루는 자리에서 쓰거나 주장의 범위를 좁히는 데 쓴다. 제목과 초록만 보고 인용을 추가하는 일은 이 절차를 통째로 건너뛰는 것이고 초록이 본문 결과보다 강하게 쓰인 논문에서 특히 자주 사고가 난다. 인용을 넉넉히 달아 두면 안전해진다는 감각도 여기서 깨지는데 지지하지 않는 인용은 검증 비용만 늘리고 심사자가 그중 하나를 확인해 지지하지 않는다는 것을 발견하면 나머지 인용의 신뢰도까지 함께 떨어뜨린다.

근거의 강도가 논문마다 다르다는 점도 판정에 들어간다. 어떤 연구는 외부의 독립적인 기준과 얼마나 일치하는지로 신뢰를 얻고(Theodoris 외, 2023), 어떤 연구는 기존 방법과의 직접 비교로 신뢰를 얻는다(Cui 외, 2024). 신뢰를 만드는 방식이 다른 논문들을 한 문장 뒤에 나란히 인용하면 독자는 그것들을 같은 강도로 읽으므로 그 차이가 중요한 자리에서는 문장을 나눈다. 근거를 찾지 못하면 그 사실을 드러내고 문장의 강도를 낮추거나 무엇을 분석해야 자기 데이터로 그 말을 할 수 있는지 정한다. 후자가 나오는 자리가 원고를 쓰다가 분석으로 되돌아가는 자리이고 되돌아가는 일은 근거 구조가 실제로 작동했다는 신호다. 이 분야 논문처럼 써 달라는 요청과 이 문장을 지지하는 논문을 찾아 달라는 요청을 한 번에 시키면, 문체는 그럴듯한데 근거는 확인되지 않은 문장이 나온다. 문체를 참고하는 일과 과학적 주장의 근거를 대는 일은 다른 작업이므로 세션도 나누고 산출물도 나눈다. 문체 참고는 저자가 검토하면 끝나지만 근거 확인은 원문까지 내려가야 끝난다.

고찰의 네 부분

고찰의 첫 문단에는 중심 질문에 준 가장 직접적인 답이 온다. 그 답은 claim-evidence 지도의 중심 주장 칸에 이미 적혀 있으므로 새로 만들 것이 없다. 우리는 무엇을 분석했다는 문장으로 첫 문단을 시작하면 답을 아직 쓰지 않은 것이고 독자는 결과 절을 한 번 더 읽는 기분이 된다. 답을 쓴 다음에는 그 답이 기존 설명을 강화하는지, 적용 범위를 좁히는지, 반박하는지, 대체하는지를 밝힌다. 네 가지 중 어느 것인지 정하지 않으면 고찰은 결과의 재진술로 끝나고 이 연구가 분야를 어떻게 바꾸는지는 심사자가 스스로 판단하게 된다. 이어서 가장 중요한 기전이나 실용적 함의를 하나만 골라 깊게 다룬다. 함의를 여러 개 늘어놓으면 각각이 알아지고 얇은 함의는 근거 없이 확장한 주장으로 읽힌다. 그 뒤에 대안 설명과 근거의 경계를 정리하고 한계를 짧게 쓰고 다음 연구가 풀어야 할 질문을 남긴다.

한계를 다룰 때 두 가지 실패가 자주 보인다. 변명을 본문 곳곳에 흠뻑 뿌리면 저자가 자기 결과를 믿지 않는다는 신호가 되고 읽는 사람은 어느 결론까지 살아 있는지 알 수 없게 된다. 반대로 불편한 결과를 빼면 심사자가 그곳을 정확히 찾아낸다. 한계는 한곳에 모아 짧게 쓰고 근거가 정한 경계로 제시한다. 표본이 적다는 서술보다 이 규모에서 어떤 크기의 효과까지 검출할 수 있었는지를 쓰는 편이 방어하기 쉽고 17장에서 미리 정해 둔 판정 기준이 있으면 그 문장을 새로 만들 필요가

없다. 향후 방향에는 현재 결론을 실제로 가를 질문을 쓴다. 15장에서 시작해 계속 쌓아 온 미해결 질문 목록에서 이 연구가 답한 항목을 지우고 나면 남은 줄이 그 후보이고 목록을 유지해 온 사람은 향후 방향을 새로 지어내지 않는다. 더 큰 코호트에서 검증이 필요하다는 문장은 어느 논문이나 붙일 수 있어서 아무것도 말하지 않으며 어떤 조건에서 어떤 결과가 나오면 이 결론이 뒤집히는지를 쓰면 그것이 곧 다음 연구의 설계가 된다. 18장의 결정 기록에 공저자들과 어디까지 말하기로 합의했는지가 남아 있으면, 마감을 앞두고 결론의 강도를 처음부터 다시 협상하지 않아도 된다.

논리 단위로서의 문단

한 문단은 하나의 중심 주장을 세우고 근거, 반례, 해석, 함의까지 밀고 나가는 단위다. 문단을 나누는 기준은 주장의 개수다. 길이가 비슷해졌다고 자르거나 화면에서 길어 보인다고 끊으면 논리 단위가 깨진다. 한 문단에 주장이 둘 들어 있으면 독자가 어느 근거를 어느 주장에 붙일지 스스로 나누어야 하고 나누는 방식이 저자의 의도와 다르면 그 문단은 오해된 채로 읽힌다. 시가 만든 짧은 토막 문단을 그대로 이어 붙이면 이 구조가 무너진다. 토막들은 각각 사실이지만 그 사이의 논리는 저자의 머릿속에만 있고 원고에는 적혀 있지 않다. 초고를 시로 만들었다면 각 문단의 중심 주장을 먼저 정하고 그 주장에 필요한 문장만 남기는 방식으로 다시 짠다. 이 절에서는 살펴본다는 식의 안내 문장은 지우고 그곳에 실제 과학적 내용을 넣는데 안내 문장은 저자가 구조를 설명해야 한다고 느낄 때 나오고 구조가 잘 잡힌 문단은 설명 없이도 읽힌다.

시가 원고를 대신 쓰는 일과 근거 구조를 점검하는 일은 다른 작업이다. 이 문단을 써 달라는 요청은 원고를 만들고 이 문단의 중심 주장이 무엇인지 말해 보라는 요청은 원고를 검사한다. 검사 결과는 참고 자료이고 판정은 저자가 한다. 요약이 저자의 의도와 다르게 나왔다면 그 문단이 실제로 그렇게 읽힌다는 뜻이므로 요약을 반박할 자리에서 문단을 고친다. 서사의 주어는 현상, 근거, 측정, 모델, 제약으로 둔다. 어떤 연구진이 무엇을 보고했다는 문장이 이어지면 원고는 문헌 목록의 산문판이 되고 독자가 알아야 할 대상은 문장 뒤로 밀린다. 숫자는 근거가 허용하는 정밀도까지만 쓴다. 표본이 스몰인 실험에서 소수점 둘째 자리까지 적힌 비율은 정말해 보이는 만큼 근거를 넘어서는 것이고 그런 자리 하나가 원고 전체의 수치를 의심하게 만든다.

스스로 원고를 반박해 보기

초고가 서면 저자는 자기 원고를 가장 못 읽는 사람이 된다. 같은 문장을 여러 번 읽으면서 논리의 빈틈을 머릿속에서 자동으로 메우기 때문이고 메운 것은 원고에 적혀 있지 않은데도 저자에게는 적혀 있는 것처럼 보인다. 그래서 반박은 따로 설계한다. 반박은 주장이 어디서 무너지는지 찾는 작업이고 무너지는 곳을 찾아 주장을 고치는 것이 정상적인 결과다. 아무것도 고치지 않고 끝난 반박은 대개 반박이 약했다는 뜻이다. 시점도 결과를 가르다. 마감 직전에 하는 반박은 고칠 시간이 없어서 형식으로 끝나고 초고가 서고 공저자에게 돌리기 전이 가장 값이 나가는 시점이다. 그 시점에는 주장을 좁히는 결정도 아직 비용이 적다. 아래 여섯 질문을 원고 전체에 한 번씩 던지고 답이 막히는 곳을 표시한다.

- 중심 주장을 무너뜨릴 수 있는 가장 강한 대안 설명은 무엇인가
- 어떤 대조군이나 민감도 분석이 빠져 있는가
- 결과가 표본 구성, 배치, 임계값, 처리 방식의 선택에 의존하는가
- 그림이 보여 주는 것보다 문장이 더 강하게 말하는 곳은 어디인가
- 인용한 논문이 실제로 이 관계를 직접 지지하는가
- 재현에 필요한 입력, 코드, 매개변수, 결과의 연결이 남아 있는가

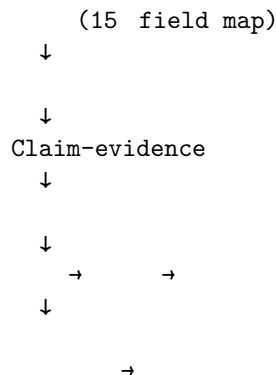
SI에게 이 질문들로 원고를 공격하게 할 때 두 가지 실패가 나온다. 표본이 작다거나 다른 코호트에서 검증이 필요하다는 식의 일반적인 반론만 반복하는 경우는 원고만 주었을 때 생긴다. Claim-evidence 지도와 그림 서사를 함께 주면 반론이 구체적인 자리를 짚기 시작하고 어느 패널이 어느 주장을 받치지 못하는지까지 나온다. 다른 하나는 원고에 없는 사실을 지어내 반론의 근거로 쓰는 경우다. 모든 반론에 근거의 위치를 함께 물어 어느 그림, 어느 표, 어느 논문에서 나온 지적인지 대개 하고 위치를 댈 수 없는 반론은 검토 목록에서 뺀다. 지적이 그럴듯할수록 이 확인이 중요한데, 그럴듯한 반론을 그대로 받아들이 주장을 낮추면 근거가 있는 결론까지 함께 약해지기 때문이다. 반론도 원자료와 설계로 검증한다. 반론의 질은 무엇을 함께 주었는지로 정해지고 원고만 건네고 좋은 반론을 기대하는 일은 배경 없이 미팅에 들어가 판단을 요구하는 일과 같다.

반박이 끝나면 각 주장에 대해 유지, 축소, 철회 중 하나를 정한다. 유지하려면 대안 설명을 배제하는 근거가 원고 안에 있어야 하고 그 근거가 저자의 머릿속에만 있으면 축소가 맞다. 축소는 조건을 덧붙이는 형태로 하고 어떤 조건에서 어떤 대상에 성립한다는 문장으로 다시 쓴다. 철회한 주장은 다음 연구의 질문 자리로 옮겨 둔다. 이 결정을 SI가 대신할 수 없다는 점이 이 절의 요점이다. 이 근거로 여기까지 말하겠다는 판단은 그 결과를 감당하는 사람이 내리고 감당한다는 것은

심사에서 방어하고 후속 연구에서 되돌아올 몫을 받는다는 뜻이다. 강하게 쓰라는 압력은 공저자에게서도 오고 마감에서도 오며 자기 자신에게서도 온다. 그 압력과 근거의 경계가 충돌하는 자리에서 무엇을 고르는지가 결국 그 연구자의 평판을 만든다.

논문 작성의 전체 순환

앞의 아홉 절이 하나의 순환으로 이어진다.



눈여겨볼 곳은 마지막 화살표다. 반론 점검의 결과가 주장의 수정으로 돌아가고 수정된 주장이 지도를 바꾸며 바뀐 지도가 새 빈칸을 만든다. 한 번 쓰고 끝나는 작업으로 보면 이 되돌림이 낭비처럼 보이지만 논문은 이 고리를 여러 번 돈다. 투고 전에 두어 번 돌고 심사 대응이 고리를 한 번 더 도는 일이며 다음 논문의 출발점도 이 고리에서 남은 빈칸에서 나온다. 지도와 그림 서사를 문서로 남겨 두면 여러 번 도는 비용이 크게 줄어든다. 남겨 두지 않으면 심사 의견이 왔을 때 어느 주장이 어느 근거 위에 서 있었는지를 처음부터 복원해야 하고 복원한 기억은 원래의 판단과 조금씩 어긋난다. 여섯 달 전의 자신은 이미 남이라서, 그때의 판단을 지금 다시 만들면 다른 판단이 나온다. 지도를 갱신하는 시점은 분석이 하나 끝날 때와 주장이 하나 바뀔 때 둘뿐이고 그 두 시점에만 손대면 문서가 원고를 따라다니는 부담이 크지 않다. 고리를 도는 횟수는 줄일 수 없지만 한 바퀴의 비용은 줄일 수 있다. Claim-evidence 지도와 그림 서사는 원고가 바뀌면 함께 바뀌는 문서이므로 원고 폴더에 두되, 심사 대응 단계에서 다시 열게 되므로 투고한 뒤에도 지우지 않는다. 문헌 근거 검증에서 나온 판단은 위기로 되돌리는데 어떤 논문이 어떤 주장을 직접 지지하는지와 두 논문의 결과가 왜 어긋나는지는 다음 원고에서도 다시 필요하기 때문이다.

고리를 여러 번 도는 동안 원고의 상태가 바뀐다. 쓰던 원고가 투고되고 심사에 들어가고 출판되거나 멈춘다. 원고 작업 공간은 그 상태를 폴더로 표현하고 원고 하나가 작업 단위가 된다.

```

Manuscript/
  AGENTS.md
  wiki/
    index.md
    log.md
  manuscript-logs/
  figures/
  tables/
  writing-style/
  reviews/
  submissions/
  affiliations/
  recurring-feedback/
  paper_{ }/
  Reviews/
  Published/
  Standstill/

```

폴더 이름은 예시이므로 자기 말로 바꾸어 쓴다. 배치에서 지켜야 하는 것은 상태를 폴더로 표현하는 것과 로그를 원고 폴더 밖으로 뽑아내는 것 둘이다. 원고가 지금 어느 단계에 있는지는 폴더의 위치가 말해 주고 파일 이름에 붙인 최종이나

날짜 표시는 그 일을 하지 못한다. 상태가 바뀌면 원고 폴더를 통째로 옮기므로 안에 든 파일의 이름을 고칠 일이 없고 심사 의견이 왔을 때 어느 폴더를 열어야 하는지도 정해져 있다. 위키의 하위 폴더는 원고 작업의 국면마다 하나씩 둔다. 그림 규격, 표 규칙, 문체, 투고 절차, 소속 표기, 심사 대응, 반복해서 지적된 것이 각각 자기 자리를 갖는다. 한 원고에서 배운 것이 다음 원고에 그대로 쓰이는 자리가 이 국면별 문서들이고 같은 내용을 원고 폴더 안에 적어 두면 그 원고가 끝나는 순간 함께 묻힌다. 원고별 로그는 wiki/manuscript-logs/ 아래에 원고마다 한 파일씩 두고 덧붙이기만 하며 항목의 형식은 17장에서 쓴 여덟 항목과 같다. Claim-evidence 지도와 그림 서사는 원고 폴더 안에 두는데 원고가 바뀔 때마다 함께 바뀌는 문서이기 때문이다.

따라해보기

아래 블록은 AI 도구의 채팅창에 그대로 붙여 넣는 프롬프트다. 원고 작업 공간을 처음 세울 때 쓰는 프롬프트다. 빈 폴더에서 에이전트를 실행하고 그대로 붙여 넣은 뒤, 꺾쇠 자리만 자기 원고에 맞게 바꾼다.

```
1. AGENTS.md . AGENTS.md wiki/index.md , wiki/manuscript-logs/
2. . Reviews/, Published/, Standstill/
3. wiki/ index.md log.md manuscript-logs/, figures/, tables/, writing-style/, reviews/, submission-logs/
4. < >
5. wiki/manuscript-logs/< >.md . , , , , ,
6. , , , , wiki/index.md .
7. .
```

20장. 새로운 주제 온보딩 하기

대상은 두 종류다. 연구실에 새로 들어온 사람이 하나이고 이미 있던 사람이 새 프로젝트나 새 주제로 옮겨 갈 때가 다른 하나다. 두 경우 모두 그 주제에 대해 아직 아무것도 읽지 않은 상태에서 시작하므로 필요한 것이 같다. 새로 시작하는 사람에게는 첫 몇 주에 무엇을 읽어야 하는지를 정해 주어야 한다. 논문 목록을 만들어 건네는 방식이 가장 흔하고 대체로 잘 작동하지 않는다. 목록을 받은 학생은 논문을 하나씩 구해서 읽고 각자의 방식으로 정리하며 그 정리는 연구실이 쌓아 온 정리와 형식도 관점도 다르다. 몇 달 뒤 그 학생이 만든 문서를 연구실 위키에 합치려 하면 합쳐지지 않는다. 파트 1에서 만든 위키가 있으면 다른 방식을 쓸 수 있다. 지도교수가 먼저 그 분야의 중요한 논문을 읽고 흐름을 파악한 다음, 정리한 결과를 건네는 대신 골라낸 논문들을 건네는 것이다. 정리는 학생이 자기 위키에서 직접 한다. 고르는 일에 들어가는 판단이 지도교수의 몫이고 그 판단의 결과를 자기 것으로 만드는 일이 학생의 몫이다. 자기가 스스로 새 주제로 옮겨 갈 때는 고르는 쪽과 받는 쪽을 한 사람이 맡게 되므로 두 절반을 순서대로 자기에게 적용하면 된다.

고른 논문을 건네는 이유

정리해 둔 문서를 그대로 건네면 학생은 그것을 읽는다. 읽고 이해했다고 느끼고 실제로 그 순간에는 이해한다. 남는 것이 다르다. 남이 정리한 문서를 읽어서 얻는 것은 그 분야의 현재 지식이고 자기가 정리하면서 얻는 것은 그 지식에 이르는 경로다. 뒤의 것이 연구자에게 필요하다. 원문에서 어느 대목이 결론을 떠받치는지 고르고 그것을 자기 문장으로 옮기고 옮기다가 이해되지 않는 대목을 발견하는 과정이 훈련이기 때문이다. 정리된 문서를 받으면 그 과정이 통째로 빠지고 그래서 이 방식이 지금까지 시험해 본 것 가운데 가장 효과가 좋았다.

그렇다고 논문 목록만 보내는 것과 같지는 않다. 목록을 받으면 학생은 논문을 구하는 데서 시작하고 구하는 일은 배우는 것이 없으면서 시간만 든다. 그래서 PDF를 함께 보낸다. 무엇을 읽을지 고르는 판단과 원문을 손에 넣는 수고를 지도교수가 맡고 읽고 정리하고 잇는 일을 학생이 맡는 분업이다. 고르는 일에는 실제로 시간이 든다. 새로 온 학생은 아직 그 분야의 논문을 읽지 않았으므로 무엇이 중요한지 알 수 없다. 정리 문서와 논문 페이지를 함께 보낼 때도 있는데 그때는 형식의 예시로 준다는 것을 분명히 한다. 예시로 받은 문서는 자기 것을 만들 때 틀로 쓰고 그 논문의 내용은 원문에서 확인한다.

이 책이 근거로 삼은 위키에는 논문 페이지가 17,591건 있고 50개 카테고리에 나뉘어 있다. 새 학생에게 이 폴더를 통째로 주면 어디서 시작해야 하는지 알 수 없고 검색을 걸어도 자기 주제와 상관없는 문서가 대부분 올라온다. 규모가 큰 위키는 그 위키를 몇 년 동안 키운 사람에게만 지도로 작동한다. 그래서 잘라 낸다. 무엇을 잘라 내는가가 이 장의 실제 내용이고 잘라 내는 기준은 그 학생이 앞으로 석 달 동안 무엇을 해야 하는가다. 잘라 낸 묶음의 크기는 논문 열 편에서 스무 편 사이가 적당하다. 그보다 많으면 읽지 않고 그보다 적으면 비교할 대상이 생기지 않는다.

목록에 함께 담는 것

목록의 본체는 골라낸 원문 PDF다. 여기에 몇 가지가 함께 간다. 첫째는 읽는 순서와 그 순서의 이유다. 열 편을 아무 순서로나 읽으면 앞의 논문이 뒤의 논문을 이해하는 데 도움이 되지 않고 왜 이 순서인지가 적혀 있으면 그 자체가 분야의 지형을 알려 준다. 둘째는 위키의 지침 파일이다. 무엇을 어디에 두고 무엇을 확인하는지가 정해져 있어야 학생이 만든 문서가 나중에 연구실 위키와 맞는다. 셋째는 정리 문서와 논문 페이지의 예시 한둘이다. 스무 개를 보내면 읽을 자료가 되어 버리므로 한둘로 제한하고 그 논문은 순서의 맨 앞에 두어 학생이 자기 것과 대조해 볼 수 있게 한다. 넷째는 종합 문서다. 논문들이 어느 지도 위에 놓여 있는지는 종합 문서에만 적혀 있고 그것이 없으면 열 편이 서로 무관한 열 편으로 읽힌다.

종합 문서도 한두 개 함께 넣는다. 논문 페이지들이 어느 지도 위에 놓여 있는지는 종합 문서에만 적혀 있고 그것이 없으면 열 편의 논문이 서로 무관한 열 편으로 읽힌다. 넣을 종합 문서는 그 논문들이 실제로 연결된 것으로 고른다. 분야 전체를 서술하는 개관 하나와 반복해서 나오는 개념 문서 한둘이면 충분하다. 아직 답이 없는 열린 질문을 담은 문서를 함께 넣는 것도 방법이다. 연구실이 지금 무엇을 모르고 있는지가 그 문서에 적혀 있고 새로 온 학생이 가장 궁금해하는 것도 대체로 그것이다. 목차 문서는 반드시 넣는다. 폴더에 무엇이 있는지를 알려 주는 입구가 없으면 학생도 에이전트도 어디서 시작할지 정하지 못한다.

학생마다 달라지는 목록

무엇을 담을지는 그 학생이 무엇부터 해야 하는가로 정해진다. 분석 방법을 익혀야 하는 학생과 분야의 지형을 파악해야 하는 학생은 서로 다른 목록을 받아야 한다. 앞의 경우에는 방법을 서술한 논문과 그 방법을 쓴 원저 몇 편, 그리고 그 방법의 한계를 다룬 문서를 넣는다. 뒤의 경우에는 분야를 훑는 리뷰 한 편과 그 리뷰가 인용한 원저 몇 편, 그리고 최근 몇 년의 변화를 담은 개관을 넣는다. 두 목록에 같은 논문이 들어가도 상관없다. 같은 논문이 어느 목록에 들어가느냐에 따라 읽는 순서와 함께 놓이는 문서가 달라지고 그 차이가 무엇을 배우는지를 가르킨다. 목록을 만들 때 순서도 함께 정해 준다. 열 편을 아무 순서로나 읽으면 앞의 논문이 뒤의 논문을 이해하는 데 도움이 되지 않는다.

따라해보기

아래 블록은 AI 도구의 채팅창에 그대로 붙여 넣는 프롬프트다. 목록을 만드는 요청은 다음 정도면 된다. 자기 위키 폴더에서 세션을 열고 쓴다.

```
: < . >

1. .
2. .
3. 10 20 .

- PDF, source note, wiki page
-
- index.md
```

사람이 확인하는 단계를 두는 이유는 무엇을 넣을지가 지도의 판단이기 때문이다. 검색으로 올라온 논문이 그 학생에게 지금 필요한지는 그 학생의 배경을 아는 사람만 정할 수 있다. 순서에 이유를 한 줄씩 붙이라고 한 것도 확인을 위해서다. 이유가 붙어 있으면 순서가 이상할 때 어디가 잘못되었는지 보이고 그 이유를 그대로 index.md에 옮기면 학생이 왜 이 순서인지 알게 된다.

여러 학생에게 반복해서 만들 때

학생이 늘어나면 묶음을 만드는 일이 반복된다. 매번 처음부터 고르면 같은 논문을 매번 다시 고르게 되고 고를 때마다 조금씩 다른 목록이 나온다. 그래서 두 부분으로 나눈다. 연구실에 들어오는 사람이면 누구나 읽어야 하는 공통 부분과 그 학생의 주제에 맞춘 부분이다. 공통 부분에는 연구실이 무엇을 하는 곳인지 알려 주는 개관, 반복해서 쓰이는 방법을 정리한 문서, 그리고 데이터를 다루는 규칙이 들어간다. 이 부분은 한 번 만들어 두고 새 논문이 나올 때만 고친다. 개인 부분은 앞 절에서 말한 기준으로 매번 고른다. 두 부분을 나눠 두면 공통 부분을 고칠 때 그 개선이 다음에 오는 모든 사람에게 적용된다.

공통 부분이 실제로 고쳐지는지는 학생들이 어디서 막혔는지에 달려 있다. 세 사람이 연달아 같은 개념에서 막혔다면 그것을 설명하는 문서가 공통 부분에 없거나 있어도 읽히지 않는 것이고 그곳이 다음에 고칠 곳이다. 이 판단의 근거는 학생이 던진 질문이다. 온보딩 기간에 받은 질문을 그때그때 답하고 넘기면 같은 질문이 다음 사람에게서 다시 온다. 답한 내용을 개념 문서나 열린 질문 문서로 만들어 공통 부분에 넣으면 그 질문은 한 번만 받는다. 연구실의 위키가 자라는 방식 가운데 이것이 가장 값싼 축에 든다. 새 논문을 읽어서 늘리는 것과 달리 이미 아는 것을 적는 일이라 시간이 적게 들고 적어 두지 않으면 사람이 바뀔 때마다 사라진다.

잘라 낼 때 끊어지는 링크

위키의 일부만 뽑아낼 때 반드시 생기는 문제가 링크다. 논문 페이지들은 서로 위키링크로 이어져 있고 종합 문서는 수십 개의 논문 페이지를 가리킨다. 스무 편만 뽑으면 그 링크의 상당수가 묶음 안에 없는 문서를 가리키게 된다. Obsidian 같은 도구에서 그런 링크를 누르면 빈 문서가 새로 만들어지고 학생은 그 빈 문서를 원래 있어야 하는데 비어 있는 것으로 오해한다. 이 책이 근거로 삼은 위키에서 링크 규약이 어긋났을 때 정확히 같은 일이 벌어졌고 열 때마다 유령 페이지가 생겼다. 처리 방법은 두 가지다. 하나는 묶음 안에서 달하는 링크만 남기고 밖으로 나가는 링크는 일반 텍스트로 바꾸는 것이다. 다른 하나는 링크 대상을 한 단계까지 따라가 함께 담는 것인데, 한 단계만 따라가도 문서 수가 두세 배로 늘어난다는 점을 감안해야 한다.

링크가 성립하려면 vault의 루트가 어디인지도 맞아야 한다. 원래 위키에서 링크는 지식 총 폴더를 루트로 삼아 적혀 있고 잘라 낸 폴더에서도 한곳이 루트가 되어야 한다. 폴더 구조를 평평하게 펴서 건네면 링크가 전부 어긋난다. 잘라 낼 때 폴더의 총 구조를 그대로 유지하는 편이 간단하고 그렇게 하면 학생이 나중에 자기 논문을 넣을 때도 한곳에 넣게 된다. 링크가 제대로 걸렸는지는 건네기 전에 확인한다. 묶음 안의 모든 위키링크를 훑어 대상이 실제로 있는지 세어 보고, 없는 대상이 몇 개인지 학생에게 함께 알려 준다. 몇 개는 남아도 되지만 남아 있다는 사실을 모르면 학생이 자기 잘못으로 오해한다.

밖으로 나가면 안 되는 것

묶음을 건네는 일은 위키의 일부가 연구실 밖으로 나갈 수 있는 상태가 된다는 뜻이다. 학생은 자기 노트북에 그 폴더를 두고 그 노트북은 연구실 관리 아래에 있지 않을 수 있다. 그래서 건네기 전에 확인할 것이 정해져 있다. 지식 총에는 원래부터 출판된 내용만 담고 준비 중이거나 심사 중이거나 중단된 연구실 원고는 제목조차 들어가면 안 된다. 이 규칙이 지켜지고 있어도 새어 드는 경로가 여럿이다. 머리말의 상태 표시와 비어 있는 식별자, 출판 논문 페이지에 붙인 진행 중 과제 관련 절, 태그에 남은 코드명, 코드명으로 제목을 단 종합 문서, 그리고 그 문서를 가리키는 링크가 그렇다. 여섯 가지를 따로 적어 두는 이유는 하나만 막으면 나머지 다섯으로 새기 때문이다.

확인하는 기계로 한다. 살아 있는 과제의 코드명과 원고를 가리키는 표현을 묶음 전체에서 검색해 결과가 0건인지 본다. 특히 종합 문서를 조심해야 한다. 개관과 개념 문서에는 여러 논문을 읽고 내린 판단이 들어 있고 그 판단에 아직 발표하지 않은 결과가 섞여 들어가 있을 수 있다. 논문 페이지는 논문 한 편에 대한 서술이라 새어 나갈 것이 적지만 종합 문서는 그렇지 않다. 보충 자료를 함께 넣을 때는 배포 조건도 본다. 출판사가 배포를 제한한 파일이 있고 그 조건은 파일마다 다르다. 학생에 관한 기록은 이 묶음에 들어갈 일이 없다. 사람에 관한 기록은 애초에 다른 작업 공간이 소유하고 지식 총에 있지 않으며 그 기록은 점수나 등급이나 동료 비교를 만들지 않는다는 원칙 아래 운영된다.

묶음을 받은 학생이 첫 주에 하는 일

여기서부터는 받은 쪽의 이야기다. 폴더를 받으면 가장 먼저 지침 파일과 목차 문서를 읽는다. 두 문서가 이 폴더에서 무엇을 어디에 두기로 했는지를 알려 주고 그 규칙을 모르면 자기 문서를 아무 데나 만들게 된다. 그다음에 정해진 순서대로

논문을 읽는데 읽는 방식은 정리 문서를 먼저 읽고 원문으로 내려가는 순서가 빠르다. 정리 문서를 읽고 이 논문이 무엇을 했는지 파악한 뒤, 결론을 떠받치는 수치와 조건을 원문에서 확인한다. 확인하다가 정리 문서와 원문이 어긋나는 곳을 찾으면 그것을 적어 둔다. 어긋난 대목을 찾아내는 일은 새로 온 사람이 가장 잘하는 일 가운데 하나인데, 그 문서를 쓴 사람은 자기가 쓴 것을 다시 읽을 때 원문 대신 기억을 보기 때문이다.

세 편쯤 읽고 나면 위키에 질문을 던져 본다. 답이 근거와 함께 나오는지, 근거로 제시된 문서가 실제로 그 내용을 담고 있는지를 확인한다. 이 단계에서 위키가 어떻게 작동하는지가 손에 잡힌다. 그다음이 자기 논문을 한 편 넣는 것이다. 자기 주제와 관련해 최근에 읽은 논문 한 편을 골라 4장의 여섯 단계를 그대로 따라간다. 이때 남이 쓴 문서 열 개가 형식의 기준이 되므로 빈 문서 앞에서 막히지 않는다. 넣고 나면 그 논문을 기존 종합 문서와 잇는데 잇는 일이 가장 어렵고 가장 배우는 것이 많다. 어느 개관에 붙일지 정하려면 그 개관을 읽어야 하고 읽고 나면 그 분야가 지금 어디까지 왔는지 처음으로 자기 문장으로 정리된다. 첫 주에는 이 순환을 한 바퀴 돈다.

이 단계에서 흔한 실수가 하나 있다. 스무 편을 다 읽은 다음에 무언가를 시작하려는 것이다. 다 읽고 나면 앞의 논문이 기억나지 않고 기억나지 않는 채로 종합하려니 다시 읽게 된다. 세 편째부터 질문을 던지고 다섯 편째부터 자기 논문을 넣으라고 하는 이유가 여기 있다. 읽기와 쓰기를 번갈아 하면 앞에서 읽은 것이 뒤의 작업에서 쓰이므로 기억에 남는다. 다른 실수는 정리 문서를 원문 대신 읽고 넘기는 것이다. 정리 문서는 잘 쓰여 있고 읽기 편해서 그것만으로 이해했다고 느끼기 쉬운데, 그 느낌은 남의 판단을 자기 판단으로 착각한 상태다. 논문 한 편에서 원문으로 내려가 확인할 대목은 서너 개면 충분하므로 시간이 많이 들지도 않는다. 어느 대목으로 내려갈지 고르는 감각이 붙는 것이 첫 달의 실제 성과다.

학생의 문서가 연구실 위키로 돌아오는 경로

학생이 만든 문서가 연구실 위키로 돌아올 수 있는지는 처음에 같은 형식을 주었는가로 대체로 정해진다. 형식이 같으면 파일을 옮기고 링크를 잇는 것으로 끝나고 형식이 다르면 다시 쓰는 것과 비용이 같다. 다만 자동으로 합치지 않는다. 학생이 만든 정리에는 검증을 거치지 않은 판단이 들어 있고 그것이 그대로 위키에 들어가면 이후의 모든 종합에 그 오류가 섞인다. 합치는 절차는 미팅을 거친다. 학생이 넣은 논문 가운데 연구실 위키에 필요한 것을 함께 고르고 그 문서의 결론 부분을 지도교수가 원문과 대조한 뒤에 옮긴다. 이 과정 자체가 지도의 한 형태다. 어느 문장이 원문에 있고 어느 문장이 해석인지를 함께 짚는 일이 논문을 읽는 훈련의 상당 부분이다.

학생에게서는 문서와 함께 읽다가 찾아낸 어긋남, 정리 문서에서 빠져 있던 조건, 그리고 위키에 답이 없던 질문이 함께 돌아온다. 세 번째가 특히 값이 크다. 새로 온 사람이 던지는 질문은 그 위키를 오래 쓴 사람이 이미 답을 안다고 여겨 문서에서 빠진 대목을 정확히 짚는다. 그 질문을 열린 질문 문서로 만들어 두면 다음에 오는 학생이 한곳에서 막히지 않는다. 학생마다 다른 대목에서 막히므로 이 문서는 학생이 늘어날수록 촘촘해진다.

묵음이 낡는 속도

건넌 묵음은 만든 날의 스냅샷이고 연구실 위키는 그 뒤로도 계속 자란다. 반년이 지나면 새 논문이 들어와 종합 문서가 고쳐져 있고 학생이 가진 묵음의 종합 문서는 그때 그대로다. 그래서 묵음을 정본으로 삼지 않는다. 학생에게 건넌 것은 시작점이고 그 뒤로는 학생이 자기 폴더를 자기 축으로 키운다. 낡았다고 다시 뽑아 주는 방식은 대체로 좋지 않다. 학생이 그동안 넣은 문서와 새로 뽑은 묵음을 합치는 일이 생기고 합치다 보면 어느 쪽이 자기 것인지 알 수 없게 된다. 지도교수의 위키에서 무언가가 크게 바뀌었다면 그 문서 하나만 알려 주는 편이 낫다. 어느 문서를 건넸는지 기록해 두면 그 판단이 쉬워지는데 목록이 없으면 그 학생이 무엇을 가지고 있는지 반년 뒤에 알 수 없다.

같은 이유로 학생의 폴더를 지도교수의 폴더와 동기화하지 않는다. 두 폴더는 소유자가 다르고 판단의 책임도 다르다. 지도교수의 위키에 있는 문장은 지도교수가 검증한 것이고 학생의 위키에 있는 문장은 학생이 검증한 것이다. 두 폴더를 이어 붙이면 누가 무엇을 검증했는지가 사라지고 학생은 자기 판단을 훈련할 기회를 잃는다. 두 폴더는 앞 절에서 말한 미팅을 통해 연결한다. 학위 과정이 끝날 때도 같은 원칙이 적용된다. 학생이 키운 폴더는 그 학생이 가지고 나가고 연구실 위키에 남는 것은 미팅을 거쳐 옮겨 온 문서다. 나가는 사람의 폴더를 통째로 흡수하려 하면 검증하지 않은 문장이 한꺼번에 들어오고 그때는 어느 것이 검증된 것인지 가릴 방법이 없다. 반대로 나가는 사람에게 폴더를 두고 가라고 하면 그 사람이 몇 년 동안 만든 작업 기억을 빼앗는 것이 된다. 사람이 여럿이 되면 이런 결정이 매번 생기므로 그때마다 새로 판단하지 않으려면 규칙이 필요하다.

21장. 에이전트가 대신해 주지 않는 것, 당신의 과학

파트 2의 여섯 장에서 논문을 읽고 주장을 쓰며 새 사람에게 건넨 산출물을 만들었다. 이 산출물이 늘어도 학생의 실력이 저절로 늘지는 않는다. 논문을 위키에 넣고 요약하면 그 내용을 안다고 느끼기 쉽지만 그 느낌과 실제로 아는 것 사이의 거리는 멀다. 요약을 문서 없이 설명하고 원문과 대조하고 다른 사람의 질문에 답해 봐야 그 거리가 드러난다. 이 확인에는 사람이 직접 시간을 써야 한다. 도구가 빨라지면 자료를 모으고 정리하는 시간은 줄어도 이해와 판단에 드는 시간은 남는다. 읽고 쓰고 말하는 훈련을 건너뛰면 산출물만 늘고 실력은 그대로다. 학생이 이 구분을 놓치면 연구 결과의 강도를 스스로 정하지 못한다. 지도하는 사람이 놓치면 같은 오해가 다음 학생에게 전달된다.

요약을 읽는 것과 아는 것

논문을 넣고 정리 문서를 받으면 그 논문이 무엇을 했는지가 몇 문단으로 정리되어 나온다. 잘 쓰여 있고 읽기 편하다. 문제는 읽기 편하다는 것 자체에 있다. 읽는 데 힘이 들지 않으면 머릿속에서 다시 구성하는 단계가 생략되고 생략된 채로 이해했다는 느낌만 남는다. 며칠 뒤에 그 논문에 대해 질문을 받으면 답이 나오지 않고 그때 다시 문서를 열게 된다. 아주 빠른 학생은 요약만 보고도 여러 지식을 서로 잇는다. 대부분의 경우는 그렇게 되지 않는다. 지식이 이어지려면 그 내용을 자기 구조 안에 놓아야 하고 그 놓는 작업은 읽는 것과 다른 작업이다.

요약은 두 번째 읽기로 돌아올 자리를 마련한다. 서문에서 적은 대로 조건이 하나 붙는다. 정리 과정에서 나오는 내용을 이해되지 않더라도 눈으로 한 번은 읽어야 한다는 것이다. 길지 않다. A4 반 장 정도다. 이 한 번의 읽기가 있는 것과 없는 것의 차이가 크다. 읽어 두면 나중에 그 주제에 대한 질문이 생겼을 때 어디를 열어야 하는지 알게 되고 열어서 다시 읽을 때 두 번째 읽기가 된다. 읽지 않고 넘기면 그 문서는 폴더 안에만 있고 머릿속에는 없다. 4장에서 ingest의 완료 조건에 종합 총 연결을 넣은 것도 같은 이유였다. 연결하려면 기존 종합 문서를 읽어야 하고 읽는 순간 그 논문이 어디에 놓이는지를 생각하게 된다.

줄지 않고 옮겨 가는 시간

에이전트를 쓰면 시간이 줄어든 것 같지만 실제로는 옮겨 간다. 원문을 처음부터 끝까지 읽는 시간은 줄고 대신 확인하고 판단하는 시간이 는다. 정리 문서가 원문과 어긋나지 않는지 대조하는 시간, 이 논문이 기존 판단을 어떻게 바꾸는지 정하는 시간, 그리고 그 판단을 문장으로 적는 시간이 새로 생긴다. 이 시간들은 건너뛸 수 있고 건너뛰면 실제로 빨라진다. 빨라진 만큼 남는 것이 없어질 뿐이다. 처음 이 방식을 쓰는 사람이 겪는 것도 대체로 이 순서다. 몇 주 동안 논문이 빠르게 쌓이고 몇 달 뒤에 그중 무엇을 기억하고 있는지 물으면 답이 잘 나오지 않는다.

배우는 과정만 놓고 보면 오히려 시간이 더 걸릴 수도 있다. 예전에는 논문 한 편을 읽고 넘어갔다면 지금은 읽고 정리하고 기존 문서와 대조하고 어긋나는 자리를 확인하고 판단을 적는다. 단계가 늘었으므로 한 편당 드는 시간은 비슷하거나 더 길다. 달라지는 것은 그 시간의 결과가 남는가다. 예전 방식에서 읽은 논문 스무 편은 기억에만 있고 서로 이어져 있지 않았다. 지금은 스무 편이 문서로 남고 서로를 가리키며 반년 뒤에 그 위에서 시작할 수 있다. 그래서 이 도구의 강점은 배우는 과정을 체계화하고 지식을 구조적으로 배열하는 데 있다. 배움 자체를 대신해 주지는 않는다.

자기 말로 설명하기

지식이 자기 것이 되었는지 확인하는 방법 가운데 가장 확실한 것은 설명해 보는 것이다. 문서를 보지 않고 다른 사람에게 이 논문이 무엇을 했고 왜 중요한지 말해 보면, 아는 부분과 모르는 부분이 곧바로 갈린다. 말하다가 막히는 대목은 아직 이해하지 못한 것이다. 이 확인은 글로는 잘 되지 않는다. 글은 고쳐 쓸 수 있고 문서를 보면서 쓸 수 있어서 쓰는 동안에는 아는 것처럼 느껴지기 때문이다. 그래서 소리 내어 말해 보는 훈련을 따로 해야 한다. 12장에서 채팅으로 묻는 방식을 주로 쓰는 이유로 든 것과 같은 원리다. 말로 질문을 만드는 동안 자기가 아는 것을 자기 입으로 다시 읽게 되고 그 과정에서 한 번 더 생각하게 된다.

에이전트를 상대로 하는 설명도 훈련이 된다. 이 논문을 이렇게 이해했다고 말해 보고 어디가 어긋나는지 물으면 답이 온다. 다만 여기에는 한계가 있다. 에이전트는 설명이 이상해도 대체로 받아 주고 받아 주는 답에서는 자기가 틀렸다는 것을 알기 어렵다. 사람에게 설명하면 표정에서 먼저 드러난다. 그래서 두 가지를 함께 쓴다. 혼자 정리할 때는 에이전트에게 설명해 보며 다듬고 정리가 되었다고 느끼면 사람에게 설명한다. 두 번째 단계를 빠뜨리면 자기 이해의 구멍을 발견할 기회가 사라진다.

따라해보기

아래 블록은 AI 도구의 채팅창에 그대로 붙여 넣는 프롬프트다. 이해했는지 확인하는 데 쓸 수 있는 요청이 하나 있다. 논문을 넣고 정리를 받은 뒤에 문서를 닫고 시작한다.

< >

- 1.
- 2.
- 3.

세 번째 항목이 이 요청에서 가장 값이 크다. 자기가 원문에서 읽은 것과 스스로 채워 넣은 것을 구분하는 훈련이기 때문이다. 논문을 읽고 나면 원문에 없던 연결을 자기도 모르게 만들어 두는 일이 흔하고 그 연결이 맞을 때도 있고 틀릴 때도 있다. 어느 쪽이든 그것이 자기 해석이라는 것을 알고 있어야 나중에 인용할 때 저자가 하지 않은 말을 저자의 말로 옮기지 않는다. 마지막 줄을 넣는 이유는 격려가 확인을 흐리기 때문이다. 잘 이해했다는 답을 받으면 그 답을 근거로 넘어가게 되는데 이 단계에서는 어긋난 대목을 찾아야 한다.

사람을 만나는 훈련

연구가 사람 사이에서 이루어진다는 점이 이 대목에서 중요해진다. 사람을 만나고 관찰한 것과 모은 것을 상대방에게 주장하고 그 주장이 오가는 과정에서 다음 아젠다와 다음 길이 나온다. 이 과정은 문서로 대체되지 않는다. 발표를 해 보고, 질문을 받고 답하지 못한 질문을 들고 돌아오는 일이 실력을 만든다. 답하지 못한 질문이 특히 값이 큰데, 그 질문은 자기 혼자서는 떠올리지 못했을 각도에서 온다. 그 질문을 받아 적어 두는 습관도 함께 붙이면 좋다. 발표 직후에는 기억나지만 이틀이면 흐려지고 적어 두지 않은 질문은 다시 떠오르지 않는다. 위키의 열린 질문 문서가 그 질문들이 놓일 자리이고 같은 질문이 두 번 나오면 그것이 지금 그 분야가 답하지 못한 자리일 가능성이 높다. 그래서 위키에 무엇을 넣었는지보다 그것을 들고 몇 번 사람 앞에 섰는지가 더 중요할 때가 있다.

지도교수가 바쁘다면 연구실 동료와 하면 된다. 같은 주제를 파는 사람이 아니어도 되고 오히려 다른 주제를 하는 사람에게 설명할 때 더 많이 드러난다. 그 사람이 배경을 모르므로 어디서부터 설명해야 하는지를 자기가 정해야 하고 그 정하는 작업이 자기 이해의 구조를 드러낸다. 실행 방법은 간단하다. 논문 한 편을 넣고 정리한 뒤에 그 내용을 십 분 동안 누군가에게 설명하는 자리를 만든다. 매주 한 번이면 충분하고 정해 두지 않으면 하지 않게 된다. 18장에서 다루는 미팅 준비도 같은 훈련의 연장선이다. 무엇을 결정해 달라고 요청할지 정하고 그 근거를 골라 내는 일이 곧 자기 이해를 정리하는 일이기 때문이다.

발표가 만드는 것

발표를 자주 하라는 말은 흔하지만 왜 그런지는 잘 설명되지 않는다. 이유가 몇 가지 있다. 첫째는 순서를 정해야 한다는 점이다. 문서에는 항목이 나열되어 있어도 되지만 발표에는 순서가 있어야 하고 무엇을 먼저 말할지 정하려면 무엇이 무엇을 떠받치는지를 정해야 한다. 이 정리가 곧 논리의 정리다. 둘째는 분량을 잘라야 한다는 점이다. 십 분에 다 넣을 수 없으므로 무엇을 뺄지 정하게 되고 빼도 되는 것과 빼면 안 되는 것을 가르는 판단이 그 주제에 대한 이해를 그대로 드러낸다. 셋째는 질문을 받는다는 점이다. 자기가 가진 지도에는 없던 각도의 질문이 오고 답하지 못한 질문 하나가 며칠치 읽기의 방향을 정한다.

발표 자료를 만드는 일에도 도구를 쓸 수 있지만 순서를 정하는 일은 넘기지 않는다. 무엇을 먼저 말할지 정해 달라고 하면 그럴듯한 순서가 나오고 그 순서대로 발표하면 자기가 왜 그 순서인지 설명하지 못한다. 질문이 그 대목에 들어오면 답이 막힌다. 그래서 순서와 무엇을 뺄지는 자기가 정하고 정한 뒤에 그 순서가 말이 되는지 물어보는 방식이 낫다. 순서를 정하는 일은 시간이 오래 걸리지 않는다. 슬라이드 네다섯 장의 제목을 자기가 지지하려는 결론으로 달아 보면 대체로 그때 드러난다.

하루 30분의 글쓰기

글쓰기는 따로 다뤄야 한다. 지금은 요약도 초안도 도구가 만들어 주고 만들어진 문장은 문법적으로 완전하고 읽기에 무리가 없다. 그런데 글에는 그 사람이 주장하고 싶은 날카로운 지점이 있고 그 지점은 대신 만들어지지 않는다. 무엇을 앞에 놓고 무엇을 뒤로 미룰지, 어디서 단정하고 어디서 물려설지, 어느 대목을 한 문장으로 끝낼지가 그 사람의 판단이다. 이 판단은 연습으로만 붙는다. 도구가 만든 문장을 고치는 일만 반복하면 고치는 능력은 늘고 처음부터 쓰는 능력은 늘지 않는다. 두 가지가 다른 능력이라는 것을 알아 두는 편이 좋다.

도구가 쓴 문장과 자기가 쓴 문장을 구분해 두는 것도 훈련의 일부다. 초안을 받아 고치는 방식으로 작업하다 보면 어느 문장이 자기 것인지 몇 주면 알 수 없게 되고 그 상태에서는 자기 문체가 무엇인지도 흐려진다. 초안 파일과 자기가 쓴 파일을 따로 두고 나중에 합치는 방식이 이 문제를 줄인다. 한국에서 공부한 경우에는 이 훈련이 더 필요하다. 글쓰기를 따로 배우는 과정이 흔하지 않고 영어로 논문을 쓰는 일은 그 위에 언어가 하나 더 얹힌다. 그래서 시간을 정해 두고 도구 없이 쓰는 훈련을 권한다. 하루 30분이면 된다. 주제는 지금 하고 있는 연구여도 되고 그날 읽은 논문에 대한 생각이어도 된다. 중요한 것은 도구를 켜지 않고 자기 언어로 처음부터 쓰는 것이다. 완성도는 상관없고 고치지 않아도 된다. 이 훈련이 쌓이면 나중에 위키에 정리해 둔 지식과 결합했을 때 훨씬 나은 글이 나온다. 재료가 좋아도 그것을 배열하는 감각이 없으면 글이 되지 않고 그 감각은 자기가 써 본 양에 비례한다.

학습 시간의 배분

사람이 깊이 읽을 논문은 자기 연구의 축에 닿는 몇 편이면 된다. 4장에서 ingest에 등급을 두지 않는다고 한 것은 문서의 품질 기준을 나누지 않는다는 뜻이고 사람이 그 논문을 얼마나 깊이 이해할지는 별개다. 대부분의 논문은 무엇을 했고 어디에 놓이는지를 아는 것으로 충분하다. 어느 논문에 원문까지 내려가 시간을 들일지는 대체로 두 번째로 열게 될 때 드러난다. 한 번 넣고 다시 열지 않는 논문과 자꾸 되돌아가는 논문이 갈리고 뒤쪽이 시간을 들일 대상이다.

이 구분을 처음부터 알 수는 없으므로 순서를 뒤집는 편이 낫다. 넣을 때는 가볍게 넣고 되돌아가게 되면 그때 깊이 읽는다. 되돌아간 횟수는 위키가 알려 준다. 어느 문서를 몇 번 인용했는지, 어느 종합 문서를 몇 번 고쳤는지가 작업 기록에 남기 때문이다. 처음부터 모든 논문을 깊이 읽으려 하면 몇 편 못 가서 지치고 지치면 읽기 자체를 미루게 된다. 학습에 드는 시간을 늘리라는 말이 모든 논문에 균등하게 쓰라는 뜻이 되면 그 말은 실행되지 않는다.

도구를 쓰는 것과 의존하는 것

경계가 애매하게 느껴질 수 있어서 판단 기준을 하나 적어 둔다. 도구가 없으면 그 일을 아예 못 하게 되었는데가다. 정리 문서를 만드는 일은 도구가 훨씬 빠르지만 도구가 없어도 원문을 읽고 정리할 수는 있다. 이 경우는 쓰는 것이다. 어떤 주제에 대해 자기 생각을 말해 보라고 했을 때 문서를 열지 않으면 아무 말도 못 한다면 그것은 의존이다. 앞의 것은 문제가 없고 뒤의 것은 훈련으로 되돌려야 한다. 확인하는 방법은 앞 절의 따라해보기와 같다. 문서를 닫고 말해 보는 것이다.

도구에 의존하게 되는 경우에는 공통점이 있다. 결과가 그럴듯해서 확인할 이유를 느끼지 못했다는 것이다. 요약이 어색하면 원문을 열게 되지만 매끄러우면 그대로 넘긴다. 그래서 확인 절차를 미리 정해 두는 편이 낫다. 논문 한 편마다 원문으로 내려가 확인할 대목을 서너 개 정해 두고 그 서너 개는 예외 없이 화면과 대조한다. 대조할 대상으로는 결론을 좌우하는 수치, 대조군의 구성, 저자가 인정한 한계가 알맞다. 서너 개면 시간이 많이 들지 않고 이 습관이 붙으면 요약을 읽는 태도 자체가 달라진다.

위키가 학습에 기여하는 조건

위키는 학습할 수 있는 구조를 유지해 준다. 첫째는 되짚을 수 있다는 것이다. 반년 전에 읽은 논문에 대해 그때 무엇을 판단했는지가 문서에 있으므로 지금의 판단과 비교할 수 있다. 자기 생각이 어떻게 바뀌었는지를 볼 수 있는 것이 학습에서 드물게 얻는 기회다. 둘째는 빈자리가 보인다는 것이다. 어느 주제에 종합 문서가 없다면 그 주제를 아직 정리하지 못한 것이고 그 사실이 문서 구조에서 드러난다. 머릿속에서는 아는 것과 모르는 것의 경계가 흐리지만 문서에서는 선명하다.

셋째는 같은 것을 두 번 정리하지 않아도 된다는 것이다. 어떤 방법의 한계를 한 번 제대로 정리해 두면 그 뒤로는 그 문서를 가리키면 되고 그 만큼의 시간이 새 주제로 간다. 넷째는 남에게 설명할 재료가 준비되어 있다는 것이다. 발표를 준비할 때 처음부터 찾는 것과 정리된 문서에서 고르는 것은 드는 시간이 다르고 시간이 줄면 발표를 더 자주 하게 된다. 앞 절에서

발표가 실력을 만든다고 했는데 그 발표의 빈도를 올려 주는 것이 이 구조의 기여다. 위키는 학습할 수 있는 상태를 유지해 주고 그 상태에서 무엇을 하는지는 사람의 몫이다.

자기가 만든 지식에 대한 책임

도구가 정보를 요약하고 표현한 뒤의 책임은 사람에게 남는다. 무엇을 연구할지, 그 연구가 누구에게 무엇을 남기는지, 그리고 그 결과에 대한 책임을 누가 지는지는 사람 쪽에 있다. 과학이 인류 전체의 삶을 밀어 올리는 일이 되려면 그 일을 하는 사람에게 내재적인 동기가 있어야 하고 그 동기는 대체로 사람을 향한다. 무엇이 궁금해서 누구를 돕고 싶어서 어떤 문제가 풀리지 않는 것이 답답해서 시작한다. 이 동기는 도구가 만들어 주지 않고 만들어 줄 이유도 없다.

책임은 구체적인 자리에서 나타난다. 위키에 적힌 문장 하나가 나중에 논문의 근거가 되고 그 논문을 읽은 사람이 다음 실험을 설계한다. 확인하지 않은 문장을 적어 두면 그 오류가 그 경로를 따라 퍼진다. 이 책이 확인한 것과 확인하지 못한 것을 나눠 적으라고 반복하는 이유가 여기 있다. 확인하지 못한 것을 확인하지 못했다고 적는 일은 절차이기 이전에 태도이고 그 태도가 없으면 절차는 며칠이면 무너진다. 도구를 쓰는 사람이 늘어날수록 이 태도가 더 중요해진다. 그럴듯한 문장을 만드는 비용이 거의 0이 되었으므로 그 문장이 근거를 가졌는지 확인하는 일이 유일하게 남은 구분점이 된다.

자동화되는 쪽과 남는 쪽

자동화에 맡길 수 있는 일은 자료를 찾고 모으는 일, 원문에서 정해진 항목을 뽑아 형식에 맞추는 일, 여러 문서에서 같은 항목을 모아 나란히 놓는 일, 형식이 어긋난 곳과 빠진 곳을 찾는 일, 그리고 초안을 만드는 일이다. 다섯 가지 모두 양이 많고 규칙이 분명해서 사람이 하면 지친다. 이 다섯 가지가 줄어드는 것만으로도 실제 작업 시간은 크게 줄어든다.

남는 쪽은 다르다. 무엇을 연구할지 정하는 일, 근거가 결론을 떠받치는지 재는 일, 어긋나는 결과가 진짜 모순인지 조건의 차이인지 판정하는 일, 어디까지 말할 수 있는지 강도를 정하는 일, 그리고 그 판단이 틀렸을 때 책임을 지는 일이다. 이 다섯 가지는 앞의 다섯 가지가 아무리 빨라져도 그대로 남는다. 오히려 앞쪽이 빨라지면 뒤쪽에 도달하는 빈도가 늘어서 판단할 일이 많아진다. 학생이 이 도구를 쓰면서 바빠졌다고 느낀다면 그것이 정상이다. 읽는 시간이 줄어든 만큼 판단하는 시간이 늘어난 것이고 판단하는 시간이 늘어난 것이 훈련의 관점에서는 이익이다.

22장. MCP 기본 개념

MCP는 Model Context Protocol의 줄임말이고 AI 에이전트가 바깥 시스템의 자료를 읽고 동작을 시킬 수 있게 하는 공통 규약이다. 파트 1과 2에서 위키는 폴더 안에서 완결되었다. 논문이 들어오고 정리되고 연결되고 산출물이 나오는 순환이 파일 시스템 위에서만 일어났다. 연구책임자의 하루는 그 폴더 밖에서 시작한다. 결정을 요구하는 것들이 메일로 오고 일정은 달력에 있고 학생과의 대화는 메신저에 있고 프로젝트 상태는 팀이 함께 보는 문서에 있고 기관에 낼 서류는 HWP 파일로 온다. 이 자료들을 손으로 위키에 옮겨 적으면 옮긴 그날에만 정확한 사본이 생긴다. 에이전트가 그곳을 직접 읽을 수 있으면 옮겨 적을 필요가 없어지고 위키는 사본 대신 판단을 담는다. 그 연결을 만드는 규약이 MCP다. MCP를 연결하기 전에는 각 시스템에서 무엇을 읽고 어디까지 바꿀 수 있는지를 정해야 한다.

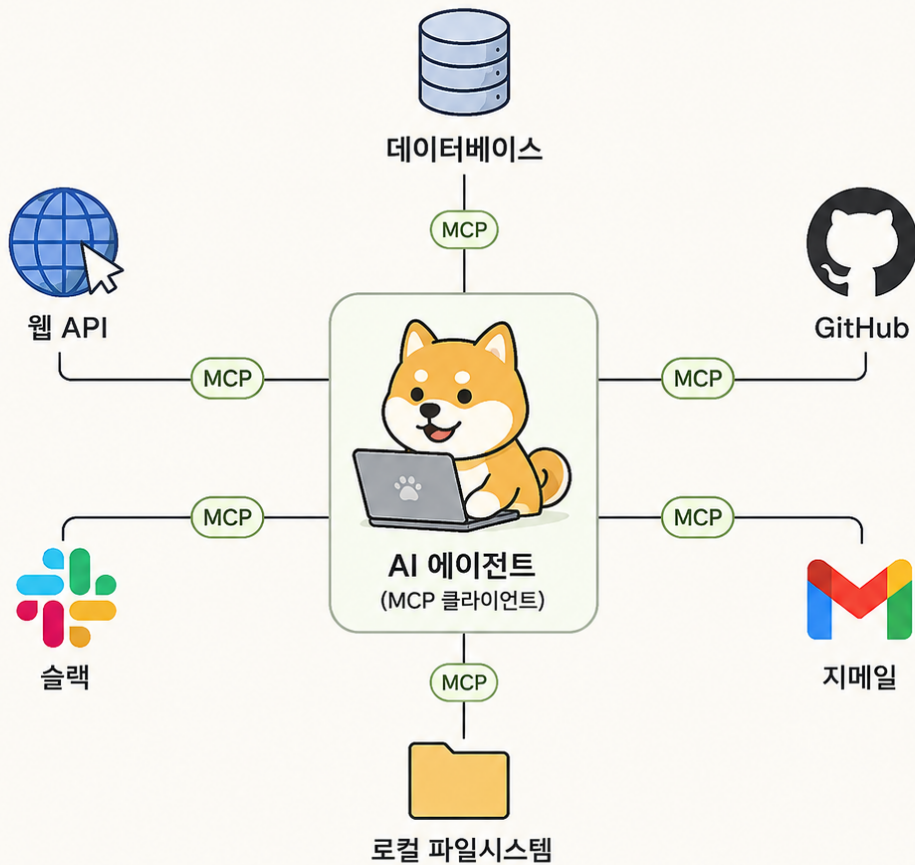
MCP는 에이전트가 데이터베이스, 웹 API, GitHub, Gmail, Slack, 로컬 파일시스템을 공통된 형식으로 호출하게 해 주는 연결 규약이다. 각 연결에서는 읽기와 쓰기의 범위를 따로 정한다.

규약이 연결을 도구에서 떼어 놓는 방식

규약이 있기 전에는 도구마다 연결 방식을 따로 만들어야 했다. 메일을 읽는 기능은 이 도구에만 있고 달력을 읽는 기능은 저 도구에만 있는 식이었고 도구를 바꾸면 연결을 다시 만들어야 했다. 공통 규약이 생기면서 서버가 제공하는 도구와 자료의 표현 방식은 여러 클라이언트가 이해할 수 있게 되었다. 연결한 서버를 다른 에이전트에서 쓰려면 각 클라이언트가 그 MCP 전송 방식과 기능을 지원해야 하고 서버 설정과 인증과 승인 정책도 적용되는 자리에서 갖춰야 한다. Codex의 데스크톱 앱과 명령줄과 편집기 확장은 같은 호스트에서 MCP 설정을 공유하지만 ChatGPT 웹은 플러그인이 제공하는 원격 MCP 도구를 쓰는 별도 경로다. 연결 뒤의 동작은 원자료를 바꾸지 않는 조회와 외부 상태를 바꾸는 쓰기로 나누어 볼 수 있다. 메일함을 검색하는 것은 조회이고 메일을 보내는 것은 쓰기다. 조회와 쓰기의 구분은 13장에서 본 권한의 경계를 바깥 시스템에 그대로 적용한다.

MCP란 무엇인가요?

AI 에이전트가 외부 도구와 데이터를 표준 방식으로 연결할 수 있게 해주는 프로토콜



MCP(Model Context Protocol)는
AI 에이전트가 다양한 도구와 데이터를 안전하고 표준화된 방식으로
연결하고 사용할 수 있도록 돕는 '연결 규약'입니다.

Figure 11: MCP가 AI 에이전트와 외부 도구를 연결하는 구조

연결이 늘어난다고 위키가 저절로 좋아지지는 않는다. 연결은 자료를 가져올 수 있는 통로일 뿐이고 무엇을 가져와 어디에 둘지는 여전히 사람이 정한다. 통로를 열어 두고 규칙을 정하지 않으면 위키에 그날의 메일 내용과 그날의 일정이 그대로 복사되기 시작한다. 그렇게 만들어진 문서는 만든 날에만 정확하다. 메일은 답장이 오면서 바뀌고 일정은 옮겨지는데 복사본은 그대로 남아 있다가 반년 뒤에 틀린 정보로 읽힌다. 그래서 규칙은 하나로 정해 두었다. 사실은 그 사실을 소유한 자리에 두고 위키는 그곳을 가리킨다. 위키가 담는 것은 지금 이 일이 어디까지 왔고 다음에 무엇이 필요한지에 대한 현재 판단이다. 판단을 뒷받침하는 원자료는 메일함과 달력과 팀 문서에 그대로 두고 링크로 잇는다.

연결하기 전에 정하는 경계

MCP 연결에서 실제로 보이는 범위는 서비스의 인증 방식, 요청한 OAuth 범위, MCP 서버가 노출한 도구, 조직의 정책에 따라 달라진다. 어떤 메일 연결은 계정의 넓은 범위를 검색할 수 있고 다른 연결은 정해진 기능이나 공유된 자료만 볼 수 있다. 따라서 연결 이름만 보고 받은 편지함 전체나 모든 일정을 읽는다고 단정하지 않고 인증 화면과 실제 도구 목록을 확인한다. 연구책임자의 계정에는 심사 정보, 인사 관련 내용, 미공개 연구 내용, 학생의 개인 사정이 들어 있으므로 넓은 범위가 열렸다면 그 사실을 작업 경계에 반영해야 한다. 연결을 켜기 전에 첫째로 어떤 자료와 동작이 실제로 노출되는지를 기록한다. 둘째로 그중 무엇을 위키에 남겨도 되는지를 정한다. 읽을 수 있다는 것과 남겨도 된다는 것은 다르고 위키는 나중에 공유되거나 논문 초안 작성에 쓰일 수 있다. 조회는 원래 서비스의 상태를 바꾸지 않지만 조회된 정보의 공개와 모델 처리는 되돌릴 수 없으므로 조회 역시 안전을 보장하지 않는다. 외부 상태를 바꾸는 발송, 게시, 일정 변경은 별도의 승인 경계에 둔다.

내보내는 동작의 절차는 지침 파일에 적어 둔다. 메일의 경우 순서가 정해져 있다. 초안을 만들고 파일로 저장하고 사람에게 내용을 보여주고 기다리고 명시적인 승인을 받은 뒤에 발송한다. 답장해 달라거나 리마인드 해 달라는 말은 이 순서의 앞부분을 지시한 것이고 초안을 만들어 보여주고 기다리라는 뜻이다. 초안을 고쳤으면 고친 것을 다시 보여주고 새 승인을 받는데 고친 대목이 수신자거나 첨부거나 약속의 수위일 수 있어서 앞선 승인이 고친 초안에 적용되지 않기 때문이다. 같은 원칙이 일정과 팀 대화에도 적용된다. 남의 달력에 무언가를 넣거나 옮기는 일, 팀 채널에 메시지를 올리는 일은 되돌리는 비용을 상대가 함께 치른다. 읽기는 열어 두고 내보내기는 매번 확인받는 조합이 연구실 운영에서 실제로 쓸 만한 기본값이다.

연결	읽기에서 얻는 것	내보내기에서 막는 것
메일	무엇이 결정을 기다리는지, 무엇이 마감인지	발송은 초안 제시와 승인 뒤에만
달력	다음 주에 무엇이 있고 무엇을 준비해야 하는지	남의 일정 생성과 변경
팀 대화	어느 논의가 진행 중이고 무엇이 막혀 있는지	채널 게시와 답글
팀 문서 HWP 양식	프로젝트의 현재 상태와 공유 범위 양식의 구조와 이미 채워진 값	공유 설정 변경과 외부 초대 서명 배치와 제출본 생성

메일과 일정

메일 연결에서 실제로 값이 큰 일은 답장을 대신 쓰는 것보다 무엇이 결정을 기다리고 있는지 찾아내는 것이다. 받은 편지함에는 읽고 넘길 것과 지금 판단해야 할 것이 섞여 있고 둘을 가르는 일에 매일 시간이 든다. 에이전트에게 시킬 수 있는 것은 지난 며칠의 메일에서 사람의 결정을 요구하는 것만 골라 목록으로 만들고 각 항목마다 무엇을 결정해야 하는지와 언제까지인지를 한 줄로 적게 하는 일이다. 이 목록은 위키에 남길 대상이 아니고 그날의 작업 목록이다. 목록에서 결정이 내려지면 그 결정만 소유 문서에 반영한다. 메일 본문을 위키에 옮기지 않는 이유가 여기서 분명해진다. 결정은 오래 살고 메일 본문은 그다음 답장에서 바뀐다.

달력 연결은 준비의 시점을 알려 주는 데 쓴다. 다음 주에 어떤 미팅이 있는지 알면 그 미팅에 필요한 자료가 무엇인지를 미리 정리할 수 있고 18장에서 본 대로 미팅 전에 자료가 건네지는지가 그 미팅의 결과를 가른다. 여기서 조심할 것은 일정 자체를 위키에 복사하지 않는 것이다. 일정은 자주 바뀌고 복사본은 바뀌지 않는다. 위키에 남기는 것은 그 미팅에서 무엇을 결정했고 다음 조건이 무엇인지이며 언제 어디서 만나는지는 달력이 소유한다. 일정을 만들거나 옮기는 동작은 열지 않는다. 참석자가 여럿인 일정을 옮기면 그 사람들 전부의 하루가 바뀌고 되돌리려면 다시 전부에게 알려야 한다.

팀 대화

메신저 연결은 연구실 운영에서 성격이 조금 다르다. 메일과 달리 대화가 흐르고 대부분의 메시지는 기록으로 남길 가치가 없으며 그중 일부만 결정이나 약속이다. 그래서 이 연결에서 하는 일은 읽어서 요약하는 것보다 골라내는 것이다. 어느 논의가 결론 없이 늘어지고 있는지, 어떤 요청이 답을 받지 못한 채 지나갔는지, 어느 채널에서 결정이 내려졌는데 그것이 어디에도 기록되지 않았는지를 찾는다. 세 가지 모두 사람이 매일 훑어서는 잘 보이지 않는다. 찾아낸 것 가운데 결정과 약속만 소유 문서로 옮기고 나머지는 그대로 둔다. 대화 전체를 위키로 가져오면 위키가 검색되는 대화 기록이 되고 1장에서 본 문제로 되돌아간다. 메시지를 올리는 동작은 사람이 확인한다. 27장과 28장에서 이 연결을 실제 운영에 붙이고 개인 작업 공간과 연구실 공간을 나눈다.

프로젝트 문서

팀이 함께 보는 문서 도구는 위키가 하지 못하는 일 하나를 한다. 접근권을 사람 단위로 나누는 일이다. 파일 시스템 위의 위키는 그 폴더를 가진 사람만 볼 수 있고 공동연구자에게 그중 일부만 보이게 하는 방법이 마땅치 않다. 프로젝트 문서 도구는 페이지마다 누가 볼 수 있는지를 정할 수 있으므로 밖으로 나가는 자료를 그쪽에 두고 안에서 쓰는 판단은 위키에 둔다. 두 층을 나누는 기준은 수명이다. 공유 페이지는 그 프로젝트가 끝나면 접근권과 함께 닫히고 지식 저장소는 그보다 오래 산다. 같은 논문을 두 곳에 올리게 되는 것처럼 보이지만 실제로 두 번 읽는 일은 없고 두 번 쓰는 일만 있다. 한 번은 몇 해 뒤에 다른 주제를 파는 사람을 위해 쓰고 한 번은 이번 분기에 결정을 내려야 하는 사람을 위해 쓴다. 25장이 이 연결을 프로젝트 관리에 실제로 붙이는 방법을 다룬다.

HWP 양식을 다루는 연결

국내 기관에 내는 서류는 대부분 HWP 파일로 온다. 연구비 신청서, 실적 보고서 각종 동의서와 확인서가 이 형식으로 배포되고 제출도 같은 형식으로 받는다. HWP 계열은 일반적인 문서 도구로 열리지 않고 .docx로 바꿔서 작업하면 표 구조와 서식이 무너져 반려된다. 그래서 이 형식을 직접 다루는 연결이 필요하다. 그 자리는 kordoc이 맡고 저장소가 공개되어 있다. 다루는 형식은 2020년 이후의 HWPX와 예전 이진 형식인 HWP 5.x, 그보다 오래된 HWP 3.x, 그리고 XML 기반의 HWPML까지이며 PDF와 표 계산 파일과 .docx도 함께 읽는다. 이 도구가 하는 일은 크게 셋이다. 문서를 읽어 구조가 살아 있는 형태로 바꾸는 일, 양식의 칸을 원래 서식을 유지한 채 채우는 일, 그리고 결과를 눈으로 확인할 수 있게 그려 주는 일이다.

세부 기능 가운데 실제 작업에서 값이 큰 것 몇 가지를 짚어 둔다. 표의 병합된 칸을 그대로 살려 읽는다는 점이 첫째인데, 행정 양식은 병합 칸이 많고 그것이 무너지면 어느 값이 어느 항목의 것인지 알 수 없게 된다. 둘째는 서식을 유지한 채 양식을 채우는 기능이다. 새로 만드는 것과 원래 문서를 열어 값만 넣는 것은 결과가 다르고 기관은 원본 서식을 바꾸면 반려한다. 셋째는 문서 두 개를 비교해 무엇이 달라졌는지 보여 주는 기능으로, 작업 사본이 원본과 어디가 달라졌는지 확인할 때 쓴다. 넷째는 결과를 그림으로 그려 주는 기능이다. 서명을 붙이거나 칸을 채운 뒤에 파일 구조만 검사하면 서명이 옆 칸의 글자를 덮었는지 알 수 없고 그려 봐야 알 수 있다. 스캔된 PDF에서 한국어 글자를 읽어 내는 기능과 개인정보를 가리는 기능도 함께 있다. 이 도구는 명령줄과 MCP 연결에서 모두 쓸 수 있다.

행정 문서를 다룰 때 지키는 경계는 두 가지다. 원본 양식은 덮어쓰지 않고 작업 사본을 따로 만들며 제출본은 이름으로 구분되게 둔다. 그리고 서명을 배치하는 일과 그 배치를 서명 행위로 인정하는 일을 나눈다. 배치는 도구가 하고 그 파일을 제출해도 되는지는 사람이 열어 보고 정한다. 29장이 이 절차를 서류 종류별로 다룬다.

따라해보기

아래 블록은 AI 도구의 채팅창에 그대로 붙여 넣는 프롬프트다. 연결을 켜 뒤에 처음 시킬 것은 읽기만 하는 요청이다. 무엇이 보이는지 확인한 다음에 범위를 정하는 편이 낫다.

-
-
-

마지막 두 줄이 이 요청의 핵심이다. 초안을 만들지 말라고 적지 않으면 에이전트는 도움이 되려고 답장을 함께 만들어 오고 만들어진 초안이 있으면 읽어 보게 되고 읽다 보면 보내게 된다. 결정 목록을 보는 일과 답장을 쓰는 일은 다른 작업이고 섞으면 무엇을 결정할지 정하기 전에 답장의 문장을 다듬고 있게 된다.

확인하는 순서를 지침 파일에 적기

연결이 여러 개면 무엇부터 읽을지가 문제가 된다. 순서를 정해 두지 않으면 세션마다 다른 곳부터 읽고 그날 처음 읽은 자리의 정보가 답의 뼈대가 된다. 그래서 지침 파일에 확인 순서를 적어 둔다. 아침에 오늘 할 일을 물었을 때 무엇부터 보는지, 어떤 프로젝트의 상태를 물었을 때 어느 문서를 먼저 여는지를 정해 놓는 식이다. 순서를 정하는 기준은 소유다. 소유 문서를 먼저 읽고 그다음에 그 문서가 가리키는 원자료를 본다. 반대로 원자료부터 읽으면 지금까지 내린 판단을 모르는 상태에서 자료를 해석하게 되고 이미 결론이 난 것을 다시 논의하게 된다.

순서를 적어 두면 답의 형태도 안정된다. 같은 질문에 어제와 오늘의 답이 달라지는 원인은 흔히 읽은 순서에 있다. 순서가 고정되면 답이 달라졌을 때 그 차이가 자료의 변화에서 왔다고 볼 수 있다. 이 순서는 위키의 지침 파일과 한곳에 둔다. 연결마다 따로 적으면 새 연결을 켤 때마다 순서가 하나씩 늘어나고 늘어난 순서들이 서로 어긋난다.

연결을 고르고 검증하는 일

MCP 서버는 누구나 만들 수 있고 공개된 것이 많다. 연결을 켜면 그 서버가 에이전트와 외부 서비스 사이에서 자료나 동작을 중개하므로 무엇을 켜지는 보안 판단이 된다. 확인할 것은 배포자의 신원, 소스와 업데이트 경로, 요청하는 인증 범위, 서버가 호출하는 외부 주소, 보관 정책이다. 서비스를 만든 회사가 직접 배포했다는 사실은 배포자의 신원을 확인하기 쉽게 하지만 최소 권한이나 안전을 자동으로 보장하지는 않는다. 소스가 공개되어 있어도 실제 설치한 판본과 실행 중인 설정이 같은지 확인하지 않으면 충분한 검증이 되지 않는다. Codex가 지원하는 로컬 STUDIO 서버는 컴퓨터에서 프로세스로 시작되고 원격 서버는 주소를 통해 접속한다. 이 구분만으로 자료가 컴퓨터 밖으로 나가는지를 판정할 수는 없다. 로컬 프로세스도 외부 API를 호출할 수 있고 도구의 결과는 에이전트가 판단할 수 있도록 클라이언트와 모델 처리 경로로 전달된다. 반대로 조직이 관리하는 원격 서버는 접근 범위와 보관 정책이 명시되어 있을 수 있다. 연구 자료와 학생 정보와 심사 정보를 다룰 때는 로컬이라는 이름 대신 실제 데이터 흐름과 적용되는 서비스 정책을 확인해야 한다.

연결을 켜 뒤에 무엇을 할 수 있는지도 실제로 확인해 두는 편이 낫다. 설명에 적힌 기능과 실제로 노출되는 동작이 다른 경우가 있고 읽기만 한다고 적혀 있는데 쓰기 동작이 함께 들어 있는 경우도 있다. 처음 켜 세션에서 그 연결이 제공하는 동작의 목록을 뽑아 보고, 그중 되돌릴 수 없는 것이 무엇인지 확인한다. 확인한 결과는 적어 둔다. 13장에서 지금 어떤 권한이 열려 있는지를 파일에 적어 두라고 한 것과 같은 이유이고 연결은 그보다 더 읽기 쉽다. 도구를 갱신하면 연결이 제공하는 동작이 늘어나는 경우가 있으므로 갱신 뒤에 한 번 다시 본다.

연결이 늘어날 때 생기는 문제

연결을 여러 개 켜면 같은 정보가 여러 곳에서 보이기 시작한다. 어떤 미팅은 메일에도 있고 달력에도 있고 팀 대화에도 있으며 세 곳의 내용이 조금씩 다르다. 누가 무엇을 소유하는지 정해 두지 않으면 에이전트가 세 곳을 다 읽고 그때그때 다른 답을 준다. 그래서 정보의 종류마다 소유하는 자리를 하나로 정한다. 일정의 시각은 달력이 소유하고 무엇을 하기로 했는지는 팀 문서나 위키가 소유하고 그 결정이 오간 경위는 메일이 소유한다. 소유 자리가 정해지면 확인하는 순서도 정해진다. 지침 파일에 그 순서를 적어 두면 다음 세션의 에이전트가 같은 순서로 확인한다. 순서가 없으면 세션마다 다른 곳부터 읽고 다른 곳부터 읽으면 다른 답이 나온다.

두 번째 문제는 연결이 실제로 살아 있는지가 조용히 달라진다는 것이다. 인증이 만료되거나 도구의 판이 올라가면서 연결이 끊기면, 에이전트는 그곳을 읽지 못한 채로 답을 만든다. 읽지 못했다는 사실이 답에 나타나지 않으면 사람은 그 답을 전체를 본 결과로 읽는다. 그래서 확인하지 못한 것을 이상 없음으로 적지 않는다는 규칙이 여기에도 적용된다. 어느 자리를 읽었고 어느 자리를 읽지 못했는지를 답에 함께 적게 한다. 세 번째는 권한이 넓어지는 방향으로만 움직인다는 것이다. 한 번 열어 준 범위는 좁히기 번거로워서 그대로 남고 몇 달이 지나면 처음에 정한 경계가 무엇이었던지 기억나지 않는다. 지금 어떤 연결이 켜져 있고 각각이 무엇을 할 수 있는지를 목록으로 적어 두고 주기적으로 실제 설정과 대조한다. 27장에서 예약 작업 목록을 같은 방식으로 관리하는데 이유도 같다.

네 번째는 비용이다. 연결마다 요금 체계가 다르고 어떤 것은 계정 자체가 유료이며 어떤 것은 무료 등급에서 조회 횟수가 제한된다. 연구실 규모에서는 큰 금액이 아니지만 자동화를 붙여 매일 돌리기 시작하면 사람이 쓰던 것과 다른 규모로 호출이 일어난다. 예약 작업이 매일 아침 여러 연결을 훑는 구성이라면 그 횟수를 한 번 세어 보고 시작하는 편이 낫다. 교육 기관 할인이 있는 서비스도 있어서 실제 부담이 크게 줄어드는 경우가 있다. 요금과 등급은 공급자의 현재 안내에서 확인하고 자동화를 켜기 전에는 예상 호출 횟수를 세어 본다.

23장. 도반(道伴), 당신의 공동연구자를 지도하는 법

같은 길을 걷는 동반자로서. 도반(道伴)은 한국 불교 전통에서 유래한 말로, 같은 길을 함께 걷는 수행의 동반자를 뜻한다. 본래 영적인 맥락에서 쓰이지만 서로를 지지하고 배움을 나누며 더 높은 목표를 향해 나아가는 사람들을 가리키는 표현이기도 하다. 우리 연구실은 발견과 혁신, 지적 성장의 여정을 함께하는 과학적 동반자들이 모인 곳이다. 각자가 서로의 도반으로서 기여하고 함께 성장한다. 오후에 세 사람과 차례로 면담이 잡혀 있다고 하자. 첫 번째는 공개 데이터 재분석으로 방법을 익히는 학부 연구생 1년차이고 두 번째는 첫 주저자 원고를 준비하는 대학원 3년차이며 세 번째는 독립 과제를 준비하면서 후배도 지도하는 박사후연구원이다. 세 사람이 지금 붙들고 있는 문제는 서로 겹치지 않는다. 앞 면담에서 다음 면담으로 넘어가는 사이에 머릿속의 맥락을 통째로 갈아 끼워야 하고 갈아 끼우는 데 실패하면 면담의 앞부분은 지난번에 무엇을 합의했는지 되묻는 데 쓰이고 되묻는 시간이 길어질수록 오늘 내려야 할 결정은 다음으로 밀린다. 사람에 관한 기록을 한곳에 모으면 이 문제가 풀릴 것 같지만 실제 운영은 반대로 갔다. 학위와 행정 서류의 파일 위치와 처리 상태를 담는 작업 공간이 하나 있고 연구 주제와 담당과 역량과 멘토링 판단은 문헌 위키의 registry가 소유한다. 세 사람은 합성 사례다. 여러 연구실에서 반복해서 나타나는 상황을 합쳐 만들었고 이름 대신 마이클, 스테판, 오타니로 부른다. 사람에 관한 기록에서 실제 인물의 조합을 쓰면 식별 정보를 지워도 당사자는 알아본다. 시스템의 구조와 반복되는 원칙은 실제 운영에서 가져오되 사람과 기관에 관한 것은 합성한다는 규칙을 파트 3 전체에 적용한다.

지도교수는 토요일 아침이 두렵다

토요일 아침은 적막하다. 모처럼 쉬려고 의자에 앉으면 지난 한 주 동안 못다 한 생각들이 스멀스멀 올라온다.

“이 논문은 어떻게 진행되고 있지?”, “실험 결과는 잘 나왔나?”, “분명 랩미팅과 개인 미팅 때 전달받았는데, 왜 기억이 안 나지?”, “그때 스테판 학생이 보낸 자료는 어디에 있지?”

노화가 오는 것이 아니라 내가 노화에게 간다. 최근 들어 기억하지 못하는 일이 너무 많아졌다. 연구도 해야 하고 행정과 잡일도 쌓여 있는데, 아이들도 키워야 하고 빚도 갚아야 한다. 오만 생각을 하다 보면 논문 어느 문장에 무슨 내용을 썼는지도 도무지 기억이 나지 않는다. 논문만 쓰던 그 찬란했던 예전이 좋았다. 그래서 토요일 아침이 두렵다.

“학생에게 카톡을 할까 말까?” “카톡을 보내면 쏸대처럼 보이겠지.”

그러다가도 눈을 질끈 감고 카톡을 보낸다.

“지금 당장 읽어 달라는 것은 아니나... 생각난 김에 남겨 봐요...”

“내가 노화에게 간다.” 출처: YouTube <핑계고>.

지도교수가 맥락을 잃는 일은 기억력과 노화의 문제이기도 하지만, 많은 부분은 저장 위치의 문제이기도 하다. 한 사람의 분석 진행 상황은 그 분석 과제를 소유한 작업 공간에 있고 그 사람이 쓰고 있는 원고의 최신본은 원고를 소유한 작업 공간에 있고 지난 면담에서 정한 다음 할 일은 이메일이나 메신저에 있고 지금 어느 과제에서 어떤 역할을 맡았는지는 사람에 관한 기록이 따로 소유한다. 네 조각을 모아야 다음 대화가 시작되는데 각 공간은 자기가 소유한 질문에만 답하므로 조립은 매번 사람의 머릿속에서 일어난다. 조립에 실패하면 지난번에 무엇을 합의했는지 다시 묻게 되고 상대는 이미 보고했다고 생각한 내용을 처음부터 다시 설명한다. 말은 사람이 여럿이면 조립 실패의 빈도는 사람 수에 비례해 늘고 반복되는 실패는 세 가지 결과를 낳는다. 첫째는 같은 피드백의 반복이다. 지난 면담에서 지적한 것을 기억하지 못하면 이번에도 같은 지적을 하게 되고 받는 쪽에서는 고쳤는데도 같은 말을 듣는 상황이 된다. 둘째는 한 번의 관찰이 오래된 인상으로 굳는 일인데, 어느 시기에 진행이 늦었던 사람은 그 뒤로 얼마나 나아졌든 늦는 사람으로 남고 인상을 뒤집을 근거가 기록되어 있지 않으므로 뒤집힐 기회도 없다. 셋째는 관심의 편중이다. 자주 찾아오고 자주 보고하는 사람의 맥락은 저절로 복원되지만 조용히 일하는 사람의 맥락은 복원되지 않아 결국 지도의 밀도가 달라진다.

목표를 잘못 잡기도 쉽다. 맥락을 잃지 않으려면 더 많이 기록해야 한다고 생각하면 면담 녹취와 메일 사본이 쌓이고 쌓인 만큼 다시 읽지 않게 된다. 필요한 것은 다음 대화를 정확한 지점에서 다시 시작하게 하는 최소한의 상태다. 그 상태는 네 가지로 압축된다. 지금 무엇을 맡고 있는가, 최근에 실제로 만든 것은 무엇인가, 반복해서 필요한 도움은 무엇인가, 다음에



Figure 12: 노화와 기억에 관한 핑계고의 한 장면

무엇을 보면 다음 단계로 넘어갈 수 있는가. 면담의 첫 문장이 지난번에 정한 확인 지점을 묻는 질문이면 정확한 지점에서 시작한 것이다. 지난 면담에서 실행 결과와 실패 원인 세 줄을 다음 확인 지점으로 정했다면 이번 면담은 그 세 줄을 앞에 놓고 시작한다. 세 줄이 없으면 왜 없는지가 첫 안건이 되고 있으면 그 내용이 첫 안건이 된다. 어느 쪽이든 면담은 결정할 것을 이미 앞에 두고 출발한다.



Figure 13: 도반의 맥락을 읽고 관찰과 해석과 지도 행동을 나누는 흐름

면담 전에는 이메일, 원고, 분석 상황, 역할 기록을 모아 시작점을 복원한다. 면담에서 생긴 관찰, 해석, 지도 행동은 네 칸으로 나누고 각 정보는 owner 문서로 되돌린다.

학생별 기록의 여섯 층

어떤 사실에 대해 현재 값을 소유하는 문서를 하나로 정하고 나머지 문서는 그 문서를 가리키게 하는 규칙을 SSOT(single source of truth)라고 부른다. 우리말로는 정본에 해당한다. 그 사실을 소유한 문서를 그 사실의 owner라고 부른다. 규칙이 필요한 이유는 같은 사실이 두 곳에 적히는 순간 한쪽만 고쳐지기 때문이다. 어떤 사람의 현재 역할이 면담 노트와 과제 문서 양쪽에 적혀 있고 역할이 바뀌었을 때 한쪽만 고쳤다면 그 뒤로는 어느 쪽이 맞는지 아무도 모른다. SSOT와 owner의 정의는 이 장이 소유하고 이후 장에서는 각자의 맥락에 적용만 한다. 학생별 기록을 한 파일에 모아놓으면 반대 방향의 문제가 생긴다. 연락처, 현재 역할, 합류 계획, 최근 산출물, 지도하면서 관찰한 것, 원고 이력이 한 문서에 섞여 있으면 그 문서는 사람에 관한 신상 자료가 되고 목적이 다른 정보가 한 자리에 모이면 보존 기간도 접근 권한도 정할 수 없다. 연락처는 오래 유지해야 하지만 지도하면서 세운 잠정적 해석은 근거가 바뀌면 지워야 하고 원고 이력은 과제가 끝나면 과제 쪽에 남아야 한다. 층을 나누는 기준은 그 정보를 언제 고치고 언제 지우는가이고 실제 운영의 registry는 여섯 층으로 갈라져 있다.

기록 층	담는 것	담지 않는 것
연락처와 식별 정보	검증된 연락처와 공식 소속	성격 평가와 과제에 대한 해석

기록 총	담는 것	담지 않는 것
현재 담당	지금 어느 과제에서 어떤 역할인지, 누가 owner이고 누가 지원인지	지난 역할을 현재 책임처럼 굳힌 정보
새 과제 인선 판단	후보가 맡을 연구 갈래, 필요한 지원, 확인한 근거	최근 산출물에서 확인되지 않은 인상
신규 합류 계획	처음 얼마간의 읽기, 환경 설정, 작은 산출물	모든 사람에게 똑같이 적용하는 획일적 일정
개인별 근거 기록	지금 읽는 것, 지금 하는 일, 익힌 방법, 날짜가 붙은 근거	확인되지 않은 소문과 사생활 추정
작업 습관과 멘토링 개입	반복 관찰, 배정할 때의 주의점, 시도한 개입	진단, 낙인, 고정된 인격 묘사

여섯 층이 문헌 위키 안에 있는 데는 이유가 있다. 사람에 관한 판단은 그 사람이 무엇을 읽고 무엇을 만들었는지에 붙어 있고 논문과 연구 주제는 이미 그 위키가 소유한다. 반면 학위 서류와 행정 서류는 파일이 어디 있고 처리가 어디까지 갔지만 알면 되므로 별도의 작업 공간이 맡는다. 그 서류 작업 공간의 규칙 문서는 전체 작업 공간 가운데 가장 얇고 연구 지도에 관한 판단은 여기에 두지 않는다고 스스로 문서에 선언한다. 분석 작업 공간은 분석을, 원고 작업 공간은 원고를 정본으로 관리하고 학생 지도와 달는 접점은 다음 사람이 이어받을 수 있게 넘기라는 인계 문서 규칙과 원고 로그에 남기는 확인 항목뿐이다. 사람에 관한 기록을 성격에 따라 갈라 두면 각 층이 서로 다른 속도로 갱신되고 서로 다른 규칙으로 지워진다. 갈라 두지 않으면 가장 민감한 층의 보존 규칙이 나머지 전부에 적용되거나 가장 느슨한 층의 접근 범위가 나머지 전부에 적용된다. 지도 중인 구성원 한 명을 골라 30분 안에 문서 하나를 만들어 보면 어느 층이 비어 있는지 드러나는데 실제 이름 대신 임의의 식별자를 쓰고 문서는 자기 작업 공간 안에 둔다.

1. 지금 맡은 과제와 역할을 한 줄로 적고 그 사실의 owner가 어느 기록인지 함께 표시한다
2. 확실하지 않으면 확실하지 않다고 적는다

이메일과 메신저와 면담 노트는 증거가 될 수 있지만 그 자체를 현재 상태로 삼지 않는다. 메일에 적힌 역할은 그 메일을 쓴 시점의 역할이고 그 뒤에 바뀌었는지는 메일이 알려 주지 않는다. 원자료는 무슨 일이 있었는지를 증명하고 owner 문서는 지금 무엇이 참인지를 말한다. 두 층을 섞으면 가장 최근에 읽은 메일이 현재 상태가 된다. 파트 3 전체가 반복하는 순환은 흘러진 원자료에서 출발해 owner를 식별하고 에이전트가 모으고 빠진 것을 찾고 사람이 판단하고 판단의 결과를 다시 owner 문서로 되돌리는 형태다. 여섯 단계 중에서 사람이 반드시 말아야 하는 단계는 네 번째와 다섯 번째다. 앞의 세 단계는 규칙이 분명하고 양이 많아 에이전트가 하는 편이 정확하고 마지막 단계는 사람이 정한 결과를 받아 적는 일이므로 초안을 맡길 수 있다. 순환이 한 바퀴 돌 때마다 owner 문서가 갱신되고 갱신된 문서가 다음 바퀴의 출발점이 된다. 순환이 끊기는 자리는 대체로 마지막 단계인데, 결정은 났지만 어디에도 적히지 않아 다음 바퀴가 다시 첫 단계에서 시작한다.

개인별 기록에 남기는 것

개인별 기록의 형식도 고정되어 있다. 지금 읽고 있는 것, 지금 하는 일, 익힌 방법, 날짜가 붙은 근거 기록, 배정할 때의 주의점, 확인이 필요한 항목으로 이루어진다. 최근에 그 사람이 실제로 만든 산출물은 세 개까지 적고 각각 언제 확인했는지를 함께 남긴다. 근거로 삼는 것은 확인할 수 있는 산출물이어야 한다. 논문을 읽었다는 기록보다 그 논문의 핵심 질문을 자기 말로 설명한 한 장짜리 문서가 근거가 되고 방법을 배웠다라는 기록보다 자기가 정한 조건에서 돌린 결과와 실패 원인이 근거가 된다. 마이클의 다음 확인 지점은 재분석 결과를 한 장짜리 질문 문서로 만드는 것이며, 문서가 나오면 재분석 결과를 연구 질문으로 바꾸는 과정을 확인할 수 있다. 확인할 산출물이 없으면 해당 항목을 비워 둔다. 협업이 원만하다거나 글쓰기가 아직 부족하다는 식의 문장은 어느 산출물에서 나왔는지 물으면 답할 수 없고 역지로 채운 그런 서술은 나중에 근거 없이 인용된다. 오래된 관찰을 현재 상태로 이어 쓰지 않으려고 기록에는 신선도 표시를 붙이고 일정 기간이 지나면 항목을 재검토 필요 상태로 바꾼다. 해당 항목은 다시 확인한 근거가 생겨야 현재 상태로 갱신한다. 반년 전에 확인된 것은 반년 전의 상태이고 그 뒤로 근거가 갱신되지 않았다면 지금은 모른다고 적는 편이 정확하다.

관찰과 해석과 지도 행동의 분리

칸	묻는 것	마이클의 경우
관찰	언제 어떤 산출물이나 행동을 확인했는가	재분석 절차를 끝내고 자기 방식으로 그린 분석 흐름도를 가져왔다
해석	그것이 무엇을 시사할 수 있는가	절차의 이해는 진전됐지만 결과를 질문으로 되돌리는 경험은 아직 확인되지 않았다
지도 행동	어떤 발판을 놓을 것인가	재분석 결과에서 해석이 갈리는 지점을 하나 골라 질문 형태로 쓰게 한다
다음 확인 지점	무엇을 보면 넘어갈 수 있는가	한 장짜리 질문 문서와 그 질문을 구분할 분석 한 줄

지도 기록의 최소 단위는 네 칸이다. 무엇을 보았는가, 그것이 무엇을 시사할 수 있는가, 그래서 어떤 발판을 놓을 것인가, 무엇을 보면 다음 단계로 넘어갈 수 있는가. 네 칸을 한 문장에 붙여 쓰면 관찰과 해석이 섞이므로 칸을 나눈 채로 적는다. 나누는 데 드는 수고는 한 줄에 몇 초 정도이고 나누지 않아서 생기는 비용은 몇 달 뒤에 청구된다. 관찰과 해석을 붙여 쓰면 해석이 사실의 지위를 얻는다. 흐름도를 가져왔다는 관찰은 다시 확인할 수 있지만 이해가 진전됐다는 해석은 관찰자의 판단이고 틀릴 수 있다. 두 칸을 나누어 두면 나중에 해석만 교체할 수 있고 관찰은 그대로 남아 다음 해석의 재료가 된다. 붙여 쓴 기록은 해석이 틀렸을 때 관찰까지 함께 버리게 만든다. 사람에 관한 기록에서 그 손실은 특히 큰데, 관찰은 그 사람이 실제로 한 일이고 지워지면 그 사람의 이력에서 사라지기 때문이다.

해석에는 언제 버릴지를 함께 적는다. 절차 이해는 됐지만 실행 경험이 확인되지 않았다는 해석이라면 실행 결과와 실패 원인을 스스로 정리해 오면 이 해석을 버린다고 적어 둔다. 확인 항목에는 다섯 개의 상태 라벨 중 하나를 붙인다. 확인됨, 불일치, 미확인, 보류, 재실행 필요하다. 라벨을 다섯 개로 벌려 둔 이유는 확인하지 못한 것을 이상 없음으로 적지 않기 위해서다. 시간이 없어 보지 못한 항목과 봤는데 어긋난 항목은 다음 면담에서 다르게 다뤄야 하는데 라벨이 둘뿐이면 두 상황이 미확인 하나로 뭉개진다. 네 칸에서 에이전트가 할 수 있는 일과 사람이 해야 하는 일도 갈린다. 흩어진 날짜와 산출물을 모아 관찰 칸의 후보를 만드는 일, 같은 기간에 서로 어긋나는 기록이 있으면 표시하는 일, 지도 행동은 적혀 있는데 확인 지점이 비어 있는 줄을 찾아내는 일은 에이전트가 한다. 해석이 타당한지, 개입을 얼마나 강하게 할지, 지금 어느 축의 발판이 필요한지는 사람이 정한다. 경계를 정하는 기준은 결과를 감당하는 쪽이 누구인가이고 해석이 틀렸을 때 곤란해지는 쪽은 기록된 사람과 지도하는 사람이다.

1. 반복해서 필요했던 도움을 한 줄로 적고 그 줄이 관찰인지 해석인지 표시한다
2. 해석으로 표시한 항목마다 무엇을 보면 그 해석을 버릴지 한 줄로 적는다
3. 아직 닫히지 않은 확인 항목을 나열하고 각각에 확인됨, 불일치, 미확인, 보류, 재실행 필요 중 하나를 붙인다

미팅 전 현재 상태의 복원

면담 준비는 상태를 다시 조립하는 일이다. 조립에 필요한 조각은 네 개다. 지금 맡은 과제와 그 안에서의 역할, 직전 면담에서 내린 결정, 그 뒤로 실제로 만들어진 산출물, 아직 닫히지 않은 질문. 네 조각은 서로 다른 작업 공간에 있고 각각 다른 문서가 owner다. 어느 공간에 들어가든 읽는 순서는 같다. 얇은 지침 파일에서 강제 규칙과 어디로 갈지를 읽고 지도 문서에서 무엇이 어디에 있는지 확인하고 변경 이력에서 최근에 무엇이 바뀌었는지 보고, 마지막에 해당 owner 문서를 연다. 어느 공간에 들어갈지를 작업 문구나 날짜나 인접한 폴더 이름으로 추측하지 않는다. 확인한 뒤에도 모호하면 묻고 넘어가는데 사람에 관한 기록에서 잘못된 공간을 열면 다른 사람의 맥락을 그 사람의 것으로 읽게 된다. 여러 공간을 읽더라도 이번 작업의 주인은 하나다. 준비 문서를 만드는 일이면 그 문서를 소유한 공간이 파일명과 배치와 완료 기준을 정하고 나머지 공간은 근거와 원자료와 제약만 공급한다.

1. 다음 면담의 목적을 한 문장으로 쓰고 확인 가능한 다음 지점을 하나 정한다
2. 문서를 다시 읽으며 확인 시점이 오래된 항목에 재검토 필요 표시를 붙인다

조립한 다음에 할 일은 사실을 두 종류로 나누는 것이다. 기록만으로 확인되는 사실과 본인에게 물어야 확인되는 사실은 면담에서 다르게 다뤄야 한다. 기록에 있는 것을 면담에서 다시 물으면 시간이 낭비되고 기록에 없는 것을 있다고 가정하면 잘못된 전제 위에서 결정이 내려진다. 서로 어긋나는 기록이 나왔을 때 에이전트가 그럴듯한 쪽을 골라 하나로 합치게 두면 안 되고 충돌은 충돌인 채로 남겨 면담에서 확인할 질문으로 바꾼다. 면담의 목적도 하나로 좁혀 둔다. 진행 상황을 보고받는 것인지, 과학적 결정을 내리는 것인지, 막힌 문제를 푸는 것인지, 원고에 피드백을 주는 것인지에 따라 미리 읽을

것과 부를 사람과 걸리는 시간이 달라진다. 넷을 한 시간 안에 모두 다루려 하면 대체로 진행 보고만 남고 결정은 다음으로 밀린다. 조립한 준비 문서는 한 장을 넘기지 않는다. 들어가는 것은 지금 맡은 과제와 역할 한 줄, 직전 면담의 결정과 그 뒤에 확인된 산출물, 오늘 내려야 할 결정 한 줄, 확인이 필요한 질문 몇 줄이고 각 항목이 어느 owner에서 왔는지도 함께 표시한다. 변경 이력은 최근 결정을 복원하는 데 쓰고 현재 값을 정하는 데 쓰지 않는데 로그는 그때 무엇을 했는지를 말할 뿐 지금 무엇이 참인지를 말하지 않기 때문이다. 세션을 새로 열 때마다 지침과 registry를 다시 읽고 이전 세션의 요약에 이어 쓰지 않는 규칙도 같은 이유에서 나왔다. 준비 문서가 제 몫을 하는지는 새 세션을 시작한 에이전트에게 그 문서만 주었을 때 다음 면담에서 확인해야 할 질문 목록과 아직 근거가 없는 항목을 골라낼 수 있는가로 정해진다. 해석을 버릴 조건을 적는 줄이 비어 있으면 기준을 통과하지 못한다. 반증 조건이 없는 해석은 에이전트가 사실로 취급하고 사실로 취급된 해석은 다음 요약에서 더 단정적인 문장이 되어 돌아온다.

미팅 후에 남길 것과 남기지 않을 것

면담이 끝나면 그때 작업 폴더에 메모 파일을 하나 만들고 싶어진다. 실제 운영은 그것을 명시적으로 금지한다. 진행 메모, 변경 기록, 피드백 정리를 작업 폴더 루트에 만들지 않고 위키의 로그 문서에 덧붙인다. 금지하는 이유는 두 가지다. 작업 폴더는 지금 무엇을 제출하는지가 몇 초 안에 보여야 하는 자리인데 메모가 쌓이면 산출물이 그 사이에 묻히고 로그가 풀터마다 흩어지면 한 사람의 이력을 시간축으로 읽을 수 없게 된다. 로그 항목의 형식도 고정되어 있다. 무엇이 이 작업을 촉발했는지, 그때 원본이 어떤 상태였는지, 무엇을 실행했는지, 무엇을 확인했는지, 무엇을 발견했는지, 무엇을 결정했는지, 무엇이 남았는지, 어떤 산출물이 나왔는지를 적는다. 여덟 항목 가운데 면담 기록에서 가장 자주 빠지는 것은 무엇을 확인했는지와 무엇이 남았는지다. 두 항목이 빠진 기록은 결정만 남기고 결정만 남은 기록은 다음 면담에서 무엇을 물어야 할지 알려 주지 않는다.

나온 내용은 성격에 따라 다른 owner로 간다. 과제의 결정과 할 일은 과제를 소유한 공간의 로그로 가고 원고에 관한 결정은 원고 쪽으로 가며 사람에 관한 기록에는 장기적으로 다시 필요할 것만 남는다. 같은 내용을 세 곳에 적으면 세 곳이 서로 다른 속도로 낡는다. 사람에 관한 기록에 남길 것은 네 가지로 좁힌다. 바뀐 역할, 이번에 확인된 기술, 성장의 근거가 되는 산출물, 아직 닫히지 않은 확인 항목이다. 일회성 실수와 감정이 오간 대화는 남기지 않는데 남기면 그 사람의 특성처럼 읽히기 때문이다. 어떤 항목을 남길지 망설여질 때 던지는 질문은 하나다. 반년 뒤에 이 사람의 다음 발판을 정할 때 이 항목이 필요한가. 결정을 적을 때는 무엇을 근거로 그렇게 정했는지 한 줄을 붙이는데 근거가 없으면 몇 달 뒤에 같은 논의가 다시 열리고 대체로 처음과 같은 결론에 같은 시간을 들여 도달한다. 뒤집힌 결정도 지우지 않고 뒤집혔다고 표시한 채 남기고 할 일에는 누가 하는지와 무엇을 내놓는지와 언제까지인지와 무엇을 보면 끝난 것으로 볼 것인지를 함께 붙인다.

분석을 소유한 작업 공간은 이 규칙을 최상위 배치로 굳혀 두었다. 규칙 파일 하나와 위키 하나가 위에 있고 그 아래로 프로젝트 폴더가 작업 단위로 늘어선다.

```
Projects/
  AGENTS.md
  wiki/
    index.md
    conventions.md
    references.md
    projects.md
    lint.md
    log.md
    project-guides/
    project-logs/
  registry
  { }/
```

폴더 이름은 실제 운영에서 쓰는 것을 옮긴 예시이고 독자는 자기 업무에 맞는 이름을 쓰면 된다. 이 배치가 다른 작업 공간과 갈라지는 자리는 마지막 두 줄이다. 프로젝트별 가이드는 그 프로젝트의 세션이 무엇부터 읽어야 하는지를 담고 프로젝트별 로그는 그 프로젝트에서 오간 대화와 작업을 덧붙이기만 한다. 이 작업 공간은 입력 자료와 스크립트와 결과와 원고에 더해 사용자와 나눈 대화를 원자료 층에 넣고 그 대화를 전랑 로그로 남긴다. 대화를 원자료로 두면 그 안의 판단은 아직 검증되지 않은 상태로 남고 검증을 통과한 결론만 프로젝트 문서로 올라간다. 로그는 프로젝트에 속한 것과 프로젝트와 무관한 것으로 갈리고 어느 쪽도 이미 적힌 줄을 고치지 않으며 분석 폴더 루트에는 로그 파일을 만들지 않는다. 공용으로 쓰는 자료는 정보 한 곳에서 경로로 읽고 프로젝트로 복사하지 않는데 동기화 서비스가 symlink를 사본으로 평탄화하므로 중복을 막는 수단이 정보와 경로 참조밖에 남지 않기 때문이다. 위키가 담는 것은 무엇이 어디에 있고

어떻게 일하는지까지이고 분석의 결론은 각 프로젝트 문서가 소유한다. 프로젝트가 커지면 그 폴더가 자기 규칙 문서와 자기 위키를 갖고 프로젝트 안의 라우터가 오히려 혼란을 만드는 경우에는 중앙의 프로젝트별 가이드만 두고 로컬 문서를 두지 않는다.

따라해보기

아래 블록은 AI 도구의 채팅창에 그대로 붙여 넣는 프롬프트다. 분석 작업 공간을 처음 세울 때 쓰는 프롬프트다. 꺾쇠 안은 자기 경로와 이름으로 바꾼다.

```
< >
- AGENTS.md wiki/
- wiki/ index.md, conventions.md, references.md, projects.md, log.md
- wiki/project-guides/ wiki/project-logs/
- index.md
- AGENTS.md
  1. < >
  2. , ,
  3. wiki/project-logs/
  4. AGENTS.md, wiki/index.md,
-
```

원고를 함께 쓰는 순환

원고 지도는 되돌아가기 쉬운 작업이다. 같은 지적이 여러 번 나오고 고쳐졌는지 확인되지 않은 채 다음 판이 오고 구두로 합의한 것이 다음 판에 반영되지 않는다. 스테판은 첫 주저자 원고를 준비하는 단계이고 반복해서 필요한 도움은 결과를 나열하는 것과 주장을 구성하는 것의 구분이다. 결과가 많은 원고는 대체로 그림 순서가 결과를 얻은 순서를 따라가고 주장의 순서와 어긋난다. 지도가 문장 다듬기로 흘러가면 어긋남은 그대로 남는다. 한 판을 도는 순서는 여섯 단계다.

1. 현재 원고 판본과 그림이 말하는 이야기를 확인한다.
2. 이번 판에서 바꿀 주장, 그림, 분석의 범위를 먼저 고정한다.
3. 문장 취향을 빼고 피드백을 근거, 논리, 담당, 기한으로 분류한다.
4. 새 판본과 변경 요약을 함께 받는다.
5. 지난 요구가 실제로 해결되었는지 두 판본을 나란히 놓고 비교한다.
6. 같은 병목이 반복해서 확인될 때만 사람에게 관한 기록의 멘토링 층을 갱신한다.

세 번째 단계가 지도의 질을 가른다. 원고가 부족하다는 말은 다음에 무엇을 해야 할지 정해 주지 않는다. 두 번째 그림의 결과가 서론의 중심 주장을 직접 지지하지 않으므로 주장을 좁히거나 분석을 하나 더한다는 말은 다음 행동을 정한다. 피드백을 근거와 논리로 나누면 어느 쪽이 부족한지가 드러나는데 근거가 부족하면 분석을 더하고 논리가 부족하면 주장을 고친다. 스테판의 다음 확인 지점은 주장과 증거를 대응시킨 지도의 초안인데, 19장에서 다룬 대응 관계를 원고 단위로 적어 두면 세 번째 단계가 훨씬 빨라진다. 두 번째 단계를 건너뛰면 한 판에서 모든 것을 고치려 하게 되고 주장과 그림과 분석을 한꺼번에 손대면 무엇 때문에 나아졌는지 알 수 없다. 고정된 범위 밖에서 발견한 문제는 그때 고치지 않고 열린 항목으로 적어 두는데 적어 두지 않으면 다음 판에서 새로 발견된 것처럼 다시 나온다. 같은 지적이 반복되는 원인 중 상당수가 여기에 있다.

원고를 소유한 작업 공간은 원고만 정보로 관리한다. 그 공간이 학생 지도와 닿는 지점은 두 개뿐인데, 다음 사람이 이어받을 수 있는 형태로 넘기라는 인계 문서 규칙과 원고 로그에 남는 확인 항목이다. 대신 그 공간은 판본을 다루는 규칙을 촘촘하게 갖는다. 받은 원본은 작업 단위 폴더 안의 원본 폴더에 두고 덮어쓰지 않으며 파생본은 새 이름으로 만든다. 지금 제출하는 판본은 작업 폴더의 루트에 두고 초안 폴더 아래에 숨기지 않으며 새 판본이 나오면 직전 판본을 같은 레벨의 지난 판본 폴더로 밀어내고 중간 산출물은 그 아래나 작업 공간 밖의 임시 폴더로 보낸다. 규칙이 이렇게 촘촘한 이유는 되돌리기 수단이 동기화 서비스의 버전 기록뿐이기 때문이다. 이 전제가 규칙 전체를 지배해서 덮어쓰기 전에 먼저 읽고 무엇을 바꿀지 보이고 승인을 받는 순서가 모든 작업 공간에서 같다. 학생이 보낸 판본을 열어 그대로 고쳐 저장하는 습관은 이 순서를 통째로 건너뛴다.

학생마다 다른 순환의 설계

세 사람에게 같은 방식을 적용하면 셋 중 하나에게만 맞는다. 마이클에게 필요한 것은 무엇을 왜 하는지의 연결이고 스테판에게 필요한 것은 결과 나열과 주장 구성의 구분이며 오타니에게 필요한 것은 연구 범위를 스스로 정하고 시간을 배분하는 훈련이다. 세 가지는 지금 무엇이 병목인가라는 축에서 갈리고 같은 위키와 같은 에이전트를 쓰더라도 확인 지점의 모양이 달라져야 한다. 반복되는 병목의 모양에 따라 처방이 갈린다. 완성될 때까지 공유하지 않는 사람에게 완성분을 요구하면 공유 주기가 더 길어지므로 작은 중간 산출물과 짧은 기한을 둔다. 분석은 빠르는데 해석이 약한 사람에게는 결과표를 하나 더 요구하는 대신 그 결과를 설명할 수 있는 대안 세 가지와 그 셋을 구분할 분석을 요구한다. 읽기는 많은데 실행이 늦은 사람에게는 읽을 목록을 늘리지 않고 실제로 돌아가는 산출물을 다음 확인 지점으로 둔다. 세 처방의 공통점은 요구의 양을 바꾸지 않고 요구의 종류를 바꾼다는 데 있다.

오타니처럼 독립을 준비하는 단계에서는 병목의 성격이 달라진다. 할 수 있는 일이 여럿인데 무엇을 하지 않을지 정하지 못해 막힌다. 여기에 필요한 발판은 범위를 좁히는 기준을 함께 세우는 것이다. 확인 가능한 결과가 나올 수 있는 크기인지, 그 결과가 독립 과제의 첫 목표를 지지하는지, 지금의 자원으로 감당할 수 있는지 같은 질문을 두고 후보를 줄인다. 후배 지도에 쓰는 시간과 자기 과제에 쓰는 시간의 배분도 같은 방식으로 정하는데 감각에 맡기면 대체로 눈앞의 요청이 이긴다. 반복되는 침묵을 인격 특성으로 단정하지 않는다. 연락이 줄어드는 데에는 과제가 너무 큰 경우, 무엇이 맞는지 몰라 보고할 것이 없다고 느끼는 경우, 피드백 채널이 불편한 경우, 일정이 다른 일과 충돌하는 경우가 모두 포함된다. 네 가지를 먼저 점검하지 않고 성향으로 설명하면 개입이 엉뚱한 곳으로 간다. 점검은 과제 크기부터 시작하는 편이 나운데 크기를 줄이는 개입이 가장 빠르게 결과를 확인시켜 주고 결과가 확인되면 나머지 세 가지 중 무엇이 남았는지도 좁혀지기 때문이다.

합류에서 독립 연구자로

review

↓

↓

↓

↓

↓

↓

세 카드를 한 사람의 시간축에 놓으면 하나의 경로가 된다. 실제로는 사람마다 순서가 다르고 건너뛰는 단계도 있지만 다음 단계를 정할 때 참고할 뼈대는 만들어진다. 앞의 두 단계는 15장과 17장이 다루는 분야 지도와 방법 문서가 그대로 산출물이 되고 가운데 세 단계는 설계 문서와 실행 결과와 미팅에서의 설명이 산출물이다. 마지막 두 단계에서 확인되는 것은 다른 사람이 이어받을 수 있는 형태로 넘겼는가다. 오타니가 독립 과제 준비와 후배 지도를 함께 맡고 있는 이유가 여기에 있는데 넘길 수 있게 정리하는 능력과 독립 과제를 운영하는 능력은 같은 뿌리에서 나온다. 과제가 커지면 그 과제는 자기 지침 파일과 자기 위키를 갖는 하나의 작업 공간이 된다. 실제 운영에는 그런 중첩이 여럿 있고 상위 규칙은 하위에 자체 규칙이 있으면 그 안에서 하위가 우선한다고 못 박는다. 독립을 준비하는 사람에게 자기 공간의 규칙을 직접 쓰게 하는 것이 그 자체로 훈련이 되는 이유가 여기에 있다.

마지막 단계의 인수인계는 별도의 산출물로 만든다. 넘기는 사람이 머릿속에 있는 것을 말로 설명하고 끝내면 받는 사람은 몇 달 뒤에 같은 질문을 다시 하게 되고 그때는 넘긴 사람이 이미 답할 수 없는 자리에 있다. 문서에 들어가야 하는 것은 과제의 목적, 정보 데이터의 위치, 어떤 대상을 어떤 기준으로 골랐는지, 분석 절차와 실행 방법, 지금까지의 결과와 그 해석, 아직 닫히지 않은 문제다. 인수인계 문서가 실제로 작동하는지 확인하는 방법은 하나인데, 받는 사람이 그 문서만 보고 마지막 분석을 다시 돌릴 수 있으면 된다. 성장의 기준을 지시를 덜 받는 것으로만 정의하면 잘못된 방향을 보상하게 된다. 묻지 않고 혼자 진행하는 것과 스스로 판단할 수 있는 것은 다르고 앞의 것은 대체로 확인 지점을 건너뛰는 결과다. 기준에 놓아야 할 것은 근거를 제시하는가, 불확실한 부분을 불확실하다고 말하는가, 다른 사람이 이어받을 수 있게 넘기는가이고 세 가지는 모두 산출물로 확인되므로 인상에 의존하지 않는다. 독립성이 자라면 지도의 내용이 절차에서 판단으로 옮겨가고 지도가 필요 없어지는 상태는 오지 않는다.

팀의 기억과 개인의 존엄

사람에 관한 기록은 만들기 전에 두 가지를 정한다. 누가 읽을 수 있는가와 언제 지우는가. 두 가지를 정하지 않은 기록은 시간이 지나면 지울 수 없게 되는데 왜 만들었는지 잊히면 지워도 되는지 판단할 근거도 함께 사라지기 때문이다. 기록의 목적은 성장 지원과 작업 연속성이고 목적에 필요하지 않은 정보는 처음부터 넣지 않는다. 건강, 가족, 심리 상태에 관한 정보는 연구 수행을 설명하는 근거로 쓰지 않는다. 진행이 늦은 이유를 개인적 사정으로 추정해 적으면 추정이 사실처럼 남고 당사자가 그 기록을 볼 수 없는 상태에서 계속 인용된다. 사정을 알게 되는 일 자체는 피할 수 없지만 알게 된 것을 지도 기록에 축적하는 일은 선택이다. 기록에는 합의된 일정 조정처럼 작업에 관한 사실만 남긴다. 심리 상태에 대한 판단은 지도교수의 전문 영역이 아니고 문서로 남기는 순간 전문가의 판단처럼 읽힌다. 넣지 않기로 정한 항목의 목록은 짧게 적어 작업 공간의 지침 파일에 넣는다.

내부 기록의 문장을 밖으로 나가는 글에 그대로 옮기지 않는다. 멘토링 층의 문장은 잠정적 해석을 적은 것이고 잠정적이라는 표시는 그 문서 안에서만 유효하고 밖으로 나가면 판정처럼 읽힌다. 외부로 전달하는 것은 구체적인 산출물과 기한과 다음 행동이고 해석은 대화에서 말로 전한다. 에이전트가 초안을 쓸 때 내부 기록을 참조하는 것과 내부 기록의 문장을 결과물에 그대로 넣는 것은 다른 일이므로 뒤의 것을 막는 규칙을 지침에 적어 둔다. 에이전트가 만든 인물 요약은 요약일 뿐이므로 원자료와 최근 상태를 다시 확인하지 않은 채 근거로 삼지 않고 채용과 평가와 징계의 판단에는 쓰지 않는다. 규칙을 명시적으로 적어 두어야 하는 이유는 요약이 잘 쓰여 있을수록 확인을 건너뛰게 되기 때문이다. 학생이 준비 중인 원고에도 같은 종류의 경계가 걸린다. 문헌 위키는 출판된 지식을 담는 층이므로 준비 중이거나 심사 중인 연구실 원고는 제목조차 들어가지 않고 코드명으로 붙인 태그나 위키링크 없는 서술형 언급도 함께 막힌다. 사람에 관한 registry가 그 위키 안에 있는 이상 경계는 registry에도 그대로 적용되고 진행 중인 원고에 관한 메모는 별도 문서에 적는다. Registry 문서에는 날짜가 붙은 사실만 적으며 지금 무엇을 읽고 있는지, 어떤 작업과 방법을 익혔는지, 배정할 때 주의할 점과 확인이 필요한 항목이 무엇인지 정해진 형식으로 기록한다. 오래된 관찰이 현재 상태로 읽히지 않도록 기록마다 신선도 표시를 붙이고 일정 기간이 지나면 자동으로 재검토 필요 상태로 바꾼다. 반년 전에 병목이라고 적어 둔 내용이 지금도 같은 상태라는 보장은 없으므로 표시가 없으면 그 문장은 계속 현재형으로 읽힌다.

권한도 한 덩어리로 두지 않고 여섯 층으로 갈랐다. 연락처와 식별 정보, 지금 무엇을 맡고 있는지, 새 과제에 누구를 넣을지에 대한 판단, 새로 합류하는 사람에 대한 계획, 개인별 근거 기록, 작업 습관과 멘토링 개입이 각각 다른 층이다. 층을 나눈 실익은 인용 경로가 갈린다는 데 있다. 메일 초안을 쓸 때 필요한 것은 첫 층이고 작업 습관을 적어 둔 층은 초안에 인용하지 않는다. 그 층의 문장을 학생에게 가는 글에 그대로 옮기지 않고 구체적이고 중립적인 행동 요청으로 바꿔 쓴다는 규칙을 지침에 따로 박아 두었다. 잠정적인 해석으로 적은 문장이 밖으로 나가면 판정처럼 읽히고 문서로 남는 판정은 취소하기 어렵다. 이 경계에서 가장 자주 흔들리는 것은 요약이다. 잘 쓰인 요약일수록 원자료를 다시 열지 않게 만들고 그래서 인물 요약을 채용과 평가와 징계의 판단에 쓰지 않는다는 문장을 규칙에 남겨 두었다.

여러 작업 공간의 연결

업무를 나눈 작업 공간 여덟 곳이 모두 같은 골격을 쓴다. 얇은 지침 파일이 강제 규칙과 라우팅 표만 담고 지도 문서가 무엇이 어디에 있고 현재 규칙이 무엇인지를 표로 보여 주며 변경 이력 문서가 구조와 규칙이 언제 왜 바뀌었는지를 담고 그 아래의 owner 문서들이 실제 절차와 실패 조건과 검증 방법을 소유한다. 골격은 같아도 규칙의 두께는 크게 다르다. 판단이 자주 필요한 업무일수록 규칙 문서가 두껍고 절차가 굳어 있는 업무는 얇게 끝난다. 규칙 문서를 아예 두지 않은 폴더도 있는데 파일을 넣고 꺼내기만 하면 되는 보관소가 그렇다. 모든 폴더에 위키를 만들면 유지할 것만 늘고 읽히지 않는 규칙이 쌓인다. 큰 분석 과제나 여러 해에 걸친 업무는 상위 공간 안에 자기 지침과 자기 위키를 갖는 중첩 공간이 되고 상위 규칙은 그 안에서 하위 규칙이 우선한다고 명시한다. 우선순위를 명시하지 않으면 두 층의 규칙이 충돌할 때마다 사람이 매번 판정해야 한다.

각 공간의 위키는 문헌 위키의 내용을 복제하지 않고 절대경로로 가리킨다. 라우팅 표에는 이 위키에 중복해 적지 않는다는 열이 따로 있고 참조는 양방향이라 문헌 위키 쪽의 문서도 작업 공간의 경로를 가리킨다. 파일 복사는 하지 않고 이 계층에 symlink도 만들지 않는데 모든 문서를 실제 파일로 두어야 어느 것이 원본인지 헷갈리지 않기 때문이다. 역할도 나뉘어 있다. 사람과 연락처, 문헌 근거, 저널 정보, 문체 참고는 문헌 위키가 소유하고 폴더와 파일명과 버전과 렌더링과 검증 절차는 각 작업 공간이 소유한다. 공간 사이의 링크는 실제 의존이나 공유 정보가 있을 때만 만들고 서로를 가리키는 완전 그래프를 만들지 않는다. 두 곳의 기록이 서로 다르게 말할 때는 어느 쪽이 더 최근인지로 정하지 않고 소유 관계를 따른다. 역할에 관한 충돌은 사람에 관한 기록을, 분석의 현재 상태는 그 과제를 소유한 공간을, 원고의 최신 판본은 원고를 소유한 공간을 따른다. Owner가 아닌 곳에 적힌 값은 낡은 사본으로 보고, 낡은 사본을 발견하면 값을 고치는 대신 owner를 가리키는 링크로 바꾼다.

면담에서 얻은 내용을 공통 규칙으로 기록하려면 조건이 필요하다. 같은 혼선이 두 번 이상 반복되었거나, 한 번이라도

비용이 크거나 되돌리기 어려웠거나, 사용자가 앞으로의 처리 방법을 직접 정했거나, 다음 에이전트가 같은 결정을 재현해야 할 때다. 한 사례에만 해당하는 파일명과 상태, 검증하지 않은 임시 회피책, 특정 도구에 관한 개인적 취향, 검증할 수 없는 감상은 공통 규칙에 넣지 않는다. 규칙 문장은 상황, 행동, 검증의 세 요소로 쓰고 특정 사람, 날짜, 세션은 넣지 않는다. 처음에는 규칙을 문서에 적는다. 반복해서 확인해야 할 항목은 위반을 찾는 검사 코드로 만들고, 실행할 때마다 적용해야 하는 항목은 예약 작업이나 실행 스크립트에 포함한다. 문서 규칙을 자동 검사로 바꾸려면 서로 다른 대상 두 건 이상과 로그 항목 세 건 이상에서 같은 위반이 재발하지 않았다는 기록이 필요하다. 자동 검사가 반복해서 잘못 작동하거나 실제 관례와 충돌하면 검사를 해제하고 재검토 목록에 올린다. 실패와 성공은 함께 기록한다. 반복해 지적된 실패 패턴을 모은 문서와 반복해 효과를 본 전략을 모은 문서를 함께 관리하며, 전략에는 적용 시점, 방법, 근거, 현재 사용 여부, 사람이 정했는지 에이전트가 제안했는지를 적는다. 같은 혼선이 반복되는지를 확인하는 이유는 한 사람에게 통한 개입이 다른 사람에게도 맞다고 볼 수 없기 때문이다. 확인 없이 공통 규칙으로 적용하면 나머지 사람에게 맞지 않는 요구가 된다.

24장. Meetings 폴더로 PI의 외부 활동 관리하기

외부 강연 요청 메일 한 통에는 날짜, 장소, 청중, 발표 제목, 초록, bio, CV, 사진, 강연료, 교통비와 회신 기한이 섞여 있다. 수락하려면 캘린더에서 당일 일정과 이동 시간을 확인하고, 답장 초안을 만들고, 발표 준비 시간을 확보해야 한다. 주최 측이 보낸 양식이 있으면 원본을 보존한 채 작성본을 만들고 서명과 개인정보도 필요한 항목만 넣는다. 발표 자료에는 기존 슬라이드와 LLM 위키의 논문을 가져오고 행사 시간과 청중에 맞춰 다시 구성한다. 행사가 끝나면 영수증, 승차권, 지급 서류와 후속 메일이 남는다. 자문과 평가도 순서는 비슷하지만 제출물과 보안 조건이 다르다. 공동연구 미팅은 사전 자료, 회의 전사, 결정, 담당자와 다음 일정이 이어진다. 저희 연구실은 이 외부 활동을 Meetings 폴더에서 행사 단위로 관리한다.

제가 Meetings 폴더에 넣는 활동은 공동연구 미팅보다 넓다. 대학과 병원의 초청 세미나, 학회의 강연과 좌장, 정부와 연구기관의 자문위원회, 과제와 용역 평가, 학위 심사, 외부 워크숍, 국내외 출장도 같은 폴더에서 다룬다. 예정된 활동은 폴더 최상위에 두고, 끝난 활동은 연도 폴더로 옮긴다. 각 행사 폴더에는 초청 메일과 첨부 파일, 제출한 서류, 현재 발표본, 영수증과 정산 자료가 남는다. wiki/에는 여러 행사에서 되풀이되는 절차와 실수를 줄이는 규칙을 기록한다. 에이전트는 이 규칙을 읽고 메일, 캘린더, LLM 위키, 행사 폴더에서 필요한 정보를 찾아 준비물을 만든다. 사람은 수락 여부, 외부 발송, 비용 지출, 공개 범위와 과학적 판단을 승인한다. 아래의 공동연구 사례도 PI의 외부 활동 가운데 반복이 많은 미팅 업무로 이어진다.

Meetings 폴더가 다루는 활동

활동마다 준비물은 다르지만 요청을 받고 마무리하는 흐름은 반복된다. 그래서 행사 폴더는 활동의 원본과 결과물을 모으고, wiki/는 유형별 절차를 알려 준다.

활동	행사 폴더에서 관리하는 것	자주 반복되는 작업
외부 세미나와 특강	초청 메일, 제목, 초록, bio, CV, 사진, 발표 자료	일정 확인, 회신, 자료 제출, 슬라이드 준비, 강연료 서류
학회와 워크숍	초록, 프로그램, 등록, 발표 자료, 배포 자료	제출 마감 확인, 발표 준비, 등록과 출장 정리
자문과 위원회	위촉 자료, 회의 자료, 자문의견서, 참석 기록	자료 검토, 의견서 작성, 서명, 자문료 서류
평가와 심사	평가 자료, 평가표, 보안서약서, 제출본	이해충돌과 보안 확인, 점수와 의견 작성, 제출 형식 검증
공동연구 미팅	사전 자료, 핵심 논문, 전사, 결정, 후속 행동	참석자와 안건 확인, 자료 공유, 회의 기록, 다음 작업 배정
국내외 출장	초청장, 프로그램, 교통, 숙박, 영수증, 보고서	이동 일정 확보, 예약 자료 보존, 사후 정산

강연 요청을 예로 들면 에이전트는 메일 스레드와 첨부 파일에서 행사명, 주최, 날짜, 장소, 청중, 요청받은 자료와 마감을 뽑는다. 캘린더에서는 실제 약속과 이동 시간, 발표 준비에 쓸 시간을 확인한다. 수락할 수 있는 일정이면 답장 초안을 Gmail에 저장하고 수신자, 제목, 본문, 첨부 예정 파일을 사람에게 보여 준다. 승인을 받은 뒤 메일을 보내고 확정된 일정을

캘린더에 반영한다. 기존 행사 폴더를 날짜와 주최, 행사명으로 검색한 뒤 중복이 없을 때 새 폴더를 만든다. 초청 메일과 원본 양식은 보존하고, 현재 발표본과 제출본은 폴더를 열었을 때 바로 보이게 둔다. 발표가 끝나면 강연료와 출장비에 필요한 증빙을 /에 모으고 후속 회신을 준비한다. 같은 유형의 행사에서 다시 쓸 수 있는 교훈은 wiki/seminars/, wiki/travel/, wiki/documents/의 해당 문서에 반영한다.

외부 활동의 진행 상태

PI의 외부 활동은 요청을 받은 시점부터 끝난 뒤의 정산까지 이어진다. 메일에 초청 내용이 있어도 일정과 역할이 확정되지 않았으면 상태다. 수락 메일을 보낸 뒤 공식 일정이 잡히면 으로 바뀐다. 발표 제목과 초록, CV처럼 먼저 보내야 할 자료가 남아 있으면 로 적는다. 슬라이드와 자문 의견서, 평가표를 만들고 있으면 산출물의 이름을 상태에 넣는다. 행사 당일이나 지나도 영수증과 지급 서류가 남아 있으면 다. 공동연구 미팅은 , , 처럼 연구의 다음 조건을 상태 이름에 쓴다. 한마디로 적으면 다음에 무엇을 해야 하는지 알 수 없으므로, 기다리는 자료나 사람의 행동을 함께 적는다.

↓
, , ,
↓
↓
↓
, , ,
↓
↓
↓
, ,

상태가 바뀌는 조건은 확인 가능한 문장으로 적는다. 는 초록과 CV를 보냈을 때 끝나고, 는 발표본을 렌더링해 확인했을 때 끝난다. 는 요구된 증빙과 지급 서류를 제출하고 수신 여부를 확인한 뒤 닫는다. 공동연구에서는 기관별 데이터 승인과 분석 담당이 다를 수 있으므로 기관마다 상태를 따로 기록한다. 메일과 캘린더의 내용을 행사 메모에 통째로 복사하지 않고 원본으로 가는 링크와 현재 판단을 남긴다. 원본 메일과 일정이 바뀌면 에이전트는 작업을 시작할 때 다시 읽는다. 이 방식이면 다음 세션에서도 에이전트가 어떤 자료를 기다리는지, 무엇을 만들고 있는지, 어떤 승인이 필요한지 말할 수 있다. 행사명과 상태, 다음 조건을 한 줄로 읽었을 때 다음 행동이 나오면 기록이 작동하는 것이다.

정보마다 정본을 맡는 작업 공간

행사 폴더에는 외부 활동을 준비하고 마친 결과가 모이지만 메일과 일정의 원본까지 복사하지 않는다. 초청 조건과 합의는 Gmail 스프레드에서 확인하고, 확정된 시간은 Google Calendar에서 확인한다. 행사 폴더에는 주최 측이 보낸 첨부 파일과 작성한 제출본, 현재 발표본, 회의 기록, 영수증을 둔다. LLM 위키는 발표와 자문에 쓸 논문 지식을 제공한다. 분석 결과와 코드는 연구 프로젝트 폴더에서 가져오고, 공동 원고의 문장과 그림은 원고 폴더에서 관리한다. 같은 정보를 여러 폴더에 복사하면 수정된 날짜와 판본이 달라진다. 에이전트는 작업을 시작할 때 각 원본을 다시 읽고 행사 폴더에는 링크와 이번 활동에 필요한 결과물만 남긴다. 무엇을 어디에서 읽고 어디에 저장할지 정하면 준비 과정에서 생기는 중복 파일이 크게 줄어든다.

정보와 산출물	정본을 확인하는 곳
초청, 요청, 합의와 회신	Gmail 스프레드
확정된 행사 시간, 장소, 온라인 접속 정보	Google Calendar와 공식 초대
초청장, 원본 양식, 제출본, 발표본, 영수증	Meetings의 행사 폴더
반복되는 세미나, 자문, 평가, 출장 절차	Meetings의 wiki/
상대의 이름, 소속, 검증된 연락처	LLM 위키의 연락처 registry
발표와 자문에 다시 쓸 논문 지식	LLM 위키

정보와 산출물	정보를 확인하는 곳
제안서, 예산, 과제 협약	Grants 작업 공간
분석 코드와 결과	Projects 작업 공간
공동 원고, 저자 순서, 소속 표기	Manuscript 작업 공간

외부 세미나에서 발표 제목을 정할 때는 초청 메일의 청중과 요청 주제를 읽고, 과학 내용은 LLM 위키와 Projects의 현재 결과에서 가져온다. 제목과 초록, bio, CV를 제출하면 그 파일은 행사 폴더에 남긴다. 캘린더에는 행사 시간과 이동 구간을 기록하고, 슬라이드 준비처럼 옮길 수 있는 작업은 별도의 집중 시간으로 확보한다. 공동연구 미팅에서 분석 방향이 바뀌면 결정은 회의 노트에 남고 실제 코드는 Projects에서 갱신한다. 원고 그림을 바꾸기로 했다면 Manuscript에서 수정하고 미팅 노트에는 결정과 연결 파일만 적는다. 자문료나 강연료 서류에는 행사 폴더의 원본 양식과 본인 정보의 정보를 사용한다. 폴더를 새로 만들기 전에는 같은 날짜와 주최, 행사명이 이미 있는지 최상위와 연도 폴더에서 확인한다. Zoom 링크만 있는 짧은 정기 미팅처럼 보존할 외부 자료가 없으면 기존 프로젝트 기록과 캘린더로 관리하고 빈 행사 폴더를 늘리지 않는다.

외부 요청 intake의 구조화

새 요청이 들어오면 메일을 한 장짜리 intake 기록으로 바꾼다. 세미나와 자문, 평가는 요청하는 역할과 산출물이 다르므로 활동 유형을 먼저 적는다. 행사명과 주최 기관, 담당자, 날짜, 장소, 온라인 여부를 확인한다. 주최 측이 원하는 역할이 발표자인지, 좌장인지, 자문위원인지, 평가위원인지도 구분한다. 제목과 초록, CV, 사진, 의견서, 평가표, 보안서약서처럼 제출할 자료와 마감을 적는다. 강연료, 자문료, 평가수당, 출장비의 지급 주체와 필요한 서류를 확인한다. 청중과 공개 범위, 보안 조건은 발표 자료와 외부 공유 범위를 정할 때 사용한다. 마지막에는 현재 답변 상태와 다음 행동, 근거가 된 메일 스레드와 첨부 파일을 남긴다.

Intake 항목	기록할 내용
활동 유형	세미나, 강의, 학회, 자문, 평가, 공동연구 미팅, 출장
행사과 주최	공식 행사명, 기관, 담당자
일정과 방식	날짜, 시간, 장소, 온라인 또는 대면
맡은 역할	발표자, 좌장, 자문위원, 평가위원, 공동연구자
제출물과 마감	제목, 초록, CV, 사진, 발표본, 의견서, 평가표, 양식
비용과 정산	강연료, 자문료, 수당, 교통, 숙박, 지급 주체
공개와 보안	청중, 외부 공유 범위, 보안서약, 자료 반출 조건
현재 상태	검토 중, 수락, 일정 확정, 자료 준비, 정산 대기
다음 행동과 근거	담당자, 기한, 메일 스레드, 첨부 파일

Intake 기록에는 확인된 사실과 아직 확인할 항목을 구분해 쓴다. 주최 측이 장소나 강연료를 알려 주지 않았으면 빈칸을 추측으로 채우지 않고 확인 요청을 만든다. 메일 본문은 Gmail 스레드에 그대로 남아 있으므로 행사 기록에는 현재 판단과 원문 링크를 둔다. 담당자의 주소와 소속은 검증된 연락처 registry를 조회한다. 받은 첨부 파일은 행사 폴더에 원본 이름으로 보존하고 작성할 양식은 별도 제출본으로 만든다. 공동연구 미팅이라면 상대가 보낸 데이터 설명과 논문 목록에 현재 위치와 검토 상태를 붙인다. 자문과 평가 자료에는 보안 조건을 적어 공개 발표 자료와 섞이는 일을 막는다. 다음 행동에는 누가 무엇을 언제까지 준비하는지 쓰며, 외부 기관의 답변이나 승인이 먼저 필요하면 그 조건도 함께 남긴다.

이메일 스레드에서 복원하는 요청과 결정

메일함에는 외부 활동에서 오간 요청과 합의의 원본이 남는다. 누가 무엇을 언제 요청했고 상대가 무엇에 동의했는지는 Gmail 스레드에서 확인한다. 행사 기록에는 메일 스레드로 가는 링크와 아직 해결되지 않은 내용 한 줄을 둔다. 스레드를 찾을 때 사람 이름의 기억에 의존하지 않고 검증된 연락처 기록에서 주소를 확인한 뒤 그 주소로 찾는다. 찾은 다음에도 최신 한 통만 읽지 않고 스레드 전체를 읽어 지금 살아 있는 요청, 이미 답한 항목, 첨부된 자료, 마감, 참조에 들어 있는 사람을 확인한다. 외부 세미나는 처음 메일에서 제목과 초록만 요청했다가 나중에 CV와 사진, 지급 서류가 추가되기도

한다. 공동연구에서는 데이터 반출 승인이나 공유 범위가 중간에 바뀔 수 있다. 메일 내용을 할 일로 바꿀 때는 요청한 사람, 산출물, 마감과 선행 조건을 분리해 적는다.

메일 초안을 에이전트가 쓰는 것과 메일을 에이전트가 보내는 것은 다른 일이고 순서가 정해져 있다. 초안을 만들고 초안을 파일로 저장하고 사람에게 내용을 보여주고 기다리고 명시적인 승인을 받은 뒤에 발송한다. 답장해 달라거나 메일 보내 달라거나 리마인드 해 달라는 말은 그 순서의 앞부분을 지시한 것이고 초안을 만들어 보여주고 기다리라는 뜻이다. 초안을 한 번 고쳤으면 고친 것을 다시 보여주고 새 승인을 받는데 고친 대목이 수신자거나 첨부거나 약속의 수위일 수 있어서 앞선 승인이 고친 초안에 적용되지 않기 때문이다. 참조에 들어가면 안 되는 사람이 있는지, 첨부한 자료가 그 기관에 나가도 되는 것인지는 협업의 공개 경계를 아는 사람이 판단해야 하고 잘못 나간 메일은 되돌릴 수 없다. 메일 본문에는 내부 지침, 전략, 정산 논리, 판단 과정을 넣지 않고 받는 사람의 관점에서만 쓴다. 에이전트가 만든 초안에는 어느 문서에서 가져왔는지, 어떤 계획으로 이 표현을 골랐는지 같은 문장이 섞여 들어오는 일이 흔하다. 저장하고 발송하기 전에 독자에게 보이는 면을 훑어 지시문과 계획과 도구 상태와 조수 어투를 걷어내는 일이 검토의 절반이고 초안 품질이 안정되면 이 검토가 형식적으로 변하므로 오래된 협업일수록 더 중요해진다.

메일 발송은 되돌릴 수 없어서 사람이 마지막으로 승인한다. 잘못 만든 파일은 지우거나 이전 판으로 되돌릴 수 있지만 나간 메일은 받는 사람의 편지함에 남는다. 정정 메일을 보내면 두 통이 되고 상대가 어느 쪽을 읽었는지 알 수 없다. 세미나 수락 메일의 문장 하나는 일정과 강연 준비의 약속이 되고, 공동연구 화신은 데이터와 인력의 약속으로 읽힌다. 에이전트는 무엇을 답해야 하는지 찾고 관련 자료를 모으며 초안과 첨부를 준비한다. 사람은 수신자와 참조, 일정, 비용, 공개 범위와 약속의 수위를 확인한다. 초안에 내부 판단 과정이나 지시문이 들어가지 않았는지도 받는 사람의 관점에서 다시 읽는다. 승인은 초안 한 건에만 적용하므로 내용을 고친 뒤에는 수정본을 다시 확인한다.

약속과 준비라는 일정의 두 층

실제로 참석해야 하는 미팅과 그 미팅을 준비하는 시간은 서로 다른 일정이다. 두 가지를 같은 층에 섞으면 일정표는 가득 차 있는데 준비할 시간이 없고 준비 시간을 아예 적지 않으면 준비는 남는 시간에 하는 일이 되어 전날 밤에 몰린다. 두 층으로 나누는 기준은 이동 가능성이다. 상대와 약속한 시간은 혼자 옮길 수 없고 사전 읽기와 자료 작성과 분석 검토는 옮길 수 있다. 옮길 수 없는 것과 옮길 수 있는 것을 같은 목록에 두면 옮길 수 없는 것이 항상 이긴다. 약속 층에서 정보는 공식 초대다. 여러 사람이 참여하는 일정을 잡을 때는 각자의 가능한 시간을 모으는 단계와 하나의 공식 초대로 확정하는 단계를 구분한다. 확정 전에 개인적으로 잡아 둔 임시 표시가 남아 있으면 같은 시간에 두 항목이 생겨 다음에 어느 것이 진짜인지 알 수 없으므로 공식 초대가 만들어지면 임시 표시는 지운다.

일정을 다루는 에이전트에게 허용하는 범위는 좁게 잡는다. 참석자를 자동으로 초대하거나 화상회의 링크를 자동으로 붙이는 동작은 하지 않는다. 초대는 상대에게 알림을 보내고 상대의 일정을 점유하므로 외부로 나가는 행동이고 외부로 나가는 행동은 사람이 결정한다. 같은 시간대에 겹치는 항목, 확정되지 않은 채 남은 항목, 이동 시간이 확보되지 않은 연속 일정을 찾는 일은 목록이 길수록 사람보다 정확하다. 일정을 바꾸는 작업에는 원장을 하나 붙여, 바꾸기 전에 무엇을 어떻게 바꿀 것인지 먼저 적고 끝난 뒤에 결과를 다시 적는다. 원장에는 이미 적힌 줄을 고치지 않고 시작과 종료를 짝으로 덧붙이기만 하므로 시작만 있고 종료가 없는 줄이 남으면 그 작업이 중간에 끊겼다는 사실이 그대로 드러난다. 상대가 그사이에 시간을 옮겼거나 같은 작업이 두 번 실행되어 항목이 중복되는 일도 이 방식에서 먼저 보인다. 다만 원장은 지나간 일의 기록이므로 지금 일정이 어떻게 되어 있는지는 언제나 일정 자체에서 읽는다.

미팅 전의 같은 출발점

좋은 미팅은 각자가 같은 지점에서 출발하는 미팅이다. 자료를 많이 보내면 읽는 사람마다 다른 부분을 읽고 다른 인상을 가지고 들어온다. 사전 자료가 해야 하는 일은 오늘 무엇을 정해야 하는지 참석자에게 알려 주는 것이다. 한 장으로 충분하고 한 장을 넘어가면 읽지 않는 사람이 생긴다. 다섯 항목이면 그 한 장이 채워진다. 이번 협업이 답하려는 질문 한 문장, 이미 확보한 데이터와 핵심 논문의 원본 위치, 지금 가능한 설명과 그와 경쟁하는 설명, 아직 확인되지 않은 사실과 자료의 한계, 그리고 오늘 필요한 결정이다. 마지막 항목은 미팅이 끝났을 때 참이나 거짓으로 판정되는 문장으로 쓴다. 소개나 상호 이해로 적으면 미팅이 끝난 뒤 성공했는지 알 수 없다.

사전 자료에 인용한 핵심 논문은 문헌 위키에 실제로 들어놓는다. 제목만 적어 두면 그 논문의 결과가 왜 이 협업에 중요한지 설명해야 할 때 다시 원문을 찾게 된다. 근거로 쓰겠다고 적은 논문은 원문까지 확인한다. 초록만 읽고 적은 근거는 질문을 받으면 곧바로 흔들리고, 다기관 협업에서 한 번 잃은 신뢰는 다음 분기까지 영향을 준다. 발표 자료는 미팅 하루 전쯤 공유하고 당일 아침에 보내면 참석자는 회의 중에 처음 본다. 여러 기관이 참여하고 시간대가 다르다면 하루의 여유가 실제로는 반나절이 되므로 무엇을 읽고 오면 되는지를 자료와 함께 한 줄로 적는다. 학생과 연구원에게는

발표 분량, 각자가 책임질 질문, 미팅 뒤에 남길 산출물을 함께 알린다. 발표 시간과 산출물을 함께 정하면 미팅이 다음 작업으로 이어진다.

미팅 중 관찰과 해석과 결정의 분리

미팅 기록에서 가장 자주 무너지는 것은 세 가지가 한 문장에 섞이는 일이다. 어떤 조건에서 어떤 값이 나왔다는 관찰, 그 값이 무엇을 뜻하는지에 대한 해석, 그래서 무엇을 하기로 했다는 결정은 각각 신뢰 수준이 다르다. 관찰은 데이터로 되짚을 수 있고 해석은 나중에 바뀔 수 있으며 결정은 바뀌더라도 언제 왜 바뀌었는지가 남아야 한다. 셋을 한 문장으로 적으면 나중에 어느 부분이 사실이고 어느 부분이 그날의 인상이었는지 구분할 수 없다. 결정되지 않은 것을 결정처럼 적지 않는 규칙도 같은 이유에서 나오고 결론이 나지 않았다면 기록에는 선택지와 아직 정해지지 않았다는 사실과 정하려면 무엇이 더 필요한지를 적는다. 결론이 난 것처럼 적으면 다음 미팅에서 그것을 전제로 논의가 진행되고 뒤늦게 아무도 동의한 적이 없다는 사실이 드러난다. 다기관 협업에서 이 오류의 비용은 특히 큰데, 기관마다 그 전제 위에서 내부 절차를 이미 시작했을 수 있기 때문이다. 그때 새 분석을 요청할 때는 그 분석이 어떤 질문에 답하려는 것인지와 어떤 결과가 나오면 무엇으로 판정할지를 요청에 붙인다. 목적과 판정 기준 없이 넘어간 요청은 결과가 나온 뒤에 해석을 두고 논의를 다시 열고 그 논의의 부담은 분석을 실제로 돌린 사람에게 간다. 확정된 후속 행동은 누가 맡는지, 무엇을 만드는지, 언제까지인지, 시작하려면 먼저 무엇이 있어야 하는지를 갖춘 형태로 적는다. 미팅에서 이해했다고 느낀 역할 부담이 두 주 뒤에 다르게 기억되는 일은 기관이 여럿이면 더 자주 일어나고 네 가지를 적는 데 드는 시간은 몇 분인데 적지 않아서 생기는 지연은 분기 단위다.

1. 지난 미팅의 후속 행동을 하나 고른다
2. 담당자, 산출물, 마감, 선행 조건 네 가지를 갖춘 형태로 다시 적는다

기록을 세 층으로 나누면 이 구분이 파일 구조로 강제된다. 첫 층은 화자 라벨을 붙인 원문 전사로, 손대지 않고 그대로 남기며 여기에는 농담과 중간에 철회된 발언과 잘못 들린 숫자가 들어 있다. 둘째 층은 요약과 할 일과 분석을 담은 정리 노트이고 사람이 전사를 읽고 확인한 판단만 들어간다. 셋째 층은 날짜와 회의와 기록을 잇는 색인이며 어느 날 어떤 회의가 있었고 그 회의의 전사와 정리 노트가 어디에 있는지만 담는다. 세 층을 나누는 이유는 신뢰 수준이 다른 것을 같은 파일에 두면 낮은 쪽이 높은 쪽의 권위를 빌려 가기 때문이다. 전사를 고쳐 읽기 좋게 다듬으면 원자료가 사라지고 참석자가 확인하지 않은 문장이 정리 노트에 섞이면 그 문장이 합의로 취급되어 다음 단계로 넘어간다. 색인이 내용을 담기 시작하면 색인과 정리 노트가 서로 어긋나고 어긋난 뒤에는 둘 다 믿을 수 없게 된다. 자동 전사는 첫 층에만 들어가고 확정된 결정으로 취급하지 않는다.

PI의 외부 미팅에서는 회의 자료 수집부터 공유 전 검토, 후속 행동, 다음 미팅 연결까지 한 흐름으로 관리한다. 원시 전사와 개인 정보는 공유 문서로 옮기지 않는다.

미팅 후 결과를 원래 작업 공간에 반영하기

미팅이 끝난 직후에 하는 일은 참석자에게 보낼 짧은 요약을 만드는 것이다. 요약에는 결정된 것과 후속 행동만 넣고 논의 과정과 오간 의견은 넣지 않는다. 짧을수록 읽히고 읽혀야 어긋난 이해가 그때 교정된다. 보내는 시점은 미팅 당일의 가장 좋은데 하루가 지나면 참석자의 기억이 각자의 방향으로 정리되어 있기 때문이다. 이 요약도 외부로 나가는 문서이므로 초안을 만들고 보여주고 승인을 받은 뒤에 발송하는 순서를 그대로 따른다. 후속 미팅이 필요하면 새 일정을 만들기 전에 기존 일정과 메일 스레드를 먼저 확인한다. 이미 잡힌 정기 미팅이 있는데 별도 일정을 잡으면 참석자는 비슷한 미팅 두 개에 들어가게 된다. 여러 기관이 참여하면 한 번 모이는 비용이 크므로 정말 모여야 정해지는지를 먼저 따지고 다뤄야 할 것이 하나뿐이면 메일로 끝난다.

미팅에서 나온 결과는 실제 작업을 맡는 공간에 반영한다. 분석 방향의 변경은 Projects에, 원고의 구성과 그림에 관한 결정은 Manuscript에, 제안서에 걸리는 약속은 Grants에 반영하고 각 산출물은 해당 폴더의 파일명 규칙과 검증 절차를 따른다. 이 협업 밖에서도 다시 쓸 문헌 판단은 LLM 위키에 남긴다. 진행 메모나 변경 기록이나 피드백 문서를 행사 폴더 루트에 새로 만들면 폴더마다 형식이 다른 기록이 흩어지고 폴더를 정리할 때 함께 사라진다. 반복해서 확인할 운영 규칙은 Meetings의 위키 로그와 관련 문서에 반영한다. 로그에는 작업을 시작한 이유, 원본 상태, 실행한 내용, 확인 결과, 발견한 문제, 결정, 남은 일과 산출물을 적는다. 확인 결과에는 확인됨, 불일치, 미확인, 보류, 재실행 필요 가운데 하나를 붙여 확인하지 못한 일을 다음 세션에서 다시 찾을 수 있게 한다. 다음 미팅에서 답해야 할 질문도 남겨 두며 질문이 비어 있다면 다음 상태로 넘어갈 조건부터 다시 확인한다.

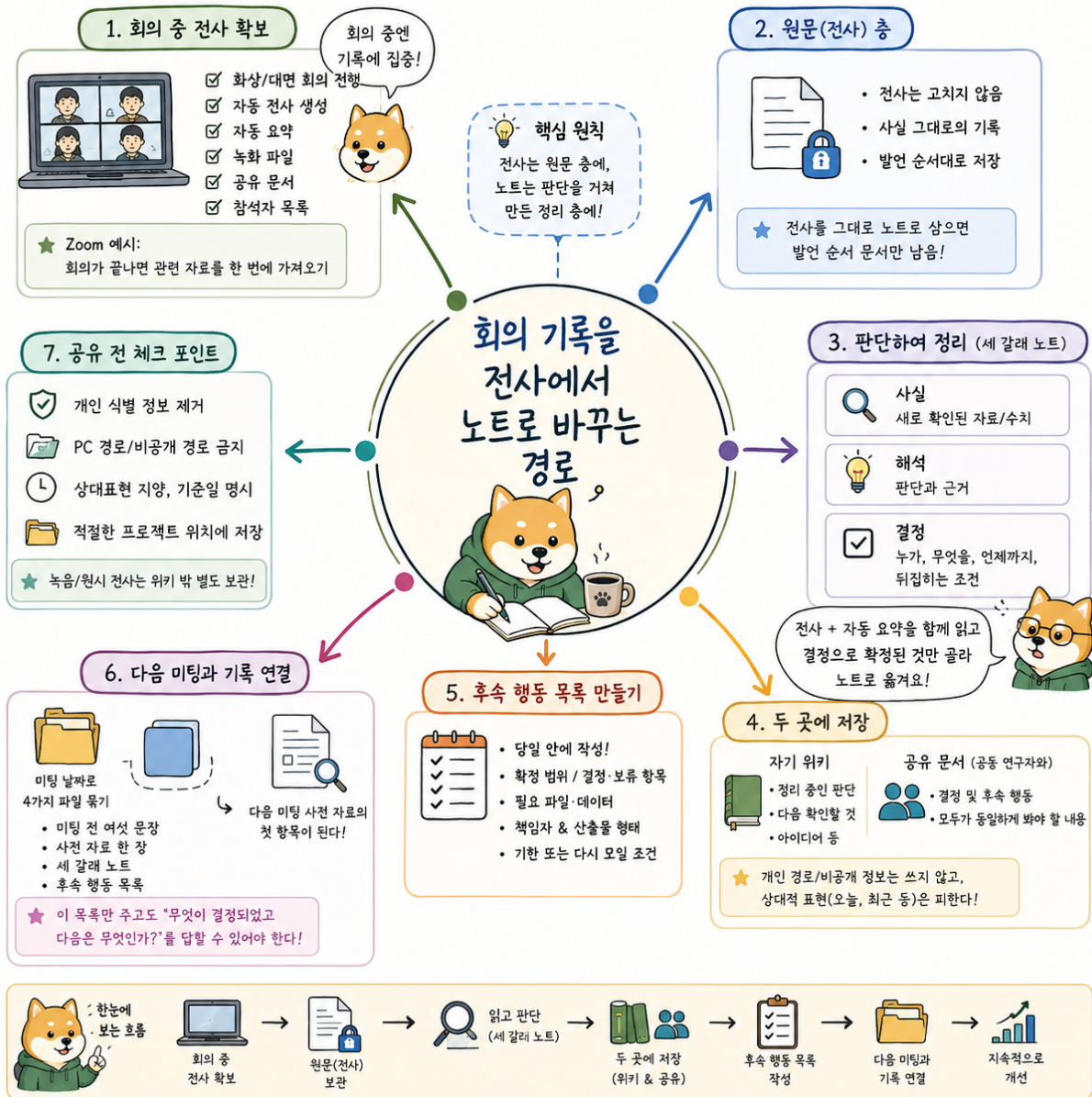


Figure 14: PI의 외부 행사에서 회의 전사를 노트와 후속 행동으로 바꾸는 흐름

새 분석 공동연구의 역할 배정

합성 사례의 첫 국면은 협업이 막 시작되어 첫 미팅을 준비하는 시점이다. 외부 기관이 데이터를 가지고 있고 분석은 이쪽에서 맡기로 한 상황에서, 연구책임자는 과학적 배경과 질문을 설명하는 역할을 맡는다. 분석을 실제로 수행하는 역할은 원고 준비 단계에 있는 대학원생에게 가는데 그 학생은 이미 자기 주저자 원고를 쓰고 있으므로 이 협업에 쓸 수 있는 시간이 제한된다. 학부 연구생은 배우는 위치로 데이터 구조를 파악하고 박사후연구원은 이 협업이 확장될 방향을 살핀다. 참여자를 나열하는 것으로는 부족해서 과학적 방향을 책임지는 사람, 데이터를 소유하고 반출 권한을 가진 사람, 분석을 수행하는 사람, 배우는 위치로 참여하는 사람을 구분한다. 다기관 연구에서는 데이터 소유자와 분석 수행자가 다른 기관에 있는 경우가 많다. 요청할 사람과 승인할 사람이 기록되지 않으면 모든 요청이 연구책임자를 거치면서 병목이 생긴다. 역할은 프로젝트 책임 기록에 쓰고 학생 개인 기록에는 현재 맡은 작업으로 가는 링크만 둔다.

첫 미팅 전에 공유하는 것은 핵심 논문 두어 편과 데이터 설명이다. 논문은 LLM 위키에 올려놓은 상태로 공유하고 데이터 설명은 상대 기관이 보낸 원본을 그대로 전달한다. 미팅에서 정할 것은 세 가지로 미리 적어 둔다. 각자가 어느 부분을 책임지는지, 데이터에 어떤 경로로 접근하는지, 첫 산출물이 무엇이고 언제까지인지다. 소개와 배경 설명에 미팅의 대부분을 쓰면 이 세 가지는 다음으로 밀리고 다음 미팅은 분기 된다. 세 항목이 모두 정해져야 상태를 담당자별 행동으로 바꾼다. 데이터 접근 경로가 정해지지 않았다면 상태로 두고 해당 기관의 반출 승인을 기다린다. 상대의 신원과 관계 등급은 연락처 registry에 갱신하고 intake 기록에서는 해결된 항목을 지운다.

분석 담당자가 바뀔 때

같은 협업의 두 번째 국면은 분석을 맡던 사람이 바뀌는 시점이다. 대학원생이 원고 마무리에 들어가면서 분석 담당이 넘어가야 하는데 이때 흔히 하는 선택이 긴 인수인계 미팅을 한 번 잡는 것이다. 세 시간짜리 미팅에서 데이터 위치와 파이프라인과 결과와 남은 문제를 모두 설명하면, 받는 사람은 그날 이해한 기본으로 나가고 이를 뒤에 첫 질문을 하게 된다. 한 번에 전달된 맥락은 한 번에 잊히고 잊힌 부분은 대개 예외 처리와 실패한 시도에 관한 것이다. 이미 해 보고 버린 방법을 새 담당자가 다시 시도하는 일이 인수인계 직후에 가장 흔하다. 주제를 나눈 인수인계가 더 잘 작동한다. 첫 만남에서는 프로젝트의 질문, 데이터 위치, 파이프라인 단계, 현재 결과와 남은 문제를 넓게 훑는다. 그다음 만남부터 전처리, 집단 관리, 분석 틀을 하나씩 열어 보면 받는 사람이 사이사이에 직접 실행해 볼 수 있다.

인수인계 문서를 누가 쓰지도 정해 둔다. 넘기는 사람이 쓰면 그 사람이 이미 아는 것을 기준으로 쓰게 되어, 정작 새로 오는 사람이 막히는 지점이 빠진다. 넘기는 사람은 실행 가능한 파일과 제약 조건을 설명하고 문서는 받는 사람이 자기 말로 다시 쓴다. 다시 쓴 문서를 넘기는 사람이 읽고 틀린 곳을 고치면, 그 문서는 두 사람이 같은 것을 이해했다는 증거가 된다. 담당이 바뀌면 Projects의 담당 기록, 연락처 registry, Manuscript의 저자 목록, 상대 기관에 통지할 창구를 함께 갱신한다. 한 곳만 고치면 나머지는 옛 담당자를 가리킨다. 반출 승인이 개인 단위로 나 있다면 담당 변경과 함께 승인 절차를 다시 시작한다. 문서 인수인계와 승인 대기를 동시에 진행하면 분석이 멈추는 기간을 줄일 수 있다.

계산 결과를 실험으로 검증하는 협업

세 번째 국면은 분석에서 나온 결과를 실험으로 확인하기로 한 상황이다. 계산팀과 실험팀이 만날 때 자주 벌어지는 일은 서로가 상대의 제약을 모른 채 논의하는 것이다. 계산팀은 검증하고 싶은 후보를 여럿 제시하고 실험팀은 그중 무엇이 실제로 가능한지를 그때 판단해야 한다. 사전 문서 없이 만나면 미팅은 후보를 하나씩 훑으며 가능한지 묻게 되고 끝난 뒤에는 무엇을 하기로 했는지가 모호해진다. 사전 문서에는 계산에서 나온 결과, 가능한 실험 방식, 필요한 시료와 조건, 판정 기준을 적는다. 판정 기준을 미리 정하면 실험팀은 그 판정이 자기 조건에서 가능한지 바로 답할 수 있다. 실험이 끝난 뒤 기준을 정하면 결과 해석부터 다시 논의하게 된다. 네 항목을 채우는 과정에서 계산 결과가 실험으로 판정하기 어려운 형태임을 발견할 수도 있다.

미팅 준비에서는 참석자의 가능한 시간, 학생이 준비할 자료, 만날 방식, 미리 읽을 논문을 함께 확인한다. 각각 따로 챙기면 대개 미리 읽을 논문이 빠진다. 협업 상태 기록에 이번 미팅에 필요한 것을 적어 두면 준비가 어디까지 되었는지 한 번에 보인다. 여러 기관의 사람이 모이는 미팅은 다시 잡기 어려우므로 자료가 부족하면 필요한 조건을 채운 뒤 새 날짜를 정한다. 미룰 때는 어떤 자료가 오면 다시 잡을지도 함께 적는다. 미팅 후 기록에는 무엇을 만들어 무엇을 측정하고 어떤 대조군을 쓰는지, 결과를 보고 어떤 결정을 내릴지를 남긴다. 결정 기준이 있으면 중간 결과를 보고 계속할지 멈출지 판단할 수 있다. 중단한 결정도 기록해야 같은 후보가 다시 올라왔을 때 이유를 확인할 수 있다.

Meetings 폴더와 위키의 역할

업무를 소유하는 작업 공간들은 같은 골격을 쓴다. 입구의 지침 파일은 반드시 지켜야 할 것과 어디로 가야 하는지를 담은 얇은 라우터이고 목차 문서가 지도와 현재 규칙 표를, 변경 이력 문서가 구조와 규칙이 어떻게 바뀌어 왔는지를, 그 아래 영역별 문서가 실제 절차와 실패 조건과 검증을 담는다. 어느 작업 공간에 들어가든 읽는 순서는 같아서 지침 파일을 먼저 읽고 목차로 가고 변경 이력의 최근 항목을 훑은 다음 해당 영역의 문서로 내려간다. 목차 문서에 규칙을 채워 넣기 시작하면 규칙이 두 곳에 생기고 두 곳이 어긋나면 어느 쪽을 따라야 할지 알 수 없다. 목차가 길어지고 있다면 규칙을 목차에 쓰고 있다는 신호다. 변경 이력은 규칙이 왜 지금 모습이 되었는지를 복원하는 역사이고 현재 규칙을 덮어쓰지 않으므로 오래된 항목을 읽고 현재 규칙을 추측하는 대신 현재 규칙 문서를 읽는다. 상세 규칙은 그것을 소유한 작업 공간에 두고 전역 파일은 작업 공간 사이의 라우팅과 진짜 전역인 것만 소유한다. 되돌리는 수단이 동기화 서비스의 버전 기록뿐이라는 전제가 규칙 전체를 지배해서 덮어쓰기 전에 먼저 읽고 무엇을 바꿀지 보이고 승인을 받는다.

행사와 미팅을 소유한 작업 공간의 최상위 배치는 이 골격을 그대로 드러낸다.

```
Meetings/
  AGENTS.md
  wiki/
    index.md
    log.md
    conferences/
    seminars/
    advisory/
    evaluations/
    travel/
    documents/
    planning/
    slides/
    institutions/
  { }/
  { }_{ }_{ }_{ }/
```

폴더 이름은 예시이므로 구분 기호와 표기는 각자의 정렬 습관에 맞추면 된다. 작업 단위는 행사 폴더 하나이고 이름은 날짜와 주최와 행사명 세 조각으로 이루어진다. 날짜를 앞에 두면 목록이 시간순으로 정렬되고 주최를 가운데 두면 같은 기관에서 반복되는 행사가 이름만으로 묶인다. 해가 바뀌면 지난 행사를 연도 폴더로 옮겨 최상위에는 지금 준비하는 활동만 남긴다. 위키 아래에서는 학회, 세미나, 자문, 평가, 출장, 서류, 일정, 발표 자료와 기관 정보를 유형별로 나눈다. 유형마다 준비물과 확인 순서가 달라서 관련 문서만 읽고 작업을 시작할 수 있다. 최상위 규칙 파일은 강제 규칙과 문서 위치를 안내하고 세부 절차는 해당 위키 문서가 맡는다. 미팅 기록은 원문 전사, 정리 노트, 색인으로 나누어 확인되지 않은 발언이 결정처럼 읽히는 일을 막는다.

행사 일정, 사전 준비, 출장 예약, 미팅 진행, 귀국 후 정산을 하나의 작업 단위로 관리하는 예다. 일정과 정보, 자료, 비용 기록은 각각의 정본에서 읽는다.

따라해보기

아래 블록은 AI 도구의 채팅창에 그대로 붙여 넣는 프롬프트다. 행사와 미팅을 소유할 작업 공간을 처음 세울 때 쓰는 프롬프트다. 꺾쇠 안은 자기 경로와 표기로 바꾼다.

```
< > .
- AGENTS.md wiki/
- { }_{ }_{ }_{ }
- wiki/ index.md log.md conferences/, seminars/, advisory/,
  evaluations/, travel/, documents/, planning/, slides/, institutions/
- AGENTS.md
1.
2. , ,
3.
```




Figure 15: 연구책임자가 미팅과 외부 출장을 준비하고 기록하는 흐름

작업 공간이 나누는 영역은 일의 종류를 따른다. 행사 폴더를 만들고 중복을 확인하는 일, 일정을 두 층으로 운영하고 변경을 원장에 남기는 일, 유형마다 다른 준비물과 완료 기준을 관리하는 일, 출장과 정산을 이어 붙이는 일, 서류와 서명과 제출물을 다루는 일이 각각 다른 영역이다. 초청 세미나와 평가 업무는 필요한 서류와 확인 순서가 달라 서로 다른 문서에서 절차를 관리한다. 규모가 커진 행사 폴더는 자기 지침 파일과 자기 위키를 갖는 중첩 작업 공간으로 운영할 수 있다. 상위 규칙은 하위 폴더에 자체 규칙이 있을 때 그 규칙을 먼저 읽도록 안내한다. 학회 보직처럼 여러 해와 여러 위원회에 걸친 업무는 한 단계 아래에 독립 작업 공간을 둔다. 협력이 오래 지속되어 자체 규칙과 이력이 필요해지면 독립 작업 공간으로 분리한다. 한 번의 행사로 끝나는 일은 Meetings의 공통 규칙을 적용한다. 하나의 외부 공동연구 미팅을 준비하는 경로는 다음과 같이 이어진다.

준비와 진행에서 나온 결과를 실제 작업 공간에 반영하지 않으면 Meetings만 커지고 Projects, Manuscript와 Grants의 기록은 낡는다. 상대가 보낸 초청과 첨부 파일과 영수증은 행사 폴더에 그대로 보존한다. 여러 행사에서 다시 쓸 수 있는 실수 방지 규칙은 Meetings의 위키에 올린다. 같은 유형의 행사에서 두 번 같은 것을 빠뜨렸다면 규칙으로 남길 후보가 된다. 한 번뿐인 사정과 파일 상태는 해당 행사 폴더에 둔다. 공동연구의 분석과 원고, 과제 자료는 각각 Projects, Manuscript와 Grants에서 계속 관리한다. Meetings에는 그 문서로 가는 링크와 이번 미팅의 자료, 결정, 후속 행동이 남는다. 행사 폴더에 다른 작업 공간의 파일을 계속 복사하면 반년 뒤에는 어느 파일이 최신인지 다시 확인해야 한다.

무엇을 규칙으로 올리질는 네 가지 조건 가운데 하나라도 맞을 때로 정해 두었다. 같은 혼선이 두 번 이상 반복되었거나 한 번이라도 비용이 컸거나 되돌리기 어려웠거나 사람이 앞으로는 이렇게 하라고 교정했거나 다음 세션의 에이전트가 같은 결정을 재현해야 하는 경우다. 올리지 않을 것도 함께 적어 두었는데 그쪽이 실제로는 더 자주 쓰인다. 한 사례에만 해당하는 파일명과 상태, 검증하지 않은 임시 회피책, 특정 도구를 기본값으로 만드는 취향, 검증할 수 없는 감상형 문장이 여기 해당한다. 행사 풀더가 부풀던 시기에 올라오던 것이 대부분 마지막 두 종류였다. 규칙 문장의 형식도 셋으로 고정했다. 어떤 상황에서, 무엇을 하고 무엇으로 확인하는지다. 이 형식에 맞춰 쓰이지 않는 문장은 대개 그 행사의 사연이어서 행사 풀더에 남는다. 승격의 근거는 로그이고 서로 다른 대상 두 건 이상과 로그 항목 세 건 이상에서 같은 위반이 재발하지 않아야 규칙을 굳힌다. 반대 방향도 정의해 두었다. 굳은 규칙이 반복해서 깨지거나 실제 관례와 충돌하면 강제를 권고 수준으로 낮추고 재검토 목록에 올린다.

저희 연구실의 Notion을 열면 연구과제마다 별도의 데이터베이스가 보인다. 학생과 공동연구자는 자신이 참여하는 과제에 들어가 실험 기록, 분석 결과, 미팅 노트, 논문 계획을 함께 갱신한다. 지도교수는 여러 과제의 진행 상황을 같은 형식으로 훑고, 확인이 필요한 카드만 골라 의견을 남긴다. 연구실 전체가 알아야 하는 대학원 과정, 랩미팅, 논문 작성법, 서버 사용법, 연구 행정 담당자 정보는 별도의 연구실 공통 데이터베이스에 모은다.

146

유료 Notion을 운영 비용에 포함하기

저희 연구실은 Notion 유료 버전을 사용한다. 여러 학생과 공동연구자가 과제별 권한을 나누어 쓰고, Notion MCP를 연구실 운영에 연결하기 위해서다. 이 장의 절차를 그대로 적용하려면 유료 구독을 운영 비용에 포함해야 한다. 도입 예산에는 구성원 수에 따른 구독료와 연결할 MCP 서비스의 이용 조건도 함께 계산한다.

Notion MCP는 코덱스와 Claude Code를 비롯한 MCP 지원 에이전트가 사용자의 Notion 페이지를 읽고 갱신하도록 연결한다. 코덱스에서 MCP 서버를 연결하는 방법은 OpenAI MCP 공식 문서에서 볼 수 있다. 에이전트가 볼 수 있는 범위는 연결한 사람의 기존 Notion 권한을 따른다. 외부 공동연구자를 특정 과제에만 초대했다면, 그 사람은 다른 과제 데이터베이스나 연구실 내부 문서를 볼 수 없다. Notion의 연결 방법과 지원 기능은 Notion MCP 공식 안내에서 확인할 수 있다. Notion의 요금제와 AI-통합 기능 범위는 바뀔 수 있으므로 실제 도입 시점에는 요금제 공식 안내도 함께 확인한다.

유료 계정을 준비한 뒤에는 권한부터 정한다. 학생은 자기 과제의 카드를 작성하고, 공동연구자는 공유받은 과제만 열며, 지도교수와 과제 관리자는 전체 진행 상황을 확인한다. 에이전트도 같은 권한 안에서 작동한다. 이 원칙을 지키면 MCP를 연결해도 새로운 공개 범위가 생기지 않는다.

과제 하나에 데이터베이스 하나

저희 연구실은 연구과제를 Notion 운영의 기본 단위로 삼는다. 과제가 시작되면 전용 데이터베이스를 만들고 참여자를 초대한다. 과제 데이터베이스에는 연구계획, 데이터, 분석, 문헌, 미팅, 논문, 행정 기록이 함께 들어간다. 구성원은 한 과제의 현재 상태를 확인하기 위해 여러 서비스와 개인 노트를 돌아다닐 필요가 없다.

데이터베이스에는 보통 세 가지 보기를 만든다.

보기	용도
Workstream 보드	세부 연구 갈래별로 카드를 나누어 진행 상황을 본다
전체 표	담당자, 기한, 문서유형, 상태를 한꺼번에 확인한다
타임라인	마일스톤, 분석 일정, 원고 작성 일정을 시간순으로 본다

가상의 를 예로 들면 Workstream은 , , 으로 나눌 수 있다. 각 열에는 그 연구 갈래에서 실제로 쓰는 카드가 놓인다. 학생은 자신의 분석 카드를 갱신하고, 지도교수는 보드 전체를 보며 막힌 작업과 다음 결정을 확인한다.

하나의 연구과제를 세부 연구 갈래별 열로 나누고, 각 열에 분석, 미팅 노트, 논문과 참고자료 카드를 배치한 실제 Notion 워크스페이스다. 공개하기 어려운 카드 이름은 가렸다.

프로젝트 보드에 만드는 카드

프로젝트마다 연구 내용은 달라도 자주 쓰는 문서의 종류는 비슷하다. 저희 연구실에서는 다음과 같은 카드를 만든다.

문서유형	카드에 담는 내용	가상 예시
Project map(프로젝트 지도)	연구 질문, 세부 과제, 참여자, 주요 일정, 산출물	면역세포 지도 과제 개요
Data hub(데이터 허브)	데이터 위치, 판본, 접근 조건, 담당자	단일세포 데이터 현황
Sample inventory(샘플 목록)	샘플 수, 처리 상태, 제외 사유, 품질 확인	1차 코호트 샘플 목록
Pipeline provenance(분석 이력)	입력 자료, 코드 판본, 실행 조건, 출력물	세포 유형 분류 파이프라인
Analysis result(분석 결과)	질문, 방법, 그림, 해석, 다음 분석	치료 반응군 비교 결과
Manuscript plan(논문 계획)	논문 메시지, 그림 구성, 집필 담당, 목표 일정	면역세포 지도 원고 계획

문서유형	카드에 담는 내용	가상 예시
Literature note(문헌 노트)	논문 요약, 과제 관련성, 주의점, 다음 행동	세포 상태 전이 참고 논문
Meeting note(미팅 노트)	논의 자료, 결정, 할 일, 담당자, 기한	세부 분석팀 미팅
Logistics(행정·운영)	구매, 협약, 자료 이전, 보고 일정	2차년도 보고 준비
Needs review(확인 필요)	지도교수나 공동연구자의 판단이 필요한 항목	샘플 제외 기준 확인
Archive(보관)	종료되었고 검색을 위해 남기는 기록	종료된 예비 분석

카드는 처음부터 완성된 보고서로 만들지 않는다. 분석을 시작할 때 질문과 입력 자료를 적고, 결과가 나오면 그림과 해석을 추가한다. 논의가 끝난 뒤에는 결정과 다음 행동을 남긴다. 한 카드가 작업의 진행 기록과 미팅 자료를 함께 제공하므로, 지난주에 어떤 판단이 바뀌었는지 쉽게 확인할 수 있다.

공동 랩노트와 진도 관리

분석 결과 카드는 공동 랩노트로 쓴다. 학생이 개인 컴퓨터에만 적어 둔 분석 메모는 지도교수와 공동연구자가 볼 수 없고, 다음 미팅에서 설명을 처음부터 다시 해야 한다. 과제 데이터베이스의 카드에 기록하면 참여자가 같은 그림과 같은 설명을 보며 의견을 남길 수 있다. 댓글과 멘션으로 받은 피드백도 카드에 남기므로 다음 분석에서 판단의 근거를 다시 확인할 수 있다.

분석 카드는 다음 항목으로 구성한다.

1. 연구 질문과 이번 분석의 목적
2. 분석 날짜와 입력 데이터 판본
3. 사용한 방법, 코드, 주요 설정
4. 결과 그림과 표
5. 현재 해석과 남은 의문
6. 오류, 품질 문제, 막힌 지점
7. 다음 작업, 담당자, 기한

원시 데이터와 큰 중간 파일은 연구실 서버나 과제 폴더에 보관한다. Notion 카드에는 구성원이 결과를 이해하고 파일을 찾을 수 있는 설명과 링크를 둔다. 데이터 판본과 코드 실행 조건을 함께 적으면, 몇 달 뒤 결과를 다시 만들 때도 당시의 입력과 방법을 복원할 수 있다.

진행상태는 Active, Waiting, Completed, Published, Needs confirmation, Archive처럼 선택형 값으로 관리한다. Waiting에는 시퀀싱 결과 대기, 공동연구자 답변 대기, 데이터 사용 승인 대기처럼 작업이 멈춘 이유를 적는다. Needs confirmation은 샘플 제외, 분석 방향 변경, 외부 공유처럼 책임자의 판단이 필요한 카드에 붙인다. 지도교수는 이 두 상태를 우선 확인하면 된다.

카드 속성을 통일하기

과제가 늘어나면 같은 질문을 여러 데이터베이스에 해야 한다. 이번 달 마감은 무엇인지, 특정 학생이 맡은 작업은 몇 개인지, 원고로 이어진 분석은 어디까지 왔는지 확인하려면 속성 이름이 같아야 한다. 저희 연구실의 과제 보드는 다음 속성을 공통으로 쓴다.

속성	기록하는 내용
이름	카드 제목
Workstream	세부 연구 갈래
문서유형	프로젝트 지도, 분석 결과, 문헌 노트, 미팅 노트 등의 구분
진행상태	Active, Waiting, Completed 등의 현재 상태
담당자	결과와 다음 행동을 책임지는 사람
기한	다음 확인일 또는 마감일

속성	기록하는 내용
관련 원고·메모	이어지는 원고, 분석, 결정 기록
대표 페이지	Workstream에서 먼저 읽어야 하는 기준 문서

모든 카드에 모든 속성을 채울 필요는 없다. 미팅 노트에는 날짜와 참석자, 분석 결과에는 입력 데이터 판본과 관련 원고가 더 중요하다. 공통 속성의 이름을 유지하면 에이전트가 여러 과제를 조회할 때 같은 방식으로 상태를 읽을 수 있다.

LLM 위키에서 프로젝트 Notion으로 논문 공유하기

LLM 위키에 논문이 ingest되면 에이전트는 새 문헌의 제목, 초록, 주요 결과, 태그와 기존 위키 연결을 읽는다. 이어서 연구실의 과제 목록과 참여자 역할을 확인하고, 이 논문을 함께 읽어야 할 학생과 공동연구자를 고른다. 관련성이 확인된 논문은 해당 과제의 페이지에 문헌 노트로 등록한다. 공유 대상은 그 논문의 방법이나 결과를 실제로 검토해야 하는 구성원으로 정한다.

이 과정은 다음 순서로 진행한다.

1. 새로 ingest된 논문의 서지정보와 위키 요약을 읽는다.
2. 논문의 질한, 데이터 유형, 분석 방법, 주요 결과를 현재 과제의 Workstream과 대조한다.
3. 관련 과제와 담당 학생, 공동연구자를 제안한다.
4. 과제별 문헌 노트 초안을 만든다.
5. 담당자가 관련성과 공개 범위를 확인한다.
6. Notion MCP로 과제 데이터베이스에 페이지를 만들고 읽을 사람을 알린다.

문헌 노트에는 다음 내용을 넣는다.

항목	내용
논문 정보	제목, 저자, 저널, 연도, DOI, 공개 주소
한 줄 요약	논문이 보여 준 결과
방법	자료와 분석 방법의 특징
주요 결과	현재 과제와 직접 관련된 결과
프로젝트 관련성	어느 Workstream의 어떤 판단에 도움이 되는지
주의점	표본, 방법, 해석의 제한
다음 행동	재현, 비교, 토론, 추가 문헌 확인 등의 후속 작업

같은 논문이 여러 과제와 관련될 때는 각 과제에 짧은 문헌 노트를 만든다. 예를 들어 한 과제에서는 세포 분류 방법을 검토하는 자료로 쓰고, 다른 과제에서는 코호트 구성을 비교하는 자료로 쓸 수 있다. LLM 위키의 원문 기반 요약은 한 번 유지하고, Notion에는 각 프로젝트에서 필요한 해석과 행동을 적는다. 과제별 접근 권한도 그대로 적용된다.

따라해보기

다음처럼 요청하면 새 문헌을 과제별로 나누는 초안을 만들 수 있다.

```
LLM ingest .
Workstream .

- Workstream
-
-
-
-
```

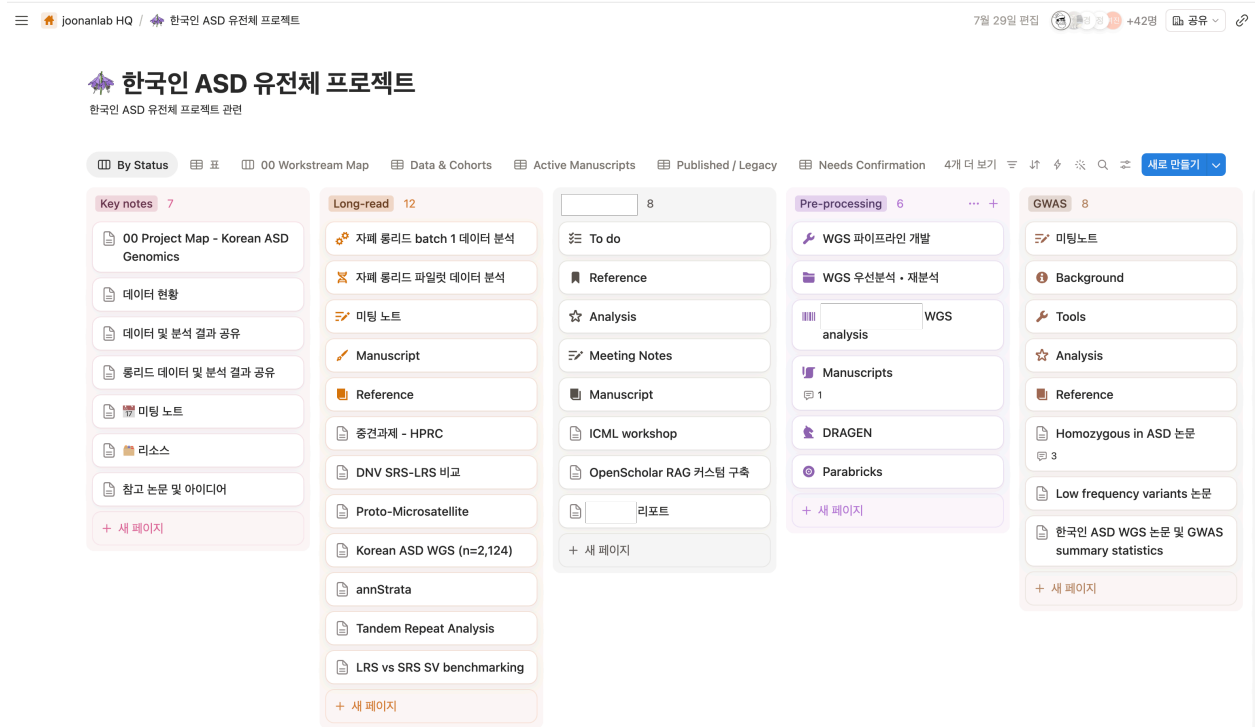


Figure 16: 과제별 Notion 데이터베이스를 Workstream별로 나눈 실제 화면

Notion

미팅 기록을 세 종류로 나누기

과제별 미팅은 성격에 따라 , , 으로 나눈다. 한 페이지 안에서 제목을 구분해 모으거나, 미팅 데이터베이스의 유형 속성으로 필터링할 수 있다.

미팅 종류	참여자 및 목적	가상 예시
총괄 미팅	세부 과제 책임자와 주요 참여자가 전체 진도, 예산, 산출물, 위험 요소를 확인한다	분기 총괄 미팅
세부 미팅	특정 Workstream 참여자가 실험과 분석 결과를 검토하고 다음 작업을 정한다	단일세포 분석팀 미팅
비정기 미팅	오류, 신규 자료, 외부 요청처럼 즉시 판단할 사안을 다룬다	샘플 품질 기준 긴급 논의

미팅 노트의 제목은 날짜와 미팅 유형, 주제를 함께 적는다. [26.08.12] - 처럼 쓰면 검색 결과만 보고도 내용을 짐작할 수 있다. 본문에는 다음 항목을 둔다.

1. 일시와 참석자
2. 안전과 미리 공유한 자료
3. 새로 확인한 사실과 분석 결과
4. 논의에서 확정된 결정

5. 후속 행동, 담당자, 기한
6. 다음 미팅에서 확인할 질문

학생은 미팅 전에 관련 분석 카드를 연결한다. 미팅 중에는 결정을 중심으로 짧게 메모하고, 끝난 뒤 담당자와 기한을 채운다. 공유받은 사람은 발언 순서를 다시 읽지 않아도 무엇이 결정되었고 누가 다음 작업을 맡았는지 알 수 있다.

Zoom MCP에서 Notion 미팅 노트까지

온라인 미팅이 끝난 뒤에는 Zoom MCP와 Notion MCP를 이어서 기록 시간을 줄인다. Zoom의 MCP 서버는 권한이 있는 계정의 회의 자료, 전사, 요약, 후속 항목을 에이전트가 활용할 수 있도록 지원한다. 사용 가능한 자료는 Zoom 계정의 라이선스, 녹화 설정, 회의 소유권과 접근 권한에 따라 달라진다. 현재 기능과 조건은 Zoom MCP Server 공식 문서와 Zoom MCP 연결 안내에서 확인할 수 있다.

저희 연구실에서 사용하는 흐름은 다음과 같다.

1. 회의가 끝나면 에이전트가 Zoom MCP에서 전사, 회의 요약, 다음 행동, 녹화 주소를 가져온다.
2. 회의에 사용한 슬라이드와 연결된 Notion 분석 카드를 함께 읽는다.
3. 전사에서 사실, 해석, 결정, 후속 행동을 구분해 미팅 노트 초안을 만든다.
4. 결정마다 근거가 된 그림이나 문서 링크를 연결한다.
5. 후속 행동에 담당자와 기한을 붙인다.
6. Notion MCP로 해당 과제의 미팅 노트에 초안을 기록한다.
7. 참석자가 결정과 담당자를 확인한 뒤 공유 상태로 바꾼다.

Zoom 전사가 제공되지 않은 회의는 녹음 파일을 로컬 Whisper로 전사한 뒤 같은 절차를 적용할 수 있다. 어느 경로를 쓰든 원문 전사는 보존하고, 공유 노트에는 확인된 결정과 후속 행동을 정리한다. 전사에 없는 내용을 에이전트가 보충하지 않도록 회의 자료와 원문을 함께 확인한다.

설명회 영상에서 Notion 회의 노트까지

이 흐름은 이메일로 받은 설명회 영상에도 적용할 수 있다. 실제로 저는 연구과제 설명회 자료와 영상 주소를 이메일로 받은 뒤 에이전트에게 메일과 영상 네 편을 함께 확인해 달라고 요청했다. 에이전트는 영상을 내려받아 로컬 Whisper로 전사하고, LLM-Wiki에 없던 과제 폴더를 새로 만들었다. 과제 개요는 index.md에 정리하고, 설명회의 주요 내용과 질의응답은 별도 문서로 나누었다. 영상별 원문 전사는 transcripts/에 보존했다. 작업이 끝난 뒤에는 처리한 파일, 전사 결과, 연구계획에 영향을 주는 조건과 추가 확인이 필요한 일정을 보고받았다. 제가 내용을 확인한 다음 에이전트에게 Notion MCP로 연구실의 해당 프로젝트 회의 노트에 올리도록 지시했다. 긴 영상을 다시 재생하지 않아도 과제 참여자가 핵심 조건과 결정 사항을 함께 검토할 수 있는 기록이 만들어졌다.

설명회 영상 네 편을 전사하고 과제별 LLM-Wiki 문서로 정리한 뒤, Notion 게시를 지시한 과정이다. 개인정보와 공개하기 어려운 연구 정보는 가렸다.

Notion 페이지에는 설명회의 핵심 결과, 결정 사항, 확인할 일정과 연구계획에 영향을 주는 조건을 중심으로 올린다. 원문이 필요할 때 돌아갈 수 있도록 LLM-Wiki의 설명회 요약과 전사 문서를 연결한다. 과제 데이터베이스의 보기에서 날짜, 회의 유형, 관련 Workstream과 참석자를 속성으로 지정하면 같은 과제의 다른 회의와 함께 찾을 수 있다. 새로 정리한 페이지는 학생과 공동연구자가 과제의 운영 조건을 이해하는 공통 기록으로 쓰인다. 이후 분석 계획이나 데이터 접근 조건이 바뀌면 관련 카드와 연결해 후속 작업을 갱신한다. 아래 화면은 Notion MCP로 정리한 실제 설명회 노트의 일부다. 검은색으로 가린 부분에는 연구자 수, 데이터 범위와 기관별 운영 조건처럼 공개하기 어려운 정보가 있었다. 공개용 책에는 구조와 활용 방법을 보여 주는 데 필요한 부분만 남겼다.

설명회의 핵심 결과와 결정 사항을 과제별 Notion 미팅 노트에 정리한 예시다. 민감한 수치와 기관 정보는 가렸다.

따라해보기

Zoom MCP 9 .

, ,

.

[국가통합바이오빅데이터구축사업단] 내부연구자 우선개방 설명회 자료 및 영상 공유의 건 이런 메일이 왔구요.

자료는 여기 있어요. 확인해서 위키 형태로 기록해줘요.

그리고 나서 설명회 내용과 질의응답을 전체 대화를 전사해줘요. 그 전사본을 위키로그로 기록해줘요. 나중에 연구를 위해 참고해야해요.

<https://www.youtube.com/watch?v=rUdh7upradg>
<https://www.youtube.com/watch?v=7R8bWA1NIJQ>
<https://www.youtube.com/watch?v=4dUhU6x6IA8>
<https://www.youtube.com/watch?v=NrB8jVm992g>

새로 만든 위키

[wiki/post-award/biko-genome/](#) 아래에 과제를 신설했습니다. 이 과제는 그동안 위키에 아예 없었고 폴더 파일만 있었습니다.

- [index.md](#) — 과제 기본 정보(2024-08~2028-12,), 연구개발 1~4와 연차 계획, 현재 3차년도, 열린 작업
- [priority-access-2026.md](#) — 배포 PDF 98쪽 정리: 운영 개요·일정, 개방 데이터 건수, 임상(OMOP CDM v5.3·품질/표준화), 유전체 (52,003건·파일 유형·QC), 연구환경(VPN/VDI/JupyterHub·디렉터리·주석 DB), 보안·반입반출, 문의처
- [priority-access-qa.md](#) — Q&A 취합 31건 전량 (플랫폼 20·임상 9·유전체 2)
- [transcripts/](#) — 영상 4편 전사본

원문 PDF·XLSX는 과제 폴더의 새 하위 폴더 `우선개방_2026/`에 복사했습니다(덜어쓴 파일 없음). [wiki/index.md](#), [post-award/index.md](#), `log.md`도 갱신했습니다.

전사

`whisper.cpp large-v3-turbo(ko)`로 4편 총 1시간 41분 53초 전량 전사했습니다. 네 편 모두 마지막 발화까지 들어갔고 누락 구간은 없습니다. 1분 단위 타임스탬프 문단이며 화자 분리는 하지 않았습니다. 반복되던 오인식은 사전으로 교정했고(생세포 변이→생식세포 변이, 오목/움푹 CDM→OMOP CDM, 패널과 개념→pheno와 genome 등), 4편 영상 41:49~42:55의 무발화 반복 생성 구간은 표시만 남겼습니다. 중간 산출물(`mp4-wav-vtt`)은 `scratch`에만 두고 보관하지 않았습니다.

음성 질의응답은 4편 영상 36:07 이후에만 있습니다. 나머지 질문은 채팅으로 처리돼 Q&A 취합본으로 갔습니다. 임상데이터 1:N 구조 질문은 현장에서 답을 못 받았고, Q&A 임상 8번이 그 서면 답변입니다.

연구 계획에 걸리는 것들

- 우선개방 기간(8월 말~12월)에는 합니다. 결과 반출이 필요한 분석은 본개방 이후로 잡아야 합니다.
- 12월 종료와 함께 연구환경 저장 파일이 삭제됩니다. 환경 정의(`yaml-requirements`)는 반입본으로 관리하는 게 안전합니다.
- 개방 유전체는 위탁과제의 구조입니다. Plink 포맷도 없지만 폐쇄망 conda로 plink 설치는 됩니다.
- 임상데이터 표준화 기준이 우선개방과 본개방이 다릅니다. 우선개방 매핑 기반 결과는 본개방에서 그대로 재현되지 않습니다.
- 확인이 필요한 일정:

네, 그러면 이 내용을 상세하게 정리해서 국정박막 노선에 올려주세요

Figure 17: 설명회 영상의 전사와 LLM-Wiki 정리를 에이전트에게 맡긴 실제 사례

[26/07/28] 국바빅 우선개방 설명회

핵심 결과

본 설명회는 본격적인 대외 공개방 이전 내부 연구자들이 실제 바이오빅데이터 플랫폼에서 데이터를 직접 활용하며 품질-안정성을 검증하는 사전 운영 단계로 개최되었다. 약 []의 연구자가 우선 개방에 신청하였으며, 임상 데이터(CDM 기반)와 유전체 데이터([])가 []개방될 예정이다. 데이터는 폐쇄망 기반 가상 연구 환경(VPN+VDI)을 통해서만 접근 가능하며, 이용 기간은 []까지로 종료 후 연장은 불가하다.

결정 사항

- 우선 개방 목적: 단순 데이터 선공개가 아닌, 데이터 품질·연구 활용 가능성·분석 환경 안정성을 사전 검증하고 연구자 의견을 공개방에 반영하는 과정
- 이용 대상: []
- 데이터 개방 범위: 전체 데이터 대상이며 표준화·품질관리 일정에 따라 [] 두 차례 개방
- 가상 환경 제공 단위: []
- 이용 리포트 제출: [] 온라인 형식으로 간소화 예정
- 비용: 현재 분석 비용 과금 없음; GPU 추가 사용도 별도 과금 없음

Figure 18: Notion MCP로 프로젝트 미팅 노트에 정리한 설명회 기록

- 1.
- 2.
- 3.
4. , ,
- 5.

Notion

연구실 전체가 함께 보는 공통 Notion

과제 데이터베이스와 별도로 연구실 공통 데이터베이스를 운영한다. 구성원이 바뀌어도 계속 참고해야 하는 규정, 절차, 연락처, 교육 자료를 모으는 곳이다. 저희 연구실에서는 다음 범주를 관리한다.

범주	기록 예시
대학원 과정	수강, 학위 요건, 자격시험, 연구윤리, 졸업 절차
랩미팅과 스터디	연간 일정, 발표 순서, 저널클럽, 교육 자료
논문 작성	원고 폴더 구성, 그림과 표, 참고문헌, 투고 점검표
컴퓨팅과 서버	계정 신청, GPU 사용, 저장 공간, 백업, 장애 대응
연구 행정	구매, 연구비, 과제 보고, 행정 담당자, 자주 쓰는 양식
연구실 운영	신규 구성원 안내, 공용 장비, 예약, 안전, 연락 체계
AI 도구	LLM 위키, MCP 연결, 에이전트 사용 규칙, 데이터 보안

새 학생이 들어오면 공통 Notion에서 대학원 과정과 서버 사용법을 찾는다. 과제에 합류하면 해당 과제 데이터베이스에서

프로젝트 지도, 대표 분석, 참고 논문, 최근 미팅 노트를 읽는다. 에이전트에게 질문할 때도 정보의 종류에 따라 조회할 곳이 정해진다.

질문	먼저 확인할 곳
이 논문이 무엇을 보고했는가	LLM 위키
이 논문을 현재 과제에서 어떻게 쓸 것인가	과제별 Notion 문헌 노트
분석과 원고가 어디까지 진행됐는가	과제별 Notion 보드
연구실의 서버와 행정 절차는 무엇인가	연구실 공통 Notion

이 구분 덕분에 에이전트는 문헌 근거, 프로젝트 진행 상황, 연구실 운영 규칙을 필요한 순서로 읽을 수 있다. LLM 위키에서 찾은 논문은 과제별 문헌 노트로 이어지고, 미팅에서 정한 작업은 분석 카드와 원고 계획에 연결된다. Notion MCP는 이 연결을 사람이 매번 복사해 옮기는 시간을 줄여 준다.

실제 운영 순서

한 주의 운영은 다음처럼 이어진다.

1. 학생이 분석 카드에 새 결과와 문제, 다음 작업을 기록한다.
2. 에이전트가 새로 ingest된 논문을 확인하고 관련 과제와 읽을 사람을 제안한다.
3. 담당자가 문헌 노트 초안을 확인하고 과제 Notion에 공유한다.
4. 미팅 전에 분석 카드와 문헌 노트를 안건에 연결한다.
5. 미팅 뒤 Zoom MCP의 전사와 자료를 이용해 노트 초안을 만든다.
6. 참석자가 결정, 담당자, 기한을 확인한다.
7. 확정된 후속 행동을 기존 분석 카드와 원고 계획에 연결한다.

이 과정이 반복되면 Notion 보드만 보아도 과제의 연구 질문, 최신 결과, 막힌 작업, 읽어야 할 논문, 최근 결정이 함께 보인다. 지도교수는 모든 세부 기록을 처음부터 읽지 않고 Waiting과 Needs confirmation을 먼저 확인할 수 있다. 학생과 공동연구자는 자신이 맡은 카드에서 다음 행동을 찾는다. LLM 위키와 Notion이 MCP로 연결되면서 논문을 읽은 결과가 실제 과제의 결정과 작업으로 이어진다.

26장. LLM-Wiki와 에이전트를 이용하여 연구과제 관리하기

그랜트는 여러 해에 걸쳐 상태가 바뀌는 연구 프로그램이고 기획, 제출, 심사, 수행, 연차 관리, 종료로 국면이 넘어갈 때마다 앞 국면이 남긴 것을 꺼내 쓴다. 제안서를 내고 몇 달 뒤 선정 통보를 받는다. 두어 해가 지나 연차 보고를 쓰는 자리에서 두 번째 Aim의 성공 기준을 왜 그 값으로 잡았는지 설명해야 하는 상황이 온다. 제안서 파일을 열면 그렇게 쓴 문장은 있는데 그 문장이 어떤 논문의 어떤 결과에서 나왔는지, 당시 어떤 대안을 검토하고 무엇 때문에 버렸는지는 어디에도 없다. 함께 쓴 사람들은 다른 일로 옮겨 갔고 그때 주고받은 메일을 뒤져도 어느 첨부가 최종 제출본인지 확인할 수 없다. 결국 근거를 다시 만들게 되는데 새로 만든 근거는 제출 당시의 판단과 미세하게 어긋나고 그 어긋남을 설명할 방법도 남아 있지 않다. 아래 사례는 여러 운영 경험에서 반복되는 구조만 남기고 만들었으며 사업명, 기관, 금액, 기간, 심사 결과, 참여자는 쓰지 않는다. 사례로 삼는 과제는 여러 해에 걸친 다년 과제이고 Aim이 셋이며 각 Aim이 서로 다른 종류의 증거에 의존한다는 조건만 갖는다. 나라와 제도에 따라 서식과 규정은 크게 다르지만 국면이 바뀌는 순서와 국면이 바뀔 때 잃어버리는 것은 대체로 비슷하다.

생애주기로 보는 그랜트

그랜트를 관리한다는 말이 대체로 제안서 파일 하나를 관리한다는 뜻으로 쓰인다. 그 파일이 완성되어 제출되면 관리가 끝난 것처럼 느껴지고 실제로 많은 폴더가 제출일 이후로 갱신되지 않는다. 그런데 제출은 전체 여정의 중간에 있는 한 시점이고 그 뒤로 심사 대응, 수행, 연차 보고, 변경 관리, 종료가 남아 있다. 각 국면은 앞 국면이 남긴 것을 필요로 하는데 남긴 것이 파일 한 개뿐이면 필요한 것을 대부분 다시 만들어야 한다. 과제는 다음 순서로 상태를 바꾼다. 각 상태에는 세 가지를 적어 둔다. 이 상태로 들어오는 조건, 이 상태를 끝냈다고 말할 수 있는 조건, 그리고 그 판정을 내리는 사람이다.

진입 조건과 완료 조건을 적어 두지 않으면 상태 이름이 느낌에 따라 붙고 같은 폴더를 두 사람이 서로 다른 단계로 이해하게 된다. 판정자를 적어 두는 이유는 상태 변경이 대부분 외부 사실에 달려 있기 때문이다. 선정 여부, 협약 체결, 변경 승인은 연구실 안에서 결정되는 일이 아니므로 확인하지 않은 채 상태를 앞당기면 폴더가 실제보다 낙관적인 그림을 그리게 된다. 폴더의 상태를 실제 행정 상태보다 먼저 바꾸지 않는다는 규칙은 사소해 보이지만 지키기 어렵다. 발표가 잘 끝났다는 느낌만으로 수행 단계 폴더를 만들어 두거나 사실상 끝난 과제를 아직 진행 중인 자리에 두는 일이 반복된다. 상태가 실제와 어긋나면 그 폴더를 근거로 만든 모든 요약이 함께 어긋나고 다음 세션의 에이전트는 어긋난 상태를 사실로 읽는다. 상태 변경은 통보 문서, 협약, 공식 회신처럼 확인 가능한 사건에 붙인다. 확인되지 않은 동안에는 상태를 그대로 두고 무엇을 기다리는 중인지만 적는다.

```
{ }_{ }_{ }/
{ }/
{ }/
```

폴더 이름은 예시이므로 상태를 나타내는 말은 각자 쓰는 표현으로 정하면 되고 정하고 나면 바꾸지 않는 것이 이름 규칙의 전부다. 다섯 단계가 최상위에 그대로 드러나 있어서 목록을 한 번 내려 보면 지금 손대고 있는 것과 제안 단계와 수행 단계와 끝난 것이 한눈에 갈린다. 공통 행정 자산은 어느 단계에도 속하지 않으므로 생애주기 밖에 따로 둔다. 기관 서식이나 반복해서 쓰는 증빙처럼 과제가 바뀌어도 그대로 쓰는 것이 여기에 들어가고 개별 과제 폴더에 두면 그 과제를 종료 처리할 때 함께 묻힌다. 위키 아래는 제안서와 수행 과제와 회의록으로 갈리고 생애주기와 명명 규칙, 발표 자료 규칙, 참고 자료가 각각 자리를 갖는다. 제안서와 수행 과제가 목록 색인을 따로 갖는 이유는 두 국면에서 열어 봐야 하는 문서가 다르기 때문이다. 제안서 하나마다 위키에 페이지 묶음이 하나씩 붙고 그 묶음이 그 제안서의 통제층이 된다. 최상위 규칙 파일은 강제 규칙과 어디로 갈지만 담고 단계 사이의 이동은 여기서도 사람이 승인한 뒤에 실행한다.

따라해보기

아래 블록은 AI 도구의 채팅창에 그대로 붙여 넣는 프롬프트다. 그랜트 작업 공간을 처음 세울 때 쓰는 프롬프트다. 꺾쇠 안은 자기 경로와 표기로 바꾼다.

```
< > .
- AGENTS.md wiki/
-
- , , , ,
-
- wiki/ index.md log.md proposals/, post-award/, meeting-minutes/,
  lifecycle/, presentations/, references/
- proposals/ index, timeline, action register,
  communications, scientific plan,
- AGENTS.md
  1.
  2. wiki/log.md
  3.
```

과제의 현재 상태를 통제하는 최소 문서

과제 하나를 운영하려면 서로 다른 갱신 주기를 가진 문서 몇 개가 필요하다. 이 문서 묶음을 control plane, 곧 과제의 현재 상태와 다음 행동을 결정하는 통제층이라고 부른다. 통제층을 나누는 기준은 갱신 단위다. 마감은 자주 바뀌고 과학적 계획은 드물게 바뀌며 자료의 위치는 파일을 만들 때마다 바뀐다. 갱신 주기가 다른 것들을 한 문서에 모아놓으면 그 문서는 매번 통째로 다시 읽어야 하고 결국 아무도 읽지 않게 된다. 제안서 하나에 붙는 묶음은 여섯 종류다.

문서	답하는 질문
과제 index	현재 상태는 무엇이고 제출일은 언제이며 기준이 되는 작업본은 어느 파일인가
Timeline	외부 마감에서 역산한 내부 milestone은 무엇인가
Action register	누가 무엇을 언제까지 어떤 결과물로 끝내는가
Communications	외부와 무엇을 주고받았고 무엇이 합의되거나 바뀌었는가
Scientific plan	Aim, 근거, 방법, 위험, 대안이 어떻게 연결되는가
자료와 렌더링 점검	원본 양식과 작업본과 참고자료가 어디에 있고 제출 파일이 제대로 렌더링되는가

과제 index는 얇게 유지한다. 현재 상태, 기준이 되는 작업본, 다음 제출일, 지금 가장 크게 열려 있는 미결, 그리고 나머지 문서로 가는 링크면 충분하다. 새 정보가 들어올 때마다 index에 문단을 덧붙이면 몇 주 만에 아무도 끝까지 읽지 않는 긴 문서가 되고 그러면 통제층이 있으나 마나 해진다. 여기에 결정 기록을 일곱 번째 문서로 더하고 싶어서는 작업 폴더

안에는 진행 메모, 변경 기록, 피드백 파일을 만들지 않는다. 결정과 그 이유는 작업 공간 위키의 로그 문서에 덧붙인다. 로그 항목의 형식은 여덟 항목으로 고정되어 있다. 무엇이 이 작업을 촉발했는지, 그때 원본이 어떤 상태였는지, 무엇을 실행했는지, 무엇을 확인했는지, 무엇을 발견했는지, 무엇을 결정했는지, 무엇이 남았는지, 어떤 산출물이 나왔는지다. 확인 결과에는 다섯 가지 상태 라벨을 붙인다. 확인됨, 불일치, 미확인, 보류, 재실행필요다. 그랜트에서 이 라벨이 특히 값을 하는 이유는, 확인하지 못한 것을 이상 없음으로 적는 순간 그 항목이 마감 전날에 되살아나기 때문이다. 상대 기관의 약속서를 아직 못 받은 상태와 받아서 내용을 대조한 상태는 전혀 다른데, 둘 다 완료로 적히면 그 구분이 사라진다.

1. 과제 index, timeline, action register, 자료 목록의 최소 골격을 만든다
2. 각 문서는 한 화면을 넘기지 않게 시작한다
3. 지금까지 한 일 가운데 하나를 골라 여덟 항목 형식으로 로그에 적는다
4. 확인 결과에 다섯 라벨 가운데 하나를 붙인다

성공 기준은 새 세션을 시작한 에이전트에게 이 과제의 현재 상태와 다음 마감, 지금 가장 크게 비어 있는 곳을 물었을 때 세 가지를 모두 문서에서 찾아 답할 수 있는가다. 파일을 두는 폴더도 역할로 나눈다. 기관이 배포한 원본 양식은 받은 그대로 원본 폴더에 두고 절대 덮어쓰지 않으며 손을 대야 할 때는 새 파일명으로 작업본을 만든다. 원본을 한 번 덮어쓰면 나중에 서식이 바뀌었는지 우리가 바꿨는지 구분할 수 없고 서식이 어긋나 반려되면 원인을 찾는 데 며칠이 든다. 지금 제출하는 산출물은 작업 폴더의 루트에 둔다. 최신본을 초안 폴더나 현재본 폴더 아래로 한 단계 내리면 폴더를 열었을 때 무엇을 제출하는 중인지 몇 초 안에 보이지 않고 급할 때 엉뚱한 파일이 올라간다. 새 버전이 나오면 직전 버전을 같은 레벨의 지난 판 폴더로 밀어내고 분석 중간 산출물과 임시 파일은 그 아래나 작업 공간 밖의 임시 폴더로 내린다. 작업 폴더 루트에 임시 폴더를 새로 만들지 않는 것이 이 배치를 유지하는 유일한 방법이다. 파일 이름에는 문서명과 버전만 넣고 무엇을 했는지는 붙이지 않는다. 수정본, 심사의견반영, 최종수정 같은 말이 이름에 붙기 시작하면 이름이 길어지고 정렬이 무너지며 며칠 지나면 어느 것이 최신인지 이름만으로는 판정할 수 없다. 무엇을 왜 바꿨고 어떻게 검증했는지는 로그가 담고 파일 이름은 어느 문서의 몇 번째 판인지만 말한다. 이 구분은 23장에서 정한 owner와 SSOT의 구분을 파일 층위에 그대로 적용한 것이다.

먼저 확정하는 공고 적합성

제안서를 쓰기 전에 확정할 것은 이 공고에서 우리가 무엇을 약속할 수 있는가다. 지원 자격, 연구 범위, 수행 기간, 예산의 한도와 항목 제약, 반드시 포함해야 하는 참여 조건, 제출 형식과 분량, 그리고 공식 마감이다. 각 항목은 반드시 지금 유효한 공고 원문에서 확인한다. 지난번에 냈던 제안서나 예전 공고의 기억을 현재 규정으로 쓰는 일이 실무에서 가장 흔한 사고이고 형식 요건 하나가 바뀐 것을 놓쳐 제출 자체가 막히는 경우가 있다. 적합성 판단에서는 두 문장을 분리해야 한다. 우리 연구가 이 공고와 관련이 있다는 문장과, 이 공고에서 경쟁력 있는 약속을 할 수 있다는 문장은 전혀 다른 주장이다. 뒤의 문장은 필요한 예비 결과가 이미 있는지, 약속한 산출물을 기간 안에 낼 수 있는지, 심사자가 물을 만한 반론에 답할 수 있는지를 묻는다. 두 문장을 섞으면 관련성만으로 착수했다가 제출 직전에 예비 결과가 없다는 사실을 발견하게 된다.

에이전트가 이 단계에서 잘하는 일은 대조다. 공고 원문의 요건 목록과 현재 우리가 가진 자료를 나란히 놓고 빈칸을 찾는 일, 이미 수행 중인 과제와 연구 내용이 겹치는지 확인하는 일, 예비 결과가 필요한 Aim이 어느 것인지 표시하는 일은 사람이 빠뜨리기 쉽고 기계는 빠뜨리지 않는다. 반대로 이 공고에 낼 만한가라는 판단은 기관 사정, 사람의 여력, 다른 과제와의 관계가 얽힌 문제여서 사람이 한다. 대조 결과를 받아 판단을 내리고 판단의 이유를 로그에 남기는 순서가 된다. 지원하지 않기로 한 결정도 반드시 남긴다. 이유 없이 지나간 공고는 다음 회차에 다시 검토 대상으로 올라오고 같은 논의를 처음부터 반복하게 된다. 자격이 맞지 않아서인지, 예비 결과가 부족해서인지, 시기가 겹쳐서인지를 한 줄로 적어 두면 다음 회차에는 그 한 줄이 바뀌었는지만 확인하면 된다. 예비 결과가 부족해서 포기한 경우라면 그 부족한 결과가 다음 해의 연구 계획에 항목으로 들어간다.

위키의 근거에서 Aim으로의 전환

Aim은 그것을 지탱하는 과학적 근거와 명시적으로 이어 두어야 나중에 다시 쓸 수 있다. 제안서를 쓰는 동안에는 근거가 머릿속에 있어 연결이 필요 없어 보이는데 그 머릿속 연결은 제출과 동시에 사라진다. 심사자가 특정 Aim의 전제를 물었을 때, 연차 보고에서 계획 대비 진척을 설명해야 할 때, 후속 과제에서 같은 논리를 다시 쓸 때 필요한 것은 제안서 문장 뒤에 있던 근거다. 근거와 주장을 짝지어 두는 구조는 19장에서 다뤘고 여기서는 그 구조를 Aim 단위로 적용한다. 위키에 잘 정리된 최신 결과가 있으면 그 결과를 중심으로 Aim을 세우고 싶어진다. 그런데 제안서의 Aim은 지원 사업의 목적과 우리가 실제로 밟을 수 있는 결정 경로를 함께 만족해야 한다. 세 해 안에 결론에 도달할 수 없는 질문, 우리 손에 데이터가 오지 않는 질문, 결과가 어느 쪽으로 나와도 다음 결정이 달라지지 않는 질문은 흥미롭더라도 Aim이 되지 못한다. 여기서

역할이 갈린다. 문헌 근거와 그 근거를 종합한 페이지는 문헌 위키가 소유하고 제안서의 파일과 버전과 검증 절차는 과제 쪽 작업 공간이 소유한다. 두 자리는 서로의 내용을 복사하지 않고 절대경로로 가리키며 과제 쪽 라우팅 표에는 이 위키에 중복해 적지 않는다는 열이 따로 있다. 참조는 양방향이어서 문헌 위키의 아젠다 문서도 과제 폴더의 경로를 되가리킨다. 근거를 과제 폴더로 복사해 두면 원문 쪽 종합 페이지가 갱신되었을 때 제안서 쪽 사본만 옛 판단으로 남고 그 사실을 알아차릴 방법이 없다. 절차는 여섯 단계로 진행된다.

1. 과제의 중심 질문을 완전한 문장 형태의 연구 질문으로 만들어 위키를 검색한다. 답을 요구하는 질문으로 검색해야 관련된 종합 문서가 걸린다.
2. 검색된 overview와 concept를 읽고 거기서 인용하는 원문 기록으로 내려가 핵심 주장의 출처를 확인한다.
3. 알려진 사실, 아직 논쟁 중인 사실, 실제로 비어 있는 자리를 구분한다. 세 가지를 섞으면 이미 답이 난 질문을 gap이라고 주장하게 된다.
4. 비어 있는 자리를 Aim, 기대하는 결과, 그리고 결과가 다르게 나왔을 때의 대안 해석으로 바꾼다.
5. 각 Aim에 대해 지금 가지고 있는 예비 근거와 아직 없는 근거를 나눠 적는다.
6. 제안서에 들어갈 인용과 강한 주장은 원문으로 다시 확인한다.

여섯 번째 단계를 건너뛰는 유혹이 크다. 정리 기록에 이미 요약이 있고 그 요약이 대체로 정확하기 때문이다. 그러나 제안서에 들어가는 주장은 심사자가 원문과 대조할 수 있는 주장이고 요약 단계에서 조건이 하나 빠진 문장은 원문 앞에서 과장으로 읽힌다. 어떤 방법이 특정 조건에서만 성립했는데 조건을 떼고 인용하면, 그 한 문장 때문에 Aim 전체의 신뢰가 흔들린다. 이 작업의 산출물은 Aim과 근거를 맞춘 표다. 각 행은 하나의 주장이고 열에는 그 주장을 지탱하는 근거, 근거의 출처, 근거의 강도, 아직 없는 근거, 그 근거를 언제 확보할 계획인지가 들어간다. 표가 완성되면 어떤 Aim이 실제로는 근거 하나에만 매달려 있는지, 예비 결과가 없다고 표시된 칸이 몇 개인지가 눈에 보인다. 근거 하나에만 매달린 Aim은 심사에서 가장 먼저 공격받는다. 예비 결과가 없다고 표시된 칸은 제출 전에 확보하거나 계획 안에서 정직하게 인정해야 한다. 그 표에서 아직 없는 근거 칸이 최소 하나 비어 있어야 하고 모든 칸이 채워졌다면 근거를 정직하게 평가하지 않았을 가능성이 높다.

1. Aim 하나를 고른다
2. 주장, 근거, 근거의 출처, 아직 없는 근거, 확보 계획의 다섯 칸으로 분해한다

역할과 점점으로 설계하는 참여 기관

여러 기관이 함께 제출하는 형태는 어느 제도에서나 반복된다. 이때 가장 먼저 적고 싶어지는 것은 기관별 전문성인데, 전문성 목록은 제안서 문장으로는 필요해도 운영에는 거의 쓸모가 없다. 운영에 필요한 것은 각 참여자가 무엇을 언제 넘겨주는가와 그것이 오지 않으면 무엇이 막히는가다. 산출물과 의존 관계를 먼저 적고 나면 전문성 문장은 그 표에서 자동으로 따라 나온다. 역할은 책임으로 나눈다. 과학적 방향을 정하는 사람, 행정 절차를 책임지는 사람, 데이터를 보유한 쪽, 분석을 수행하는 쪽, 시료를 관리하는 쪽, 예산을 집행하는 쪽은 서로 다른 책임이고 종종 서로 다른 기관에 있다. 한 사람이 여러 책임을 겸하더라도 표에서는 분리해 둔다. 검직을 하나로 합쳐 적으면 그 사람이 빠졌을 때 무엇이 함께 사라지는지 알 수 없다. 책임을 분리해 두면 담당이 바뀌었을 때 바뀐 칸만 고치면 된다.

여러 기관이 얹히면 시간이 길어진다. 확약서 한 장을 받는 데 걸리는 시간, 서명을 받는 데 걸리는 시간, 기관 내부 검토가 요구하는 여유는 연구자가 통제할 수 없고 대체로 예상보다 길다. 그래서 외부 마감에서 역산할 때 과학적 작업보다 이런 절차의 소요 시간을 먼저 자리에 놓는다. 마감 전 마지막 주에 서명을 요청하는 계획은 희망에 가깝다. 아직 오지 않은 숫자와 문서는 예정으로 표시하고 비워 둔다. 상대 기관이 아직 확정하지 않은 예산 항목, 아직 회신하지 않은 참여 의사, 아직 받지 않은 실적 수치를 그럴듯한 값으로 채우면 그 값이 며칠 뒤 확정값처럼 읽힌다. 에이전트에게 초안 정리를 맡길 때 특히 위험한데, 빈칸이 있는 표를 주면 모양을 맞추려고 값을 채우는 경향이 있기 때문이다. 확정되지 않은 partner 기여는 빈칸으로 두고 미확인 라벨을 붙인 다음 누구에게 무엇을 요청한 상태인지를 함께 적는 편이, 채워진 표를 나중에 검증하는 것보다 훨씬 싸다.

실행 항목과 집중 시간으로 바꾸는 마감

외부 마감 하나는 그 자체로 실행할 수 없는 항목이다. 마감에서 거꾸로 세어 과학적 검토, 예산 확정, 행정 검토, 서명, 업로드, 최종 확인의 자리를 잡아야 비로소 일이 된다. 각 자리에는 그 일을 끝냈다고 말할 수 있는 조건을 붙인다. 예산 확정 항목별 근거가 붙고 집행 규정에 어긋나지 않는지 확인된 상태이고 최종 확인은 올라간 파일이 우리가 의도한 그 파일인지 다시 열어 본 상태다. 완료 조건을 적지 않으면 모든 항목이 마감 당일에 동시에 걸린다. 일정표에 과제 이름을 통째로 넣는 방식은 잘 작동하지 않는다. 큰 이름이 적힌 세 시간 블록은 무엇을 하면 끝나는지 알 수 없어서 다른 일에

잠식된다. 두 번째 Aim의 논리 수정, 예산 항목과 인건비 대조, 협력 기관 협약서 검토처럼 끝났는지 아닌지가 판정되는 이름을 붙인다. 참여 기관이 함께 모이는 회의와 혼자 준비하는 시간은 다른 층에 둔다. 옮길 수 없는 회의와 옮길 수 있는 준비를 같은 목록에 두면 마감이 가까워질 때 준비 시간이 먼저 밀린다. 일정이 바뀌면 세 곳이 함께 바뀌어야 한다. Timeline의 milestone, action register의 기한, 그리고 실제 일정표다. 셋 중 하나만 고치는 일이 반복되면 어느 것이 현재 계획인지 알 수 없게 되고 마감에 가까워질수록 이 불일치가 비싸진다. 세 곳을 대조해 서로 어긋나는 항목, 담당이 없는 항목, 완료 조건이 비어 있는 항목, 이미 지난 기한을 찾아내는 일은 규칙이 분명하고 사람이 성실하게 하기 어려워서 에이전트에게 맡길 만하다. 다만 무엇을 미루고 무엇을 살릴지는 사람이 정하고 제출 버튼을 누르는 일도 결과의 책임이 사람에게 있으므로 사람이 확인하고 사람이 실행한다.

1. 외부 마감에서 역산해 과학 갈래와 행정 갈래를 나눈다
2. 각 갈래의 항목에 완료 조건을 붙인다

심사 국면의 version 보존

단계가 나뉜 심사는 여러 제도에서 반복되는 형태다. 서면 심사를 통과하면 발표나 면담이 이어지고 그 사이에 자료를 보완하거나 질의에 답해야 한다. 이 국면에서 가장 자주 일어나는 사고는 새로 만든 자료가 제출본의 약속과 어긋나는 것이다. 발표 자료를 만들면서 메시지를 다들다 보면 제출본에는 없던 표현이 들어가거나 제출본이 조심스럽게 쓴 대목이 발표에서는 단정도로 바뀐다. 그래서 심사 국면의 통제층에는 네 가지가 연결되어 있어야 한다. 지금 작업 중인 파일, 실제로 제출한 파일, 심사 측이나 주최 측이 요청한 내용, 그리고 발표에서 쓸 근거의 출처다. 제출본은 손대지 않고 지난 판 자리에 그대로 보존한다. 되돌릴 수단이 동기화 서비스의 버전 기록본이므로 제출본을 실수로 덮어쓰면 복구가 며칠 전 판본으로 후퇴하는 일이 되고 그사이의 수정이 함께 사라진다. 발표 자료는 제출본의 어느 대목을 어떻게 옮긴 것인지 표시한다. 새로 만든 숫자나 그림이 있다면 제출본의 어떤 값과 대응하는지, 값이 달라졌다면 왜 달라졌는지를 적는다. 값이 갱신되었다는 사실 자체는 괜찮고 갱신을 설명할 수 없으면 그 값을 근거로 쓸 수 없다.

검증 가능한 요구는 따로 모아 별도의 확인 목록으로 관리한다. 분량 제한, 서식, 파일 형식, 발표 시간처럼 맞았는지 틀렸는지가 기계적으로 판정되는 항목이 여기에 들어간다. 내용 논의가 길어지면 형식 확인은 마지막 날로 밀리고 마지막 날에 형식 문제가 발견되면 고칠 시간이 없다. 형식 확인은 내용과 무관하게 정해진 시점에 독립적으로 수행하고 결과를 파일 하나 단위로 확인됨이나 불일치로 적는다. 발표 국면에서 언어 모델이 만든 문장의 유혹이 특히 크다. 매끄러운 요약 문장은 몇 초 만에 나오고 대체로 그럴듯하다. 그러나 이 국면에서 먼저 볼 것은 현재 version의 그림, 숫자, 예산, 인용이 서로 맞는지다. 발표에서 말한 수치가 제출본의 표와 다르면 그때 신뢰를 잃고 매끄러운 문장은 그 손실을 보상하지 못한다. 문장을 다듬는 일은 정합성 확인이 끝난 뒤에 한다.

합성 사례의 첫 국면인 제안서 심사

앞서 밝힌 합성 사례의 과제는 Aim이 셋이고 각 Aim이 서로 다른 종류의 증거에 기댄다. 첫 번째 Aim은 이미 공개된 데이터의 재분석으로 뒷받침되고 두 번째 Aim은 아직 수집하지 않은 데이터를 전제로 하며 세 번째 Aim은 다른 기관이 보유한 자료에 의존한다. 서면 단계를 통과해 발표 국면에 들어갔을 때, 세 Aim은 준비의 성격이 완전히 다르다. 첫 번째는 결과를 보여 주면 되고 두 번째는 왜 지금 없는 데이터를 확보할 수 있는지 설명해야 하며 세 번째는 우리가 통제하지 못하는 조건을 어떻게 관리할 것인지 답해야 한다. 세 Aim을 같은 방식으로 준비하면 가장 약한 하나가 나머지 둘의 평가까지 끌어내린다. 이 차이를 실행 항목으로 옮기면 네 갈래가 나온다. 과학적 논리를 다듬는 일, 그림과 수치를 현재 version에 맞추는 일, 예산과 인력 계획을 발표 내용과 맞추는 일, 그리고 형식과 행정 요건을 확인하는 일이다. Action register에서 갈래를 나누고 각각에 담당과 완료 조건을 붙이면, 과학 논의가 길어져도 나머지 셋의 진척이 보인다. 발표 준비에서 따로 정해 두어야 하는 것이 근거의 경계다. 지금 말할 수 있는 것, 조건을 붙여야 말할 수 있는 것, 아직 말할 수 없는 것을 미리 셋으로 나눠 둔다. 질문을 받으면 아직 없는 결과를 있는 것처럼 말하고 싶어지는데 그때는 통과하더라도 수행 단계에서 그대로 부메랑이 된다. 에이전트에게 예상 질문 목록을 만들게 할 수는 있지만 어느 답변이 우리가 감당할 수 있는 약속인지는 사람이 판정한다.

합성 사례의 두 번째 국면인 공동 집필 제안서

같은 과제가 여러 기관이 함께 쓰는 형태로 진행되면 문제의 성격이 바뀐다. 원고는 여러 곳에서 동시에 자라고 각 부분의 최신본이 서로 다른 곳에 있으며 분량 제한은 전체에 걸리는데 작성은 부분별로 이루어진다. 한 사람이 자기 절을 늘리면 다른 절이 줄어야 하는데 그 조정을 마감 직전에 하면 문장부터 잘려 나간다. 어느 절이 아직 초안인지, 어느 절이 확정인지,

확정된 절의 파일이 어느 것인지가 보이면 조정이 내용 수준에서 가능해진다. 과학 track과 행정 track은 나란히 굴러가되 섞이지 않게 한다. 과학 쪽은 Aim과 방법과 근거를, 행정 쪽은 서식과 자격 서류와 요약서와 예산 양식과 서명을 다루는데 진행 속도가 다르고 막히는 이유도 달라서 한 문서에 적으면 서로의 진척을 가린다. 다만 마지막에는 합쳐서 본다. 과학 쪽에서 늘린 인력이 예산 쪽에 반영되었는지, 행정 쪽에서 바뀐 기간이 연구 계획의 일정과 맞지는 각 track 안에서는 보이지 않고 통합 검수에서만 보인다. 어느 파일이 현재 작업본이고 어느 것이 지난 판인지, 어느 파일이 기관에서 받은 원본인지가 목록에 있어야 여러 사람이 같은 파일을 고친다. 문서명과 버전만 담은 이름이 여기서 값을 하는데 작성자 이름이나 수정 내용이 이름에 붙은 파일이 여러 기관에서 동시에 돌아오면 정렬만으로는 최신본을 고를 수 없기 때문이다. 외부에서 온 요청과 합의는 메일 본문에 흩어져 있으므로 요청 내용, 요청한 쪽, 회신 기한, 현재 상태를 한 줄씩 옮겨 적는다. 옮겨 적는 일이 번거로워 보이지만 마감 사흘 전에 어떤 요약서가 아직 안 왔는지 메일함에서 찾는 일과 비교하면 싸다. 에이전트가 잘하는 일이 바로 이 대조여서 자료 목록과 소통 기록과 실행 항목을 함께 읽고 빠진 것을 찾아 달라고 요청하면 사람이 놓친 항목이 대체로 나온다.

선정 이후 달라지는 관리의 종류

선정되면 같은 연구가 다른 규칙 위로 옮겨 간다. 제안서 단계에서 중요했던 것이 설득력이었다면 수행 단계에서 중요한 것은 약속한 것을 실제로 하고 그것을 증명하는 일이다. 참여 인력의 등록과 변경, 인건비의 지급 근거, 집행할 수 있는 항목과 없는 항목, 증빙 서류, 기간을 넘긴 예산의 처리처럼 제안서를 쓸 때는 생각하지 않던 제약이 매일의 결정에 들어온다. 이 제약들은 제도마다 다르고 같은 제도 안에서도 개정되므로 현재 규정 문서에서 확인해야 한다. 수행 단계의 통제층에는 그래서 규정 확인 기록이 별도로 필요하다. 연차 관리의 산출물은 대시보드 한 장이다. 대시보드에는 계획했던 것, 실제로 진척된 것, 계획과 어긋난 부분, 어긋난 이유, 수정 계획, 그리고 다음 보고까지 남은 의무가 들어간다. 계획대로 되지 않은 항목을 적어 두면 다음 해에 그 항목이 어떻게 처리되었는지 이어서 볼 수 있고 종료 시점에 전체 이야기가 복원된다. 어긋남을 지우고 성과만 남기면 대시보드는 보고서의 초안이 될 뿐 운영 문서가 되지 못한다.

여기서 제출 시점에 만든 Aim과 근거의 표가 다시 일을 한다. 연차 보고서에서 설명해야 하는 것은 대체로 계획 대비 진척인데, 진척을 설명하려면 애초에 그 계획이 무엇에 근거했는지를 말할 수 있어야 한다. 표가 남아 있으면 각 Aim의 전제가 여전히 유효한지, 그 사이에 나온 새 연구가 전제를 흔들었는지, 예비 결과가 없다고 표시했던 칸이 채워졌는지를 그대로 확인할 수 있다. 표가 없으면 제안서 문장을 다시 읽으며 근거를 역추적해야 하고 그 작업은 제출 당시의 판단을 재현하지 못한다. 제출 시점에 몇 시간을 들여 표를 만들어 두면 몇 해 뒤에 며칠을 아낀다. 과학적 산출과 행정적 준수는 서로 연결되어 있지만 한 문단에 섞어 쓰지 않는다. 논문이 늦어진 이유와 예산 집행이 늦어진 이유는 성격이 다르고 해결 방법도 다르며 확인해야 할 상대도 다르다. 다만 인력 변경처럼 두 track에 동시에 걸리는 사건은 양쪽 모두에 기록하고 서로를 가리키게 한다. 예산과 인사에 관한 최종 판단은 규정과 기관의 승인에 달린 일이므로 에이전트는 누락과 모순을 찾는 데까지만 쓰고 결정은 사람이 한다.

종료할 때 다음 과제로 넘기는 것

종료 시점에는 성과 목록과 함께 최종 산출물이 어디에 있는지, 데이터와 코드가 어느 저장소에 있고 누가 접근 권한을 가지고 있는지, 어떤 논문이 이 과제의 지원을 받았는지, 아직 끝나지 않은 의무가 남아 있는지, 나중에 감사나 확인 요청이 왔을 때 필요한 자료가 무엇인지를 적는다. 종료 직후에는 모든 것이 기억나므로 이 정리가 불필요해 보이는데 기억은 반년이면 흐려지고 사람은 이동한다. 종료 정리는 몇 해 뒤에 이 폴더를 여는 사람을 위해 쓴다. 그 사람이 자신일 확률도 낮지 않다. 종료된 과제를 진행 중인 자리에 계속 두면 현재 상태를 보는 눈이 흐려진다. 그렇다고 실제 종료 처리가 끝나기 전에 옮기면 남은 의무를 놓친다. 이동은 종료가 행정적으로 확인된 뒤에, 그리고 남은 의무가 없다는 것을 확인한 뒤에 한다. 다섯 단계 사이의 이동은 되돌리기 어렵고 걸어 둔 링크를 끊을 수 있으므로 에이전트에게 자동으로 맡기지 않는다. 무엇을 옮겨야 하는지 목록을 만들게 하고 옮기는 조작은 사람이 확인한 뒤에 수행한다.

다음 과제로 넘길 지식은 판단이다. 선정된 제안서의 표현을 복사해 다음 제안서에 붙이는 방식은 흔하지만 대체로 효과가 없다. 통했던 것은 그 표현이 지탱한 논리 구조였고 다음 공고는 다른 목적과 다른 심사 기준을 가진다. 남겨야 하는 것은 어떤 Aim이 실제로 수행 가능했고 어떤 가정의 틀렸는지, 어떤 일정이 비현실적이었고 어디서 시간이 예상보다 오래 걸렸는지다. 틀린 가정의 기록이 다음 과제에서 가장 값이 나가는 자산이다. 과제 폴더에서 얻은 것 가운데 문헌 위기로 되돌릴 것은 골라서 옮긴다. 수행 과정에서 정리한 방법론, 검증된 개념 설명, 출판된 결과는 다른 작업에서도 쓰이므로 위키의 종합 문서에 반영한다. 반면 예산 내역, 참여자 정보, 심사 관련 기록, 미공개 데이터는 과제를 소유한 자리에 남기고 문헌 위기로 복사하지 않는다. 문헌 위키는 출판된 지식의 층이라서, 준비 중이거나 심사 중인 원고는 제목적차 들어가면 안 되고 진행 중인 원고 메모는 아젠다 자리에 남는다. 복사가 한 번 일어나면 어느 쪽이 정본인지 흐려지고

개인정보와 비공개 정보가 검색되는 자리로 새어 나간다. 무엇을 되돌릴지 판단하는 기준은 출판되었거나 재사용 가능한 지식인가 하나다.

과제별 통제층이 위키가 되는 방식

과제 lifecycle을 소유하는 작업 공간은 여러 과제를 담고 다른 업무 공간과 같은 골격을 쓴다. 입구 문서는 강제 규칙과 라우팅 표만 담은 얇은 라우터이고 위키의 index가 지도와 현재 규칙 표를, 위키의 로그가 구조와 규칙의 변경 이력을, 영역별 문서가 실제 절차와 실패 조건과 검증을 담는다. 입구 문서를 읽고 위키 index로 지도를 잡고 로그에서 최근에 무엇이 바뀌었는지 확인한 다음, 해당 영역 문서를 연다. 규칙의 출처는 언제나 현재 영역 문서이고 로그는 이력과 최근 결정을 복원할 뿐 현재 문서를 덮어쓰지 못한다. 입구에 각 과제의 세부를 적기 시작하면 과제가 셋만 넘어도 입구가 가장 오래된 정보를 담은 문서가 된다. 세부는 각 과제의 index가 소유하고 입구는 어느 과제가 지금 살아 있는지만 알려 준다. 수행 중인 큰 과제는 그 풀더 안에 자기 입구 문서와 자기 위키를 따로 갖기도 한다. 상위 규칙에는 하위에 자체 규칙이 있으면 그 안에서 하위가 우선한다고 못 박아 두는데 그렇게 하지 않으면 같은 항목에 두 층의 규칙이 동시에 살아 있게 된다.

공통 규칙과 과제별 상태를 나누는 것이 이 구조의 핵심이다. 생애주기 단계의 이름과 전이 조건, version과 파일 이름 규칙, 원본 보존 규칙, 발표 자료와 외부 자료를 연결하는 방식은 모든 과제에 같이 적용되므로 공통 규칙 문서가 소유한다. 반면 이 과제의 다음 마감이 언제이고 지금 무엇이 막혀 있는지는 과제별 문서가 소유한다. 공통 규칙을 과제마다 복사해 두면 규칙이 바뀔 때 일부만 고쳐지고 시간이 지나면 과제마다 다른 규칙으로 운영된다. 규칙 문장에는 특정 과제 이름, 날짜, 세션을 넣지 않고 상황과 행동과 검증만 적는다. 한 번 있었던 일을 규칙 문장 안에 박아 두면 다음 사람이 그 사례에 해당하지 않는다는 이유로 규칙 전체를 건너뛰는 건넌다. 다른 업무 공간의 상세 규칙도 이쪽으로 복제하지 않고 실제 의존이나 공유하는 정보가 있을 때만 링크를 만든다. 새 규칙이 필요해지는 시점은 대체로 같은 판단을 세 번째 반복할 때이고 그때 공통 규칙 문서를 고치면 이후의 모든 과제가 함께 따라온다.

세 종류의 과제는 통제층의 모양이 서로 다르다. 심사 단계를 지나는 과제는 현재 기준 파일과 version 계보, Aim 구조, 회의에서 정해진 것, 근거의 경계가 한 index에서 연결되어야 한다. 여러 손이 함께 쓰는 과제는 과학 계획, 일정, 참여자별 소통, 자료 목록, 실행 항목을 각각 다른 문서로 나눠야 동시 진행을 통제할 수 있다. 수행 중인 과제는 기본 정보, 수행 상태, 집행 규칙, 열려 있는 행정 확인, 다음 보고 기한이 중심이고 제안서 시절의 문서 구조와 다르다. 같은 틀을 세 경우에 억지로 맞추면 어느 하나는 반드시 비어 있는 칸으로 가득 찬다. 새 정보가 들어왔을 때 어디에 넣을지는 정보의 종류로 정한다. 기한은 timeline에, 담당과 완료 조건은 action register에, 외부 요청은 communications에, 과학적 변경은 scientific plan에, 파일의 상태는 자료 목록에, 무엇을 왜 바꿨는지는 로그에 넣는다. 과제 index에는 현재 상태와 링크만 유지한다. 과제 index가 길어지기 시작하면 다른 문서들이 갱신되지 않고 있다는 뜻이다. 파트 3이 반복하는 운영 순서는 공고와 메일과 파일에 흩어져 있는 원자료가 들어오고 각 정보가 어느 문서의 소유인지 판정하고 에이전트가 검색과 정리와 누락 탐지를 수행하고 연구책임자가 무엇을 약속하고 무엇을 포기할지 판단하고 제출이나 회신 같은 외부 행동이 일어나고 그 결과와 현재 상태가 소유 문서로 되돌아가는 순환이다. 마지막 단계가 빠지면 다음 순환이 다시 흩어진 원자료에서 시작한다. 순환을 닫는 데 드는 시간은 회신을 보내고 나서 몇 분이면 되지만 그 몇 분을 생략한 대가는 다음 마감 주간에 몰려서 돌아온다.

얇게 유지하는 규칙이 무너지는 조건은 대체로 하나다. 마감이 걸린 주간에 새 정보가 빠르게 들어오면 어느 문서가 그것을 소유하는지 판정하는 몇 초가 가까워지고 판정을 미룬 것이 전부 index로 간다. 그렇게 부풀어 오른 index는 다른 문서가 갱신되지 않고 있다는 표시이므로 길이 자체를 신호로 읽으면 회복 시점을 놓치지 않는다. 상태를 풀더 이름의 접두어로 표현하기로 한 것도 같은 문제에 대한 대응이었다. 작업 중과 신규 추진과 진행 중과 종료와 지난 것 다섯 단계를 풀더 이름에 붙여 두면 그 판정을 문서 안에 적을 필요가 없고 대신 상태를 옮기는 일이 되돌리기 어려워져 옮길 때 사람 확인을 강제한다. 파일 이름에는 문서명과 version만 넣고 무엇을 왜 바꿨는지는 로그가 담게 한 것도 index를 얇게 유지하는 장치다. 이름에 작업 내용을 붙이기 시작하면 파일 이름이 요약이 되고 요약이 된 이름은 아무도 갱신하지 않는다.

규칙 문서 쪽에서 실제로 무너진 사례는 다른 종류였다. 사람 이름과 프로젝트 이름을 규칙 문장 안에 박아 넣기 시작했고 그렇게 하면 규칙이 무한히 자란다는 지적을 받고서야 문제가 보였다. 고친 방향은 규칙 문장을 일반 알고리즘으로 바꾸고 한 번뿐인 결정은 별도의 데이터 파일로 옮기는 것이었다. 규칙에 이름이 박혀 있으면 다음 사람이 자기에게 해당하지 않는 사례라는 이유로 규칙 전체를 건너뛰는데 이 실패는 규칙을 어기는 것과 달리 눈에 띄지 않는다. 지금 규칙 문서에 남긴 형식은 상황과 행동과 검증 셋뿐이고 특정 과제 이름과 날짜와 세션은 넣지 않는다. 새 규칙을 언제 만들지도 정해 두었다. 같은 판단을 세 번째 반복할 때가 그 시점이고 두 번까지는 그 과제의 기록에만 남긴다.

27장. AI 비서와 예약 자동화 만들기

아침에 도착한 보고서에 한 줄이 섞여 있다. 앞 단계가 남겼어야 할 인계 파일이 오늘 자로 만들어지지 않았고 그래서 그 계열의 항목은 미확인으로 비워 두었다는 문장이다. 실제로 그날 상류 작업은 자료를 받아 검증까지 마친 상태였고 인계 파일과 실행 기록만 만들어지지 않았다. 실패를 알리는 메일도 알람도 오지 않았으며 구멍이 벌어졌다는 사실을 알려 준 것은 하류 보고서의 그 한 줄이다. 설계해 놓은 탐지가 작동했고 사람이 그 줄을 읽었기 때문에 발견되었다. 여러 작업은 정해진 시각에 차례로 깨어나 앞 작업이 남긴 파일을 읽고 다음 작업이 읽을 파일을 남긴다. 조율은 시각 순서와 파일 경로 규약으로 이루어지고 작업끼리 협상하거나 서로를 직접 호출하는 일은 없다. 판정과 집행이 다른 작업에 나뉘어 있어서 상류가 판정하고 하류는 그 판정을 다시 해석하지 않는다. 이 배치가 믿을 만한 이유는 각 단계가 앞 단계의 산출물을 검증하고 검증에 실패하면 추정으로 메우지 않고 그 계열을 통째로 멈추기 때문이다. 각 항목이 언제 만들어진 자료에서 나왔는지와 무엇을 기계가 하고 무엇을 사람이 하는지가 문서에 적혀 있지 않으면, 아침에 도착한 목록을 읽고도 무엇을 해야 할지 정할 수 없다.

정해진 시각에 무언가를 돌리는 장치

먼저 무엇으로 시각을 거는지부터 정리해 둔다. 가장 오래된 것이 cron이다. 유닉스 계열 운영체제에 들어 있는 스케줄러이고 분, 시, 일, 월, 요일 다섯 칸으로 주기를 적으면 그 시각마다 정해진 명령을 실행한다. 매일 새벽 두 시에 스크립트 하나를 돌리는 일이 한 줄로 표현된다. 이 스케줄러는 명령을 실행하고 끝나므로 그 명령이 무엇을 했는지는 명령 자신이 남기는 기록에 달려 있다. Apple의 macOS에서는 cron 대신 launchd를 쓰는 것이 표준이고 등록 파일 하나로 주기와 실행할 명령과 출력이 어디로 갈지를 함께 적는다. 컴퓨터가 꺼져 있거나 잠들어 있던 시각을 지나쳤을 때 깨어난 뒤에 한 번 실행할지도 여기서 정한다. 노트북에서 돌릴 때 이 설정이 중요해지는데 매일 새벽 두 시에 걸어 두어도 그 시각에 노트북이 닫혀 있으면 아무 일도 일어나지 않기 때문이다. 리눅스 서버에서는 systemd의 타이머를 쓰는 경우가 많고 하는 일은 같다.

두 번째 층은 AI 도구 자체가 가진 예약 기능이다. 현재 ChatGPT에서는 예약 작업이라고 부르고 활성 작업과 멈춘 작업과 완료된 작업과 최근 실행 결과를 Scheduled 화면에서 관리한다. 웹에서는 Chat이나 ChatGPT Work에서 예약을 만들 수 있고 업로드한 자료와 연결된 도구를 쓸 수 있지만 컴퓨터의 로컬 폴더에는 직접 접근하지 못한다. 데스크톱 앱의 예약 작업은 로컬 프로젝트 폴더나 격리된 작업 사본에서 실행할 수 있다. 로컬 파일이 필요한 예약은 컴퓨터가 꺼져 있고 데스크톱 앱이 실행 중이어야 한다. Codex 명령줄과 편집기 확장에는 Scheduled 관리 화면이 없으므로 웹이나 데스크톱 앱에서 예약을 만들고 관리한다. 운영체제의 cron이 정해진 명령을 실행한다면 이쪽은 자료를 읽고 중요도를 가르는 판단이 섞인 작업을 수행한다. 메일을 훑어 결정이 필요한 것을 고르거나 여러 정보원의 상태를 모아 보고를 만드는 일이 여기에 해당한다. 클라우드에서 도는 예약은 자기 컴퓨터가 꺼져 있어도 실행되지만 로컬 파일에 직접 닿지 못하고 연결된 서비스에 자료 접근 권한을 준다는 대가가 있다. 기능의 제공 범위와 사용량은 요금제와 작업 공간 설정에 따라 바뀌므로 실제로 걸기 전에 현재 공식 문서를 확인한다. 연결된 서비스에 어떤 자료를 열어 줄지도 실행 전에 정해야 한다.

셋 가운데 무엇을 쓸지는 두 질문으로 갈린다. 컴퓨터가 꺼져 있어야 하는가와 그 작업에 판단이 들어가는가다. 검색 색인을 다시 만드는 일처럼 결과가 로컬 파일에만 영향을 주고 판단이 없는 작업은 운영체제 스케줄러가 맞고 이 책이 근거로 삼은 위키의 야간 재색인이 그 방식으로 걸려 있다. 여러 정보원을 읽고 무엇이 중요한지 골라 사람이 읽을 보고를 만드는 일은 도구의 예약 기능이 맡는다.

예약 Agent라는 말이 가리키는 것

두 층위는 한 작업 환경 안에 공존하고 확인하는 방식이 서로 달라야 한다. 정해진 스크립트를 실행하고 끝나는 야간 재색인과 여러 정보원을 읽어 아침 보고를 만드는 작업이 같은 기준으로 판정될 수 없다. 앞의 것은 색인이 실제로 갱신되었지만 보면 되고 뒤의 것은 무엇을 읽었고 무엇을 못 읽었는지까지 봐야 한다. 차이가 가장 크게 드러나는 곳은 실패할 때다. 세션은 정상적으로 시작되었고 보고서도 만들어졌는데 필요한 정보원 하나에 접근하지 못했다는 사실이 한 문장으로만 적혀 있거나 아예 빠져 있을 수 있다. 그래서 운영 규칙의 첫 조항은 종료 코드가 0이라는 것만으로 성공을 판정하지 않는다는 것이다. 오류가 없다는 사실에서 성공을 추론하지 않고 의도한 산출물이 실제로 생겼는지 또는 의도한 상태 전이가 일어났는지를 따로 확인한다. 확인할 대상을 미리 정해 두지 않으면 그 작업은 실패한 날에도 조용히 성공으로 기록된다.

구현이 무엇이든 예약 자동화에는 공통된 성질이 하나 있다. 사람이 지켜보지 않는 시각에 사람의 계정 권한으로 작동한다는 점이다. 대화 중에 실행되는 작업은 결과가 이상하면 그때 멈출 수 있지만 새벽에 도는 작업은 아침에 확인할 때까지 아무도 막지 않는다. 사람이 없는 동안 만들어진 결과가 다음 날의 전제가 되므로 무엇을 만들 수 있고 무엇을 만들면 안 되는지가

실행 전에 정해져 있어야 한다. 비서라는 말은 대화하며 일을 받아 주는 상대를 떠올리게 하는데 실제로 도는 것은 한 방향으로 결과를 내보내는 배달 워커다. 사람이 답장을 보내면 그것을 받아 처리하는 리스너는 코드로 존재하지만 설치되어 있지 않고 운영 규칙을 소유한 문서가 활성화될 금지해 두었다. 지시를 주고받는 창구를 상상하고 이 구조를 만들면 실제로 손에 쥐는 것과 어긋난다. 여기서 얻는 것은 정해진 시각에 정해진 경로의 파일을 읽고 쓰는 작업들의 사슬이다. 조율의 실체도 시각 순서와 파일 경로 규약이다. 앞 작업이 늦게 끝나면 뒤 작업은 기다려 주지 않고 앞 작업이 남겼어야 할 파일의 날짜가 오늘이 아님을 보고 멈춘다. 지능적인 협상이 그곳을 대신할 것이라고 기대하면 실패는 조용해진다. 조율을 시각과 경로에 맡기면 무엇이 어긋났는지가 파일에 적힌 날짜 하나로 드러난다.

모듈로 나누는 반복 업무

하나의 거대한 지시문에 모든 계정과 모든 판단을 넣는 방식이 처음에는 가장 간단해 보인다. 실제로 며칠은 작동하고 무너질 때는 원인을 짚을 수 없는 상태로 무너진다. 한 작업의 프롬프트가 탐색과 전체 읽기와 상태 정합과 가드 갱신과 발행을 전부 맡고 있었던 적이 있는데 상한도 체크포인트도 원자적 발행 경로도 없었다. 그 작업의 인계 파일은 덧붙이기만 하는 이력처럼 자라 며칠 만에 크게 불어났다. 수습은 프롬프트를 대폭 줄이고 검색 횟수와 읽는 건수와 파일 크기에 상한을 넣는 일이었다. 나누는 기준은 세 가지다. 정보원이 다르다면 나누고 그 정보를 보고 내리는 결정이 다르다면 나누고 하나가 실패했을 때 나머지가 계속 돌아야 하면 나눈다. 메일을 선별하는 작업과 논문을 수집하는 작업은 접근 방식도 실패 방식도 다르므로 한쪽이 막혔을 때 다른 쪽이 살아 있어야 한다. 같은 정보원을 같은 결정을 위해 두 번 읽고 있다면 그것은 합칠 대상이다. 작업마다 성공 기준은 하나만 둔다. 논문을 수집하는 작업의 성공 기준은 지정된 정보원 전체를 대상으로 후보와 제외 사유가 함께 기록되었는가다. 성공 기준이 둘 이상이면 하나만 충족해도 성공처럼 보고되고 그 순간부터 그 작업의 보고는 믿을 수 없게 된다. 기준을 한 문장으로 쓸 수 없다면 그 작업이 두 가지 일을 하고 있다는 뜻이고 나누는 일은 대개 그 문장을 쓰다가 막히는 지점에서 시작된다.

1. 읽기 전용 보고 하나를 손으로 일주일 운영하고 그 결과를 계약 문서 한 장으로 남긴다
2. 매일 확인하는 정보원 다섯 개를 적고 각각을 보고 실제로 내리는 결정을 한 문장씩 쓴다
3. 결정을 쓸 수 없는 정보원은 지운다
4. 남은 것 중 두 개를 골라 producer로 정하고 각각의 입력과 인계 자료 항목과 사람이 읽는 출력을 그린다

작업을 쪼갤 때 판정 권한도 함께 정해야 한다. 상류 작업 하나만 판정자로 두고 하류 작업은 그 판정을 다시 해석하지 않고 실행만 한다. 하류가 원자료를 다시 뒤지는 것은 명시적으로 금지되어 있다. 하류가 스스로 판단하기 시작하면 같은 대상에 대해 서로 다른 두 판정이 생기고 어느 쪽이 맞는지 사람이 사후에 가려야 한다. 판정을 넘기는 형식도 달아 둔다. 처분의 종류를 만들기, 삭제 확인, 유지, 보고만, 조치 없음의 다섯 가지로 고정하면 하류가 해석할 여지가 없어진다. 열린 서술로 넘기면 표현이 조금 달라질 때마다 하류의 동작이 조용히 달라지고 그 변화는 오류를 내지 않으므로 며칠 동안 드러나지 않는다. 이름이 붙은 예외는 규칙에 두지 않고 데이터로 옮긴다. 사람과 프로젝트 이름을 규칙 문서에 박았다가 규칙이 무한히 자란다는 지적을 받고 규칙은 일반 알고리즘으로 바꾸고 일회성 결정은 별도의 데이터 파일로 옮겼다. 예외를 데이터가 흡수하지 않으면 몇 달 뒤에는 아무도 그 규칙 문서를 끝까지 읽지 못하고 읽지 못하는 규칙은 지켜지지 않는다.

생산자와 마지막 집행 단계의 분리

↓
↓
↓
↓
↓
↓

이 사슬의 마지막 단계를 종합자로 부르면 실제보다 똑똑한 무언가를 상상하게 된다. 마지막 작업이 하는 일은 상류가 이미 내린 판정을 형식대로 모아 집행하는 것이고 판정을 다시 검토하거나 뒤집는 권한은 거기에 없다. 모든 작업이 그 단계로

모이지도 않는다. 일부만 마지막 단계를 거치고 나머지는 각자 자기 산출물을 낸 뒤 끝나므로 이 구조는 부분 허브와 독립 작업의 혼합이다. 인계는 전부 로컬 파일이다. 공유 데이터베이스도, 메시지 큐도, 작업끼리의 직접 호출도 없다. 갱신할 때는 임시 파일에 다 쓴 다음 원자적으로 이름을 바꾸므로 읽는 쪽이 반쯤 쓰인 파일을 보는 일이 없다. 새 정보원이 생기면 작업 하나를 만들고 인계 파일의 형식을 맞추면 되고 그 작업이 실패해도 나머지 보고는 그대로 나온다. 반대로 마지막 단계에 정보원을 직접 붙이기 시작하면 그 작업은 다시 하나의 거대한 지시문으로 돌아간다. 실행 시각 사이의 간격은 가장 오래 걸렸던 날을 기준으로 잡는다. 여유 없이 붙여 두면 앞 작업이 평소보다 오래 걸린 날에 뒤 작업이 오늘 자 인계를 보지 못하고 멈춘다.

AI 에이전트를 Slack에 연결한 방법

내 AI 에이전트의 이름은 차무희이다. 차무희는 아침 브리핑은 Codex가 관리하는 예약 작업으로 등록되어 있다. 설정 파일에는 작업 종류가 cron으로 적혀 있고 실행 시각은 월요일부터 금요일까지 오전 6시 10분이다. 사용자 crontab에는 별도의 항목이 없다. 예약 시각이 되면 Codex가 차무희 작업을 열고 자동화 운영 규칙과 당일 인계 파일을 읽는다. 유효한 Gmail, 카카오톡과 논문 수집 인계를 모아 브리핑을 만든 뒤 날짜별 outputs/ 폴더에 Markdown 원문을 저장한다. 입력의 최신성, 실행한 변경, 확인 결과와 남은 문제는 같은 날짜의 logs/ 파일에 적는다. 브리핑 파일이 정상적으로 저장된 다음에 Slack 전송을 시작한다. 보고서 생성과 Slack 전송을 따로 판정하므로 Slack이 잠시 막혀도 완성된 브리핑 원문은 로컬에 남는다.

Slack에서는 차무희 이름의 봇을 만들고 # - 채널에 들어오도록 설정했다. 봇이 메시지를 쓰는 데 필요한 권한을 주고, 봇 토큰이나 Incoming Webhook 주소는 자동화 폴더의 비밀 설정 파일에 보관한다. 이 파일은 소유자만 읽을 수 있는 권한으로 두며 토큰과 Webhook 주소를 위키, 프롬프트와 실행 로그에 복사하지 않는다. 차무희 예약 작업은 최종 브리핑 본문을 post_to_chamuhee_slack.py의 표준입력으로 넘긴다. 기본 설정에서는 Slack Bot API 경로를 먼저 고르고, 봇 토큰이 설정되지 않았으면 Incoming Webhook 경로를 사용한다. 목적지는 biseosil처럼 의미를 나타내는 이름으로 지정하며 전송 도구가 그 이름을 실제 채널과 연결한다. 연결된 개인 Slack 계정으로 게시하지 않으므로 받는 사람은 사람이 직접 쓴 메시지와 자동 보고를 구분할 수 있다. Slack 게시 권한은 차무희 봇과 # - 채널로 범위를 제한하고 다른 채널로 보내려면 별도의 승인이 필요하다.

```
06:10 Codex
      ↓
handoff
      ↓
outputs/
      ↓
Slack
      ↓
# -
      ↓
CHAMUHEE_SLACK_DELIVERED      logs/
```

전송 도구가 정상 종료했다는 사실만으로 게시 성공을 기록하지 않는다. Slack이 반환한 응답을 확인한 뒤 CHAMUHEE_SLACK_DELIVERED로 시작하는 영수증을 출력해야 전송이 끝난 것으로 센다. Bot API를 사용한 경우에는 채널 식별자와 Slack 메시지 시각도 영수증에 포함된다. 차무희 작업은 이 값을 당일 실행 로그에 남겨 나중에 어느 브리핑이 실제로 게시되었는지 확인할 수 있게 한다. 영수증이 없거나 전송이 실패하면 로컬 브리핑 생성은 성공으로 남기고 Slack 배달은 실패로 따로 기록한다. 같은 브리핑을 처음부터 다시 만들지 않고 보존된 Markdown 원문으로 전송만 다시 시도할 수 있다. 현재 차무희 아침 브리핑은 예약 작업이 전송 도구를 직접 호출한다. 다른 자동화 가운데 일부는 대기 폴더와 별도의 배달 워커를 쓰므로 어떤 경로가 자기 작업의 운영 규칙인지 먼저 확인해야 한다.

여담: 초반엔 차무희 에이전트가 음성으로 정보를 전달하기도 했다. 지금은 이 기능을 꺼두었다. 종종 정해진 시간이 아니라, 새벽에 작동하기도 하는데, 자다가 들으면 너무 무섭다..

산출물과 handoff와 log의 분리

층	담는 것	갱신 방식
규칙	각 작업이 무엇을 하고 무엇을 하지 않는지	사람이 고친다
상태와 인계	다음 작업이 읽는 값과 만들어진 날짜	원자적 이름 바꾸기로 덮어쓴다
실행 기록	한 번의 실행에서 일어난 일	덧붙이기만 한다
산출물	사람이 그날 판단에 쓰는 보고	날짜별로 새로 만든다

네 층을 한 파일에 섞으면 각각이 못 하게 되는 일이 생긴다. 사람이 읽는 보고서에 실행 시간과 오류 문자열이 섞이면 읽는 사람이 매번 걸러 내야 한다. 다음 작업이 읽는 인계 파일에 사람을 위한 설명 문장이 늘어나면 형식이 조금씩 달라지고 형식이 달라지는 순간 다음 작업은 잘못 읽으면서도 오류를 내지 않는다. 실행 상태를 규칙 문서에 쓰면 그 문서는 매일 바뀌어 무엇이 방침이고 무엇이 흔적인지 알 수 없게 된다. 갱신 방식이 층마다 다른 것도 같은 이유에서다. 인계는 덮어쓰고 실행 기록은 덧붙이기만 하며 산출물은 날짜별로 남긴다. 실행 기록의 형식은 여덟 항목으로 고정되어 있는데 상태, 입력, 상황, 수행한 일, 확인한 것, 산출물, 오류, 남은 항목이다. 항목이 고정되어 있으면 며칠 뒤에 기록을 훑을 때 무엇이 비어 있는지가 눈에 들어오고 확인한 것 칸이 비어 있는 실행을 성공으로 세지 않게 된다. 확인 결과에는 다섯 가지 상태 라벨을 쓰는데 확인됨, 불일치, 미확인, 보류, 재실행필요이며 확인하지 못한 것을 이상 없음으로 적지 않기 위한 장치다. 논문 위키에서는 작업 과정 정보를 페이지 본문과 분리하고, 자동화에서는 규칙, 인계, 실행 기록과 산출물을 나눈다.

같은 작업을 두 번 실행해도 결과가 한 번 실행한 것과 같아야 한다. 예약 작업은 재시도와 수동 실행과 시각 변경 때문에 하루에 두 번 도는 일이 드물지 않다. 중복 방지 키를 무엇으로 잡을지가 실제 설계 문제인데, 매일이면 스레드 식별자와 날짜, 문서면 경로와 수정 시각처럼 대상 자체에서 나오는 값을 쓴다. 실행 시각을 키로 쓰면 다시 돌릴 때마다 새 항목이 생겨 중복 방지가 작동하지 않는다. 중복이 생기지 않는지는 실제로 두 번 돌려 봐야 알 수 있고 한 번 돌려 보고 넘어가면 문제는 나중에 사람이 같은 매일에 두 번 답장하는 형태로 드러난다. 네 층은 보존 기간도 다르다. 사람이 읽은 산출물은 날짜별로 남기는데 나중에 어떤 판단을 왜 그렇게 했는지 되짚을 때 필요한 것이 그날 보였던 상태이기 때문이다. 인계는 최신 것과 직전 것 정도면 충분하고 실행 기록은 실패 추세를 볼 수 있을 만큼의 기간을 두고 정리한다. 상태 표시는 언제 지워도 되는 값이며 지웠을 때 문제가 생긴다면 그것은 결과를 그곳에 두고 있었다는 뜻이다.

최신성을 적지 않은 handoff의 위험

인계 파일에는 언제 만들어졌는지가 들어가고 소비하는 쪽은 그 날짜가 오늘일 때만 내용을 신뢰한다. 생성 날짜가 없는 인계 파일은 읽는 쪽에서 현재 상태로 취급된다. 사흘 전에 마지막으로 성공한 목록이 오늘 아침 보고서에 그대로 실리면, 읽는 사람은 사흘 동안 새로 들어온 것이 없었다고 이해한다. 실제로는 사흘 동안 그 작업이 실패하고 있었을 뿐인데 침묵이 정상으로 읽힌 것이다. 오래된 인계를 만났을 때 무엇을 하는가가 이 구조에서 가장 중요한 설계다. 원본을 다시 뒤져 빠진 것을 복구하는 길은 열어 두지 않았다. 인계가 없거나 오래되었거나 완결 표시가 없으면 그 계열의 처리를 전부 멈추고 실패했다는 사실만 보고에 올린다. 멈추는 쪽을 기본값으로 둔 이유는 복구한 결과가 정상 산출물과 겹으로 구분되지 않기 때문이다. 상류가 실패한 날에 하류가 원본을 뒤져 그럴듯한 목록을 만들어 내면, 그 목록은 정상적으로 도는 날의 목록과 같은 모양으로 보고서에 실린다.

제가 운영하는 차무회 AI 비서는 이 검사를 평일 아침마다 실행 기록에 남긴다. 차무회는 오전 6시 10분에 깨어나 Gmail 선별, 카카오톡 선별, 논문 수집 작업이 남긴 인계 파일을 읽는다. 각 파일의 실행 날짜와 생성 시각, 검증 결과, 오류 수를 확인한 뒤 유효한 항목만 아침 브리핑에 넣는다. 사람이 읽는 브리핑은 날짜별 outputs/ 폴더에 저장하고, 어떤 입력을 읽었고 무엇을 실행했으며 무엇을 건너뛰었는지는 날짜별 logs/ 파일에 덧붙인다. 2026년 8월 4일 실행에서는 Gmail 인계가 당일 자료였고 78개 판정이 검증을 통과했으며 카카오톡 인계도 최신 상태였다. 논문 수집 인계의 실행일은 2026년 8월 3일이어서 당일 자료로 사용할 수 없었다. 차무회는 논문 수집 원본을 다시 뒤지거나 신규 논문 수를 추정하지 않았다. 아침 브리핑에는 ingest 라고 적었고, 실행 로그에는 오래된 인계의 경로와 날짜, 별도로 확인할 작업을 남겼다. 같은 로그에는 Calendar에서 실제로 바뀐 항목, 중복 때문에 건너뛴 항목, 변경 뒤 다시 읽어 확인한 결과와 Slack 배달 영수증까지 기록되어 있어서 브리핑의 한 문장이 어떤 입력과 실행에서 나왔는지 되짚을 수 있다.

제가 카파시 방식의 위키라고 부르는 Markdown 구조가 여기서 실제 운영 장치가 된다. AGENTS.md에는 처음 읽어야 할 문서와 금지 사항을 짧게 적고, wiki/index.md에는 현재 작업과 규칙 문서로 가는 지도를 둔다. 작업별 규칙 문서는 차무회가 읽을 정보원, 허용된 행동, 성공 판정 기준을 관리한다. wiki/log.md에는 규칙과 폴더 구조가 바뀐 이유를 날짜순으로 남긴다. 매일의 실행은 logs/에 덧붙이고 다음 작업이 읽을 현재 상태는 runtime/의 인계 파일로 관리한다.

새 세션은 대화 기록을 몰라도 이 파일들을 순서대로 읽으면 차무희가 무엇을 하기로 되어 있고 직전 실행에서 어디까지 끝났는지 복원한다. 과거 실행 로그는 당시 무엇이 일어났는지를 보여 주며 오늘의 상태는 당일 인계와 실제 서비스에서 다시 확인한다. 로그를 현재 상태처럼 사용하지 않는 규칙과 인계 파일에 생성 시각을 적는 규칙이 함께 있어야 사흘 전의 성공이 오늘의 성공으로 둔갑하지 않는다. 에이전트가 세션마다 기억을 잃어도 운영이 이어지는 이유는 판단과 실패가 채팅창을 벗어나 사람이 읽을 수 있는 Markdown 기록으로 남기 때문이다.

각 인계 자료에는 생성 시각, 대상 기간, 성공과 실패 상태, 읽지 못한 입력과 항목 식별자를 적는다. 성공 기준은 계약 문서만 읽은 새 세션의 에이전트가 오늘 인계 자료가 없거나 허용된 나이를 넘겼을 때 내용을 지어내지 않고 미확인으로 보고하는가이고 그것이 되지 않으면 최신성이 문서에 충분히 적히지 않은 것이다. 상대적인 날짜 표현은 실행 시점에서 절대 날짜로 바꾼다. 오늘, 내일, 이번 주 같은 말은 그 문서를 만든 시각과 표준 시간대를 알아야 해석되는데 예약 작업은 사람이 자는 동안 돌고 사람은 아침에 읽는다. 여러 시간대에 걸친 공동연구가 있으면 어긋남은 더 커진다. 시간대 문제는 문서 안에서만 생기지 않는다. 예약 시각을 담는 반복 규칙이 시간대를 함께 저장하지 않는 형식이면 같은 값이 계절이 바뀐 뒤에 다르게 해석되고 작업은 사람이 기대한 시각과 다른 때에 깨어난다. 한 번 고쳐도 그 형식이 유지되지 않아 같은 실패가 다시 일어난 기록이 있다. 전수를 요청받은 작업을 표본으로 끝내고 전수처럼 보고하는 것도 뿌리가 같다. 위키 운영 규칙에 전수 요청은 표본으로 끝내지 않는다는 조항이 있고 그 조항은 검수 대상 전체를 확인했다고 말했으면서 실제로는 일부만 눈으로 본 실패 뒤에 생겼다. 그래서 보고에는 분모와 분자를 함께 적는다. 대상이 몇 건이었고 그중 몇 건을 처리했으며 나머지는 왜 처리하지 못했는지가 없으면, 그 보고서는 완료처럼 보이는 문서다.

표로 정하는 자동화 권한의 경계

무엇이 막는가	해당하는 동작
코드로 만든 가드	일정 변경
하네스가 강제하는 승인	이메일 발송
작업 프롬프트의 금지 목록	자동 댓글, 메일과 파일과 문서의 삭제, 예약 정의의 생성과 변경과 중단
프롬프트에 데이터로 박아 둔 상시 승인	누가 언제 무엇을 어디까지 승인했는지
막는 것이 없음	읽기와 검색, 초안 작성, 인계 파일 생성

권한 표를 그리는 일과 표대로 막히게 만드는 일은 서로 다른 일이다. 하네스 층의 설정을 열어 보면 승인 게이트가 거의 없다. 기본값이 승인을 묻지 않는 것이고 샌드박스도 전면 접근이며 도구 단위로 사람 승인을 강제하는 항목은 전체 설정에서 손에 꼽는다. 기계적으로 강제되는 승인은 이메일 발송 하나이고 나머지 경계는 지침 문장과 작업별 프롬프트가 지킨다. 각 작업의 프롬프트에는 같은 금지 목록이 반복해서 적혀 있다. 자동으로 메일을 보내지 않고 댓글을 자동으로 달지 않고 메일과 파일과 문서를 지우지 않고 예약 작업의 정의를 만들거나 고치거나 멈추거나 지우지 않는다는 문장들이다. 상시 승인도 한곳에 데이터로 박혀 있어서 무엇을 누가 언제 승인했고 그 범위가 어디까지인지를 프롬프트를 읽으면 알 수 있다. 산문으로 지키는 경계는 지시문이 길어질수록 흐려지므로 어떤 항목이 산문에 있고 어떤 항목이 코드에 있는지를 표에서 갈라 두어야 한다. 초안을 쓰는 권한과 보내는 권한을 분리해야 책임도 나뉜다. 이메일은 예외 없이 초안을 만들고 저장하고 사람에게 내용을 보이고 기다렸다가 명시적 승인을 받은 뒤에 보낸다. 답장해 달라거나 메일을 보내 달라거나 리마인드해 달라는 말은 초안을 만들어 보여주고 기다리라는 뜻으로 읽는다. 초안을 고쳤으면 고친 것을 다시 보여주고 새 승인을 받으며 앞선 승인은 고친 초안에 적용되지 않는다. 사람이 그 턴에 요청하지 않으면 화면에 앱을 띄우지 않는다는 조항도 한곳에 있는데 배경에서 도는 작업이 사용자의 화면에 팝업이나 초점 이동이나 권한 대화상자를 만들면 그 순간 자동화가 사람의 작업을 가로챈다. 권한은 사고 뒤에 좁아진다. 아침에 허용되던 동작이 같은 날 저녁에 읽기 전용으로 바뀐 기록이 남아 있고 그 방향은 대체로 되돌아가지 않는다. 어떤 행동을 승인 없이 하면 안 되는지를 계약 문서만으로 판정할 수 없다면 권한이 아직 문서에 충분히 적히지 않은 것이다.

1. 승인 없이 해도 되는 행동과 반드시 승인해야 하는 행동을 두 칸 표로 나눈다
2. 애매한 행동은 승인 쪽에 넣는다

Transaction으로 다루는 일정 변경

코드로 강제되는 경계는 일정 변경 하나뿐이고 그것은 사고가 반복된 뒤에 만들어졌다. 이 경계는 아침 보고에 후보로 올라온 중복 일정과 집중 시간을 실제 달력으로 옮길 때 적용된다. 문서를 잘못 고치면 되돌리면 되지만 일정을 잘못

만들면 초대가 나가고 다른 사람의 달력이 바뀐다. 취소해도 상대는 이미 그 시간을 비웠거나 다른 약속을 옮겼을 수 있다. 가드의 핵심은 요청의 단계를 식별하는 키다. 같은 요청이 어느 단계까지 갔는지를 키로 표시하고 완료되었거나 위임되었거나 외부 대기이거나 억제된 상태는 같은 키로 다시 열 수 없다. 사람이 자동으로 만들어진 일정을 지우면 지웠다는 표시가 남고 같은 키는 다시 만들어지지 않으며 표시 없이 이루어진 동작은 거부된다. 삭제 권한은 극도로 좁게 열려 있어서 완료 증거가 있는 확인용 블록과 중복 생성된 항목과 확정된 변경으로 대체된 참석자 없는 개인 일정만 지울 수 있고 실제 회의와 강연과 초대와 반복 일정은 건드리지 않는다. 기록은 미리 남기고 짝으로 달는데 시작만 남고 종료 없으면 이후의 모든 일정 변경이 중단되고 자동으로 되돌리거나 다시 시도하지 않는다. 중단된 상태를 사람이 확인하고 푸는 편이 반쯤 바뀐 달력을 기계가 추측으로 정리하는 것보다 싸다. 일정 변경 한 건은 다음 순서로 진행한다.

1. 바꾸려는 일정과 그 일정이 속한 달력을 특정한다.
2. 무엇을 왜 바꾸려 하는지를 기록에 시작으로 남긴다.
3. 허용된 변경 한 건만 수행한다.
4. 같은 식별자로 다시 조회해 실제 반영 상태를 읽는다.
5. 확인된 결과를 기록하고 종료를 남긴다.

비서가 일부러 하지 않는 일

하지 않을 일의 목록은 할 일의 목록보다 짧아지지 않는다. 오래된 인계를 만났을 때 원본을 뒤져 빠진 것을 메우지 않는 것이 첫 번째다. 하류 작업이 상류의 판정을 다시 해석하지 않는 것이 두 번째이고 하류가 원자료를 다시 뒤지는 것은 명시적으로 금지되어 있다. 보고할 것이 없을 때 정해진 침묵 토큰을 돌려주는 것이 세 번째다. 네 번째는 프롬프트마다 반복해서 적혀 있는 금지 목록이다. 메일을 자동으로 보내지 않고 댓글을 자동으로 달지 않으며 메일과 파일과 문서를 지우지 않고 예약 작업의 정의를 스스로 만들거나 고치거나 멈추거나 지우지 않는다. 이미 처리된 항목을 다시 올리지는 것도 명시해야 하는 규칙인데, 답장한 메일과 달한 할 일과 지난 회의는 사람이 판단을 끝낸 것이므로 다시 올라오면 같은 판단을 다시 하게 만든다. 처리 여부를 판정하려면 항목마다 식별자가 필요하고 식별자가 없으면 이 규칙은 지킬 수 없다. 미팅 자료가 준비되지 않았는데 미리 폴더를 만들어 두는 일도 하지 않는다. 빈 폴더가 생기면 다음 세션의 작업이 그 미팅이 준비되고 있다고 읽고 사람도 어느 것이 실제로 진행 중인 일인지 헷갈리게 된다. 논문을 수집하는 작업이 새로 찾은 것을 진행 중인 과제 하나에 억지로 연결하는 일도 하지 않는다. 관련이 없는 논문을 관련 있는 것처럼 붙이면 그 과제의 문헌 목록이 조금씩 오염되고 나중에 그 목록을 근거로 판단할 수 없게 된다. 제목과 초록만 보고 검색용 기록을 만들어 두는 것도 금지되어 있다. 사람에 관한 판단을 만들지 않는 것과 불확실한 일정 후보를 확정된 일정으로 만들지 않는 것도 여기에 속한다. 확실하지 않은 것을 확실한 것처럼 표현하면 사람의 판단이 그만큼 잘못된 전체 위에서 이루어진다. 모른다고 보고하는 상태를 설계 안에 넣지 않으면 이 사슬은 아는 척하는 쪽으로 기운다. 하지 않을 일의 목록은 각 작업의 정의 문서 안에 함께 적어 두고 새 실패를 겪을 때마다 한 줄씩 늘린다.

나만의 비서를 만드는 순서

1. 일주일 동안 반복해서 확인하는 정보원을 적는다.
2. 각 정보원을 보고 실제로 내리는 결정을 한 문장으로 쓰고 결정을 쓸 수 없는 정보원은 지운다.
3. 결정이 서로 다른 것을 별도 작업으로 나누고 각 작업의 성공 기준을 한 문장으로 정한다.
4. 인계 파일의 형식과 생성 날짜 표시, 그리고 날짜가 오늘이 아닐 때의 동작을 먼저 정한다.
5. 판정을 어느 작업이 갖는지 정하고 처분의 종류를 달한 목록으로 만든다.
6. 외부 상태를 바꾸지 않는 읽기 전용 보고부터 한 달쯤 운영한다.
7. 실행 기록의 항목을 고정하고 확인한 것 칸이 비어 있는 실행을 실패로 센다.
8. 되돌리기 어려운 동작은 코드로 막고 산문으로만 막혀 있는 항목이 무엇인지 목록으로 남긴다.

처음부터 여러 개의 예약 작업을 만들면 대부분 실패한다. 무엇을 자동화할지 정하기 전에 무엇을 반복해서 확인하는지부터 적어야 하고 그 목록은 일주일쯤 실제로 관찰해야 나온다. 기억에 의존해 적으면 매일 봐야 한다고 생각하는 것이 적히는데 실제로 매일 보는 것과 그렇게 생각하는 것은 겹치지 않는 경우가 많다. 두 번째 단계가 가장 자주 생략되고 가장 중요하다. 어떤 정보원을 보고 아무 결정도 내리지 않고 있다면 그 정보원은 확인을 그만둘 대상이고 그것을 자동화하면 읽지 않는 보고서가 하나 늘어날 뿐이다. 네 번째와 다섯 번째를 앞쪽에 둔 이유는 두 항목이 나중에 붙이기 가장 어려운 것들이기 때문이다. 인계 형식과 판정 권한을 정하지 않은 채 작업 수를 늘리면 며칠 뒤에 어느 작업이 어느 판정을 내렸는지 되짚을 수 없다. 읽기 전용으로 한 달쯤 돌려 보면 처음 설계한 규칙 중 절반 정도는 고치게 되고 그 고침이 끝나기 전에 발송이나 일정 변경 권한을 붙이면 검증되지 않은 규칙이 되돌릴 수 없는 행동으로 나간다.

권한을 여는 순서에는 기준이 하나 있다. 결과를 사람이 확인하기 전 상태로 되돌릴 수 있는가다. 읽기와 수집과 정리는

처음부터 열어 두었다. 잘못 모았으면 다시 모으면 되고 잘못 정리한 문서는 고치면 된다. 밖으로 나가는 동작과 다른 사람의 일정에 영향을 주는 동작은 열지 않았다. 메일 발송이 여기 해당하고 남의 달력에 무언가를 넣거나 옮기는 일도 한곳에 있으며 두 동작 모두 되돌리는 비용을 상대가 함께 치른다. 사람의 화면을 건드리는 동작도 요청받은 턴에만 하도록 막아 두었는데 예약 작업이 새벽에 돌면서 팝업이나 권한 대화상자를 띄우면 그것을 누르는 사람은 무엇에 승인하는지 모르는 상태로 누르게 된다. 자동화가 열 수 있는 범위는 결정에 필요한 재료를 모아 두는 데까지이고 그 재료로 무엇을 할지는 사람이 그날 아침에 정한다.

실패가 만든 운영 원칙

첫 번째 문제는 논문 수집 작업에서 생겼다. 자료를 모으는 작업은 원문 확인까지 마쳤지만 다음 작업이 읽을 인계 파일을 만들지 못했다. 뒤이어 실행된 아침 보고 작업은 인계 파일의 날짜를 검사한 뒤 보고서에 이라고 적고 해당 항목을 비웠다. 이 한 줄을 보고서에서 읽은 뒤에야 앞 작업이 중간에 멈췄다는 사실을 알았다. 전날 파일을 그대로 사용했다면 보고서는 정상적으로 완성됐을 것이고, 사람은 그날 새 논문이 없었다고 오해했을 것이다.

이 문제를 겪은 뒤 인계 파일에는 생성 시각과 완료 상태를 반드시 적게 했다. 다음 작업은 두 값을 확인한 뒤에만 내용을 사용한다. 인계 파일의 날짜가 오늘과 다르거나 완료 표시가 없으면 해당 계열을 미확인으로 보고하고 처리를 멈춘다. 앞 작업의 완료 여부는 다음 작업이 실제로 읽을 수 있는 인계 파일로 판정하며, 실행 기록의 이나 표시는 참고로만 쓴다.

두 번째 문제는 외부 발송에서 생겼다. 보고서를 만드는 작업이 파일 작성과 발송을 함께 맡고 있었는데, 그 작업의 발송 설정이 비어 있었다. 보고서 파일은 매일 만들어졌고 실행 기록에도 오류가 없었지만 실제 발송은 며칠 동안 이루어지지 않았다. 파일 생성 성공을 발송 성공으로 잘못 계산한 것이다.

그 뒤로 보고서 작성과 발송을 분리했다. 각 작업은 보낼 내용을 대기 폴더에 저장하고, 외부 발송은 전용 작업 하나만 맡는다. 전용 작업은 발송에 성공하면 영수증 파일을 남긴다. 원래 작업은 이 영수증을 확인한 뒤에야 해당 항목을 처리 완료로 기록한다. 보고서 파일이 있다는 사실은 작성이 끝났다는 증거이고, 영수증 파일은 발송이 끝났다는 증거다.

두 실패에서 세 가지 규칙을 정했다.

1. 다음 단계가 확인할 수 있는 파일을 성공 판정의 근거로 삼는다.
2. 다음 작업은 인계 파일의 생성 시각과 완료 상태를 확인한 뒤에만 사용한다.
3. 외부 발송처럼 되돌리기 어려운 동작은 전용 작업 하나만 맡고, 성공 영수증을 남긴다.

예약 자동화의 목표는 실패가 정상처럼 보이지 않게 하는 데 있다. 실패 사실이 보고서와 파일에 남아야 사람이 원인을 확인하고 다시 시작할 수 있다.

위기가 소유하는 자동화의 운영 계약

자동화 코드는 어딘가의 저장소에 있지만 문제가 생겼을 때 코드부터 읽는 것은 가장 느린 경로다. 운영 폴더가 네 층으로 갈려 있는 이유가 여기에 있다. 규칙이 한 층에 있고 가변 상태와 인계가 다른 층에 있고 덧붙이기만 하는 실행 기록이 또 다른 층에 있고 사람이 읽는 산출물이 마지막 층에 있다. 조사를 시작할 때는 규칙 층에서 그 작업이 무엇을 하기로 약속했는지를 읽고 그다음 인계 파일이 만들어진 날짜와 마지막 실행 기록을 함께 본다. 산출물은 오늘 날짜인데 인계는 며칠 전 것인 상태가 실제로 자주 발생하고 산출물만 보면 그 상태가 보이지 않는다. 이 배치가 갈라 놓는 것은 자동화가 무엇을 했는가와 자동화가 무엇을 하기로 약속했는가다. 둘을 구분하지 않으면 자연어 보고서만 읽고 원인을 추정하게 되고 추정은 대체로 틀린다. 약속이 문서에 있으면 어긋난 지점이 코드의 문제인지 계약의 문제인지 판정할 수 있고 계약이 틀렸다면 계약을 고치고 코드가 계약을 지키지 않았다면 코드를 고친다.

규칙은 상황과 행동과 검증의 세 요소로 쓴다. 어떤 상황에서 무엇을 하고 무엇을 확인해야 지켰다고 말할 수 있는지가 한 문장 안에 들어가지 않으면, 다음 세션의 에이전트가 같은 결정을 재현하지 못한다. 무엇을 규칙으로 기록할지도 조건이 정해져 있다. 같은 혼선이 두 번 이상 반복되었거나 한 번이라도 비용이 크거나 되돌리기 어려웠거나 사람이 앞으로 이렇게 하라고 교정했거나 다음 에이전트가 같은 결정을 재현해야 하는 경우다. 한 사례에만 해당하는 파일명과 상태, 검증하지 않은 임시 회피책, 특정 도구를 기본값으로 만드는 취향, 검증할 수 없는 감상형 문장은 기록하지 않는다. 규칙은 처음에 문서로 적는다. 반복해서 확인해야 하는 항목은 검사 코드로 만들고, 승인 없이 실행되면 안 되는 동작은 코드가 실행을 중단하도록 막는다. 일정 변경은 현재 코드로 막혀 있지만 나머지 규칙 대부분은 자연어 지침에 의존한다. 어떤 규칙이 문서에만 적혀 있고 어떤 규칙이 검사나 코드로 강제되는지를 구분해 기록해야 한다.

규칙에 무엇을 적는지도 정해져 있다. 특정 사례와 날짜와 세션은 넣지 않고 일반형 문장만 쓰며 그래야 다음 사람이 자기 상황에 그 문장을 대볼 수 있다. 이미 설치된 도구를 다시 설치하는 일이 스무 번 넘게 반복된 뒤에 도구를 먼저 확인하고 설치를 나중에 하라는 한 줄이 붙었고 그 한 줄이 같은 낭비를 끊었다. 자동화의 계약 문서가 다른 작업 공간의 규칙을 복제하지 않는 것도 같은 위생에 속한다. 복제 대신 절대경로로 가리키고 그 계층에는 심볼릭 링크를 만들지 않으며 모든 문서를 실제 파일로 둔다. 모든 폴더에 규칙 문서를 만들지도 않는다. 판단이 자주 필요하지 않은 업무에는 규칙이 없어도 되고 규칙의 양은 그 업무가 얼마나 자주 판단을 요구하는지를 따라간다. 예약 목록 자체도 관리 대상이다. 어떤 작업이 어떤 입력을 받아 어떤 출력을 내고 지금 켜져 있는지를 적은 목록을 유지하고 실제로 등록된 예약과 주기적으로 대조하는데 문서에는 있는데 실제로는 꺼져 있는 작업과 실제로는 도는데 문서에 없는 작업이 가장 흔한 부패다. 앞의 경우 사람은 오지 않을 보고를 기다리고 뒤의 경우 아무도 읽지 않는 결과가 매일 쌓인다. 성숙도에 가까운 지표는 그 목록과 실체가 일치하는지다. 계약 문서에는 한 달에 한 번 무엇을 검토할지도 정해 둔다. 예약 목록과 실제 활성 작업이 일치하는지, 그리고 지난달에 이 자동화 때문에 실제로 바뀐 판단이 있었는지가 검토 대상이다. 뒤쪽이 계속 비어 있다면 그 자동화는 정보를 만들고 있을 뿐 결정을 돕고 있지 않다는 뜻이고 그때는 그 작업을 끄는 편이 낫다.

28장. 브리핑과 채널로 결과를 받는 법

예약 작업의 결과는 사람이 읽는 채널에 도착해야 뜻이 생긴다. 도착하지 않으면 매일 도는 작업 하나가 늘었을 뿐이다. 결과를 보낼 채널과 각 채널에 담을 순서를 정해야 한다. 이 연구실에서는 두 종류의 브리핑이 서로 다른 채널로 온다. 하나는 개인 채널로 오는 아침 브리핑이고 그날 판단할 것을 모은다. 다른 하나는 연구실 공용 채널로 오는 연구 브리핑이고 그날 읽을 것을 건넨다. 두 가지가 담는 것도 읽는 사람도 다르므로 채널을 나누고 형식을 따로 정한다. 채널을 어떻게 나누고 무엇을 내보내지 않을지도 함께 정해 둔다.

아침 briefing의 항목과 순서

1. 오늘 확정된 일정과 시간에 민감한 마감
2. 답장이나 결정이 필요한 메일과 각 건의 현재 상태
3. 아직 닫히지 않은 질문과 상대가 회신을 기다리는 항목
4. 학생과 연구원의 진행 상황과 다음에 요청할 구체적인 것
5. 새로 들어온 논문 후보와 위키 반영 상태
6. 원고와 검토 요청의 변화
7. 일정에서 정리할 후보와 새로 잡을 집중 시간 후보
8. 실패한 정보원, 오래된 인계, 담당이 불분명한 항목

실제로 도착하는 보고는 다음과 같은 모양이다. 사람 이름과 과제 이름은 지웠다.

개인 채널로 도착하는 아침 브리핑

개인 브리핑은 일정, 마감, 회신할 메시지, 학생 진행 상황을 상태와 함께 보여 준다. 정보가 없는 항목도 0건으로 적어 확인 여부를 남긴다.

일정과 마감의 맨 위에 있고 회신이 필요한 메일, 별표 표시한 메일, 메신저, 학생 진행 상황이 차례로 온다. 각 줄이 확인해야 할 것 하나이고 문장이 짧다. 눈여겨볼 것은 상태를 함께 적는 방식이다. 어떤 항목은 확인됐다고 적히고 어떤 항목은 아직 확인되지 않았다고 적히며 새 메시지가 없으면 0건이라고 적는다. 아무것도 없다는 사실을 적지 않으면 그 정보원을 읽었는지 읽지 못했는지 구분되지 않는다. 마지막 절의 스캔 시각도 같은 성격이다. 그 수치가 언제 기준인지 적혀 있어야 지금 상태로 읽지 않는다.

읽는 순서는 시간이 정해진 것에서 시작해 사람이 기다리는 것, 내가 시작해야 하는 것, 시스템의 상태로 내려간다. 되돌릴 수 없는 것과 오늘 안에 반응해야 하는 것이 뒤쪽에 있으면 그날 놓친다. 시스템 상태를 앞에 두면 실패한 정보원 하나 때문에 정작 오늘의 마감을 늦게 본다. 보고할 것이 없는 작업은 침묵하지 않는다. 아무 말이 없는 것과 할 말이 없다고 말하는 것을 구분하려고 보고할 것이 없으면 정확히 정해진 토큰을 돌려주게 해 두었다. 침묵을 계약으로 정의해 두면 빈 상태를 보고 그 작업이 오늘 돌기는 했는지 되물을 필요가 없다. 보고의 길이에는 상한을 둔다. 실행 상태 기록을 보면 결과 카드는 매일 쌓이는데 사람이 대부분 열어보지 않고 검토 대기 상태가 압도적으로 많으며 읽히지 않는 보고서는 실패를 잡아내는 두 경로 가운데 하나를 스스로 닫는 셈이다. 항목 수를 제한하면 우선순위 규칙이 실제로 작동하는지 시험받는다. 스무 개를 다 올릴 수 있으면 순위를 매길 필요가 없고 여덟 개만 올려야 하면 무엇을 빼는지가 드러난다.

같은 사건이 여러 정보원에 나타나면 한 항목으로 합친다. 회의 일정 변경 하나가 메일과 일정과 메신저에 각각 나타나는 것은 흔한 일이고 합치지 않으면 사람이 세 번 판단하게 된다. 합칠 때는 어느 정보원에서 나왔는지를 항목 안에 남겨야 나중에 그 항목이 잘못되었을 때 어디를 고칠지 알 수 있다. 우선순위는 마감까지 남은 시간, 다른 사람이 응답을 기다리고 있는지, 미루었을 때 늘어나는 비용, 그 일에 필요한 집중 시간의 길이를 함께 보고 정한다. 방금 도착한 메일이 사흘 전 도착한 메일보다 급하다는 보장은 없고 오히려 사흘 동안 답하지 않은 쪽이 대개 더 급하다. 집중 시간을 함께 보는 이유는 두 시간이 필요한 일과 몇 분이면 되는 일을 같은 목록에 나란히 두면 짧은 일만 처리되기 때문이다. 사람에 관한 항목은 가장 보수적으로 다룬다. 파트 3이 쓰는 학생 사례는 실제 운영 구조를 일반화하고 사람에 관한 정보는 합성한 것인데, 첫 주저자 원고를 준비하는 단계의 학생과 독립 과제를 준비하는 단계의 연구원은 같은 신호를 보내도 뜻이 다르다. 일주일 동안 산출물이 올라오지 않았다는 관찰은 기록해도 되고 그 기간을 동기나 성격의 문제로 해석한 문장은 자동화가 만들지 않는다. 자동화가 사람에 대해 만들 수 있는 것은 다음에 무엇을 물어볼지에 대한 후보까지다. 기록의 용도를 먼저 정하지 않으면 활동 기록은 비교와 평가로 흘러가기 쉽고 한번 그렇게 쓰이기 시작하면 되돌리기 어렵다. 무엇을 기록하지 않을지와 기록으로 무엇을 하지 않을지를 자동화보다 먼저 정한다.

1. 일주일 동안 손으로 두 보고를 만들고 종합해 읽는다
2. 매일 중복으로 올라온 항목과 빠진 정보원을 표시한다
3. 일주일 뒤 중복 제거 규칙과 우선순위 규칙을 고쳐 계약 문서에 반영한다

반복된 누락은 정보가 모자라서 생긴 것이 아니었다. 답해야 할 메일과 오늘 마감인 것과 학생이 기다리고 있는 화신은 모두 어딘가에 적혀 있었고 빠지는 것은 그날 아침에 어느 도구를 먼저 열었는지에 따라 달라졌다. 메일함부터 열면 일정이 늦게 보이고 일정부터 보면 사흘 된 메일이 하루 더 밀린다. 이 순서를 사람이 매일 다시 정하고 있었다는 것이 문제였고 그래서 자동화는 처리 순서를 고정하게 되었다. 지금 보고가 시간이 정해진 것에서 시작해 사람이 기다리는 것, 내가 시작해야 하는 것, 시스템 상태로 내려가는 순서를 쓰는 이유가 그것이다.

정보를 더 모으는 방향으로 가지 않기로 한 근거는 이미 쌓여 있던 것에서 나왔다. 예약 작업이 만드는 결과 카드는 매일 늘어나는데 실행 상태 기록을 열어 보면 검토 대기 상태가 압도적으로 많았다. 읽히지 않는 산출물은 없는 것과 같고 이 경우에는 그보다 나쁘다. 자동화가 실패했을 때 그것을 알아차리는 경로가 두 가지뿐인데 하나가 사람이 결과 카드를 여는 것이어서 카드가 쌓이기만 하면 탐지 경로 하나를 스스로 닫는 셈이 된다. 실제로 조사하던 날 상류 작업이 자료를 받아 검증까지 마쳐 놓고 인계 파일만 만들지 않은 구멍이 벌어져 있었고 하류 보고서에 적힌 한 줄이 그 문제를 잡아냈다. 설계된 탐지가 작동했지만 사람이 읽어야만 드러나는 구조였다는 뜻이다. 그래서 보고의 항목 수에 상한을 두었다. 스무 개를 다 올릴 수 있으면 우선순위 규칙은 시험받지 않고 여덟 개만 올려야 할 때 무엇을 빼는지에서 그 규칙이 실제로 작동하는지가 드러난다.

학생들과 함께 보는 연구 브리핑

앞 절의 브리핑은 한 사람의 하루를 여는 것이고 개인 채널로 온다. 연구실 전체가 함께 보는 브리핑을 하나 더 두면 성격이 달라진다. 이쪽은 결정을 요구하지 않고 그날 읽을 것을 건넨다. 세 부분으로 되어 있다. 먼저 그날의 기본 개념 하나를 두세 줄로 설명하고 함께 읽을 논문 한 편을 붙인다. 다음으로 전날 새로 나온 논문 열 편 남짓을 저널과 함께 나열하고 각 줄에 그 논문이 무엇을 보였는지를 한 문장으로 적는다. 마지막에 학술 글쓰기에 대한 조언을 한 문단 붙인다. 세 부분의 성격이 서로 다른데, 앞의 것은 배경 지식이고 가운데는 오늘의 재료이며 마지막은 훈련이다.

연구실 공용 채널에 도착하는 연구 브리핑

연구실 공용 브리핑은 기본 개념, 새 논문 목록, 글쓰기 조언을 한 번에 전달한다. 각 논문은 제목과 저널, 핵심 결과 한 문장으로 제시한다.

가운데 부분이 7장에서 만든 후보 목록의 출력이다. 관심 저널과 주제어로 모은 후보를 사람이 읽을 수 있는 길이로 줄여 내보내는 것이고 각 줄이 한 문장인 이유는 그 이상이면 아무도 끝까지 읽지 않기 때문이다. 제목만 나열하면 무엇이 새로운지 알 수 없고 초록을 붙이면 길어져서 넘긴다. 한 문장에 무엇을 보였는지가 들어 있으면 그중 두세 편을 눌러 보게 된다. 저널 이름을 함께 적는 이유는 그것이 판단의 재료가 되기 때문이다. 같은 주제라도 어느 저널에 실렸는지가 그 논문이 무엇을 주장하려 했는지를 알려 준다. 링크는 원문으로 바로 가게 걸어 두고 위키에 이미 들어온 논문이면 그 사실도 함께 표시한다.

다음 날의 브리핑. 기본 개념과 글쓰기 조언이 매일 바뀐다

다음 날에는 개념과 글쓰기 조언이 바뀌고 새 논문 목록도 갱신된다. 같은 형식을 유지해 매일 읽는 부담을 일정하게 만든다.

앞뒤의 두 부분이 이 형식을 학습 장치로 만든다. 기본 개념은 그날의 논문 목록과 무관하게 정해지고 그 분야에서 학생이 알아야 하는 것을 순서대로 하나씩 내보내면 몇 달이면 한 바퀴가 돈다. 두 줄로 설명하고 함께 읽을 논문을 하나 붙이는 형식이 지켜지면 읽는 부담이 매일 갈아진다. 글쓰기 조언도 같은 방식이다. 서론의 첫 문단을 어떻게 좁혀 가는지, 결과의 소단락을 어떤 문장으로 닫는지처럼 원고를 쓰며 실제로 부딪히는 문제를 하나씩 다룬다. 21장에서 하루 30분의 글쓰기 훈련을 권했는데 이 한 문단이 그날 연습할 내용을 정해 준다.

이 브리핑을 자동화할 때 주의할 것이 둘 있다. 첫째는 논문 요약의 정확성이다. 한 문장 요약은 초록만 보고 만들어지므로 틀릴 수 있고 그 문장을 근거로 삼는 사람이 생긴다. 그래서 이 목록은 읽을 것을 고르는 데까지만 쓰고 위키에 들어가는 것은 4장의 절차를 거친 뒤라고 채널 안내에 적어 둔다. 둘째는 개념 설명과 글쓰기 조언이 매일 새로 만들어진다는 점이다. 반복되거나 서로 어긋나는 일이 생기므로 지금까지 내보낸 개념 목록을 위키에 남기고 다음 것을 고를 때 그 목록을 읽게 한다. 남기지 않으면 몇 주면 같은 개념이 다시 나온다.

자동화의 산출물이 도착하는 채널

자동화가 만든 목록은 사람이 읽는 채널에 도착해야 뜻이 생긴다. 실패를 알리는 장치가 없는 구조에서 발견 경로는 하루 보고서의 한 줄과 사람이 결과 카드를 여는 것 두 가지뿐이고 둘 다 사람이 무언가를 열어야 시작된다. 그래서 그 보고가 어디에 도착하는지가 탐지 설계의 마지막 구간이 된다. 산출물은 개인이 혼자 쓰는 workspace의 채널로 들어온다. 혼자 쓰는 이유는 자동화가 만드는 것이 대부분 판단 이전의 재료라는 데 있다. 후보 목록과 실패 표시와 미확인 항목은 사람이 한 번 걸러야 뜻이 정해지고 걸러지기 전 상태를 여럿이 함께 보면 정리되지 않은 것이 공유된다. 연구실 workspace와는 두 조직을 잇는 연결 기능으로 필요한 채널만 연다. 두 workspace를 합치지 않는 이유는 접근권이고 개인 쪽에는 아침 보고와 실패 표시처럼 사람이 걸러야 할 것이 들어오며 연구실 쪽에는 걸러진 결과만 간다. 25장에서 접근권을 따라 문서를 갈라 둔 원리를 채널에 적용한 것이다.

자동화의 산출물을 채널 하나에 몰지 않는다. 나누는 기준은 읽는 사람이 다른가와 읽는 시점이 다른가 두 가지다. 아침에 한 번 훑고 그날의 순서를 정하는 보고와, 도착할 때마다 확인하는 자료 알림과, 학생이 함께 보는 채널은 성격이 갈린다. 앞의 것은 하루에 한 번 열면 되고 가운데 것은 열어 둔 채 훑어보내도 된다. 셋을 한 채널에 몰면 하루에 한 번 읽어야 할 것이 그때그때 들어오는 것들 사이에 묻히고 사람은 스크롤을 올려 어제 것을 찾게 된다. 잘게 나누는 쪽으로 가면 채널마다 하루 분량이 몇 줄로 줄고 몇 줄을 보려고 열어야 할 곳이 여럿이면 결국 아무것도 열지 않는다. 열리지 않는 채널은 실패를 잡아내는 두 경로 가운데 하나를 스스로 닫는 셈이다. 그래서 채널 수는 사람이 하루에 그것을 몇 번 열 마음이 있는지를 기준으로 정한다. 성격이 갈리는 것을 억지로 붙이지 않고 같은 시점에 같은 사람이 읽는 것을 굳이 떼지 않는 선에서 멈춘다.

채널을 여럿으로 나눈 뒤에도 내보내는 지점은 늘리지 않는다. 발송을 담당하는 지점을 하나 두고 모든 작업이 보낼 내용을 그리로 넘기며 대기 폴더와 배달 워커와 영수증이 그 구현이다. 채널이 셋이면 발송 지점도 셋이어야 할 것 같지만 어느 채널로 갈지는 넘기는 쪽이 요청에 적으면 되고 실제 게시는 한 지점이 맡는다. 발송 지점이 하나면 자격 증명도 한곳에만 둔다. 작업마다 자격 증명을 따로 두면 그중 하나가 낡아도 나머지가 정상으로 도는 동안 드러나지 않고 매일 발송이 실패하는데 아무도 모르는 상태가 이어진다. 두 번째 설정 권위가 만들어졌던 사고가 정확히 그 모양이었다. 설정 권위를 하나로 모으는 이유가 그것이다. 발송 경로가 하나면 그 경로가 막혔을 때 모든 채널이 함께 조용해지므로 한 채널만 조용한 상태와 전부 조용한 상태가 구분되고 어디를 봐야 하는지가 정해진다.

보내는 이름도 하나로 모은다. 모든 자동화가 같은 봇 이름으로 보내고 사람의 계정으로는 보내지 않는다. 사람 계정으로 나가면 받는 쪽이 사람이 쓴 것으로 읽는다. 자동화가 만드는 것은 판단 이전의 재료인데, 사람 이름이 붙으면 그 재료에 사람이 검토한 무게가 붙는다. 학생이 받는 자료 알림이 지도교수 계정에서 오면 그것은 골라서 보내 준 자료가 되고 실제로는 검색 결과 목록일 뿐인데도 그렇게 읽힌다. 답장이 어디로 갈지도 갈린다. 사람 계정으로 나간 메시지에는 답장이 오고 봇 이름으로 나간 메시지에는 답장이 오지 않거나 오더라도 사람이 나중에 읽으면 된다는 것이 서로 알려진다. 작업마다 이름을 따로 만들면 받는 쪽에서 그 이름들이 무엇을 뜻하는지 익혀야 하고 작업 구성을 바꿀 때마다 익힌 것이 어긋난다. 어느 작업이 보냈는지는 메시지 안에 적으면 되고 보내는 이름은 기계가 보냈다는 사실 하나만 전달하면 된다.

조작 간 연결은 양쪽 workspace의 관리자가 각각 승인해야 열리므로 연구실 쪽과 채널을 잇기로 했다면 무엇을 왜 읽는지 설명할 수 있어야 한다. 채널을 정할 때 함께 정해 둘 것은 그때 지난 것을 얼마나 거슬러 볼 수 있어야 하는가다. 사흘 만에 알아차린 실패의 시작 시점을 찾으려면 사흘 전 메시지가 그곳에 남아 있어야 한다. 채널을 기록 보관소로 삼지는 않는다. 되짚어야 할 것은 실행 기록 층에 남고 채널에는 사람이 그 기록을 열어 볼 계기가 도착한다. 이 구분을 지키면 보이는 범위가 줄어도 읽는 것이 알림 하나에 그친다.

따라해보기

아래 블록은 AI 도구의 채팅창에 그대로 붙여 넣는 프롬프트다. 저는 먼저 만들고 싶은 결과를 말하고, 그다음에 참고할 방법과 방향을 적는다. 채널 구조를 만들고 싶다는 목적을 먼저 알려 주면 뒤에 이어지는 문장들이 작업 조건으로 읽힌다.

```
workspace .
workspace .
< >, < >, < > .
.
.
< > .
workspace < > .
< > .
```

읽히고 있는지 확인하는 법

브리핑을 만들어 두면 그것이 읽히고 있는지는 따로 확인해야 한다. 매일 도착하는 것은 며칠이면 배경이 되고 배경이 된 뒤에는 열지 않아도 열었다고 느낀다. 확인하는 방법은 결과에서 거슬러 올라가는 것이다. 지난달에 이 브리핑 때문에 실제로 바뀐 결정이 있었는지 세어 본다. 아침 브리핑이라면 그 목록을 보고 순서를 바꾸거나 미루기로 한 일이 있었는지가 기준이고 연구 브리핑이라면 거기서 본 논문을 실제로 위키에 들인 적이 있는지가 기준이다. 한 달 동안 하나도 없다면 그 브리핑은 정보를 만들고 있을 뿐 결정을 돕고 있지 않다. 그때는 브리핑을 끄거나 담는 것을 바꾼다. 형식을 다듬으면 읽기 좋은 보고서가 하나 더 생기고 읽지 않는 상태는 그대로다.

담는 것을 바꿀 때 먼저 볼 것은 길이다. 항목이 늘어나면 읽는 사람은 훑게 되고 훑기 시작하면 그 안에서 무엇이 중요한지가 사라진다. 항목을 늘리는 대신 무엇을 빼도 되는지를 정한다. 며칠 동안 아무 결정도 만들지 않은 항목이 그 후보다. 반대로 사람이 매번 직접 확인하러 가는 정보는 브리핑에 빠진 항목이다. 그 정보를 추가하면 따로 확인하러 갈 필요가 없어진다. 이런 조정은 몇 달에 한 번이면 충분하다. 무엇을 왜 바꿨는지는 운영 계약 문서에 남겨야 몇 달 뒤에 같은 항목이 되풀이되지 않는다.

29장. 행정 서류를 안전하게 처리하는 법

행정 양식에서는 서식이 곧 요구사항이고 채워 넣은 값은 어느 정보에서 왔는지 밝힐 수 있어야 한다. 기관이 배포한 한글 양식은 원본과 서식을 보존한 채 읽고 채우고 서명하고 검증해 제출본으로 관리한다. 서류 한 건이 끝났을 때 남는 것은 접수 기록, 값과 출처의 대응표, 작업 사본, 검증 기록, 제출 package 다섯 가지이고 다섯 가지가 남아야 같은 요청이 반년 뒤에 다시 왔을 때 처음부터 다시 하지 않는다. 메일함에 첨부파일이 하나 들어온다. 기관이 배포한 한글 양식이고 개인 신상을 적는 칸과 최근 연구 실적을 적는 칸이 한 문서 안에 섞여 있으며 서식을 바꾸면 반려된다는 안내가 붙어 있다. 마감은 사흘 뒤이고 비슷한 양식을 반년 전에도 채운 기억이 있지만 그때 무엇을 어디서 가져다 썼는지는 남아 있지 않다. 아래 서류는 앞선 장들과 같은 합성 사례이며 개인 신상 항목과 연구 실적 항목이 한 양식에 섞여 있고 서식을 바꾸면 반려된다는 성질만 실제 운영에서 일반화한 것이다. 실제 양식 이름, 기관 이름, 사업 이름, 개인 항목의 실제 값은 이 책에 실지 않는다. 행정 서류가 다른 문서와 다른 점은 두 가지다. 하나는 서식이 요구사항의 일부라는 것이어서 표의 칸을 합치거나 행을 늘리는 순간 내용과 무관하게 반려된다. 그래서 원본을 그대로 둔 작업 사본에서 지정된 칸만 채운다. 다른 하나는 값의 출처가 문서 안에 남지 않는다는 것이다. 완성된 양식만 보면 모든 값이 똑같이 확정된 사실처럼 보이지만 실제로는 어떤 값은 정보에서 읽었고 어떤 값은 기억에서 나왔으며 어떤 값은 지난번 서류에서 복사되었다. 세 종류를 구분하지 않으면 틀렸을 때 어디를 고쳐야 하는지 알 수 없다.

행정 서류를 소유한 업무

서류를 받았을 때 가장 먼저 결정할 것은 어느 업무가 이 문서를 소유하는지다. 소유가 정해져야 원본을 어디에 둘지, 작성본을 어디에 만들지, 제출한 뒤 무엇에 연결할지가 함께 정해진다. 소유는 지금 열려 있는 폴더나 파일 이름에 들어간 단어 하나로 정하지 않고 이 일이 무엇의 결과이고 무엇을 만들어 내는지로 정한다. 정해진 뒤에는 그 작업 공간이 파일 이름과 배치와 검증과 완료 기준을 소유하고 다른 자리는 근거와 원자료만 공급한다. 서류는 그 서류를 발생시킨 일이 소유한다.

서류가 발생한 일	기본 소유	대표 예시
외부 강의, 세미나, 자문, 평가, 출장	해당 행사 폴더	위촉 수락, 비밀 유지 약속, 지급 관련 서류
과제 제안, 협약, 수행	해당 과제 폴더	참여 협약, 인력 등록, 예산 양식, 연차 보고
정규 수업과 교과 행정	해당 학기와 과목 폴더	강의 개설 신청, 교과과정 양식, 시험 관련 서류
원고 투고	해당 원고 폴더	저자 확인, 이해관계 진술, 투고 점검표
여러 업무에서 반복 쓰는 본인 자료	제한된 개인정보 정보	이력 정보, 서명, 신분 증빙 정보

행정 서류가 한자리에 모이지 않는다는 것이 이 표의 실제 결과다. 학위 행정, 교과 행정, 교원 승급, 연구비 정산은 각각 다른 작업 공간에 살고 그 전부를 담는 상위 보관소는 없다. 서류를 한곳에 모으는 순간 그 서류가 어떤 일의 결과였는지가 지워지고 다음에 그것을 찾을 단서도 함께 사라진다. 서류를 모아 두기만 하는 폴더에는 규칙 문서를 두지 않는 편이 낫다. 판단이 반복되지 않는 자리에 규칙을 만들면 아무도 읽지 않는 문서가 하나 늘고 읽히지 않는 규칙은 실제 관례와 어긋난 채로 남는다. 소유를 잘못 정했을 때의 손실은 다음번에 나타난다. 같은 기관이 반년 뒤에 비슷한 서류를 요청하면 지난 작업을 찾지 못해 처음부터 다시 하고 그 과정에서 같은 서류의 두 번째 판이 다른 폴더에 생긴다. 판이 둘이 되면 어느 쪽이 실제로 제출된 것인지 알 수 없다. 판단이 애매하면 이 서류를 다시 찾을 때 어떤 일을 떠올릴 것인지를 묻고 그 일이 소유한다. 문서에 미팅이나 연구비 같은 단어가 들어 있다고 해서 그 이름의 폴더를 새로 추측해 만들지 않는다. 작성본은 해당 업무의 제출 package 안에 두고 여러 업무가 공유하는 이력 정보나 개인정보 정보를 복사해 새 정보를 만들지 않는다. 정보가 둘이 되는 순간 한쪽만 갱신되고 어느 쪽이 현재 사실인지 알 수 없게 된다. 연구 지식을 담는 위키에는 공개 가능한 연구 내용과 다시 쓸 지식만 연결하고 고유 식별 번호, 계좌 정보, 개인 연락처, 서명 이미지는 넣지 않는다. 위키는 검색되라고 만든 공간이고 검색되면 안 되는 항목은 애초에 들어가면 안 된다.

서류 작업의 라우터인 문서 규칙 묶음

서류마다 판단이 반복되므로 판단의 결론을 모아 두는 자리가 따로 필요하다. 실제 운영에서는 행사 업무를 소유한 작업 공간 아래에 문서 규칙만 모은 묶음을 두고 그 안을 성격별 leaf로 나눈다. Leaf는 하나의 규칙만 소유하는 말단 문서를 뜻한다. 이 묶음이 소유하는 것은 반복 가능한 서류 작성 절차이고 서류 파일 자체는 그 서류를 발생시킨 업무의 폴더에 남는다. 절차의 정보와 서류의 보관 위치를 같은 것으로 읽으면 절차 한 줄을 고치려고 여러 업무 폴더를 뒤지게 되고 반대로 지난 서류를 찾으려고 규칙 문서를 뒤지게 된다. 원본과 작성본과 제출본을 어떻게 나누고 이름을 어떻게 붙이는지, 한글 양식을 다룰 때 무엇을 확인하는지, 개인 항목과 서명의 출처가 어디인지, 신분 증빙을 어떤 조건에서 첨부하는지, 이력 정보를 어떻게 갱신하는지가 각각 다른 leaf에 있다. 지금 서류를 처리할 때 투고 점검표 규칙을 읽을 필요는 없기 때문이다. 어느 작업 공간이든 골격과 진입 순서는 같다. 강제 규칙과 라우팅 표만 담은 얇은 문서가 입구이고 그 위키의 지도가 현재 규칙 표를 들고 있으며 변경 이력 문서가 구조와 규칙이 언제 왜 바뀌었는지를 담고 실제 절차와 실패 조건과 검증은 owner leaf에 있다.

```

↓
↓
, ,
↓
leaf + , , leaf

```

↓
→ → → package

순서를 고정해 두는 이유는 건너뛰기 쉬운 단계가 정해져 있기 때문이다. 급할수록 요청 원문을 다시 읽지 않고 양식부터 열게 되고 그때 놓치는 것이 대개 파일 이름 규칙이나 함께 내야 하는 다른 서류다. 규칙 leaf를 읽는 단계도 자주 생략되는데 지난번에 겪은 실패를 적어 둔 자리가 바로 그 leaf다. 순서를 문서로 적어 두면 에이전트에게 맡길 때도 같은 순서를 그대로 지시할 수 있다. 연구 지식을 담는 위키와 이 묶음 사이에도 역할이 나뉘어 있다. 사람과 연락처, 문헌 근거, 저널 정보, 문체 참고는 연구 위키가 소유하고 폴더 배치와 파일 이름과 판 관리와 조판 확인과 검증 절차는 각 업무 작업 공간이 소유한다. 한쪽의 규칙을 다른 쪽에 옮겨 적지 않고 정본의 경로를 그대로 가리키며 라우팅 표에는 이 위키에 중복하지 않는다는 열이 따로 있다. 서명 이미지, 이력 정보, 본인 증빙 같은 실물 자산은 규칙 문서와 분리해 별도의 물리 폴더에 두고 다른 작업 공간은 그곳을 직접 가리켜 끌어다 쓴다. 이 계층에는 바로가기를 만들지 않고 모든 문서를 실제 파일로 둔다. 같은 기관이 반복해서 요구하는 관행은 기관별 leaf로, 같은 사업이 반복해서 쓰는 양식은 사업별 leaf로 분류하고 한 행사에서만 요구한 항목은 행사 폴더에 남긴다.

양식을 다루는 처리 계층이 맡는 일

한글 양식은 텍스트만 뽑아서는 다룰 수 없다. 병합된 셀, 중첩된 표, 체크박스, 사진칸, 서명 자리, 쪽마다 다른 조판이 문서의 의미를 이루고 있어서 내용을 옮기는 순간 요구사항의 절반이 사라진다. 그래서 이 작업에는 원본 서식을 보존한 채 문서를 읽고 채우고 확인하는 처리 계층이 필요하다. 그 역할을 맡을 수 있는 도구로 kordoc이 있다. 다만 도구의 판이 바뀌면 명령과 옵션과 지원 범위도 함께 바뀌므로 여기서 고정할 것은 어떤 역할이 반드시 있어야 하는가다. 한글 양식을 전달하는 처리 계층이 있으면 그것을 먼저 쓰고 그 계층이 못 하는 동작에만 예전 도구를 부른다. 읽는 쪽의 역할은 둘이다. 문서를 사람이 읽을 수 있는 형태로 옮기되 표의 구조를 무너뜨리지 않는 일, 그리고 빈칸과 그 옆의 label, 체크박스, 서명 자리를 목록으로 뽑아 주는 일이다. 쓰는 쪽의 역할은 셋이다. 원본 서식을 유지한 채 지정한 칸에만 값을 넣는 일, 기존 문장의 일부만 고쳐 반영하는 일, 기관 원본이 없는 내부 문서를 새로 만드는 일이다. 검증하는 쪽의 역할도 셋이어서 파일이 규격에 맞는지 보는 구조 점검, 조판 결과를 눈으로 볼 수 있게 그려 주는 시각 확인, 개인 식별 항목이 남아 있는지 찾아 주는 점검으로 나뉜다.

이 장에서 kordoc은 HWP·HWPX를 포함한 행정 문서를 읽고 채우며 결과를 그려 확인하는 처리 계층의 예다. 원본 서식 보존과 시각 검증이 필요한 이유를 보여 준다.

원본이 이진 형식으로 오면 원본 파일을 그대로 보존하고 열린 형식의 작업 사본을 만들어 거기서 작업한다. 한글 계열 원본을 다른 문서 형식으로 옮겨 편집하지 않는다. 편집기가 다르면 표의 폭과 줄바꿈이 미세하게 달라지고 그 미세한 차이가 반려 사유가 된다. 변환기가 오류를 내지 않았다는 것은 파일이 만들어졌다는 뜻이고 배치가 보존되었다는 보장은 어디에도 없다. 이 제약은 원본이 한글 형식일 때만 적용된다. 원래부터 다른 형식으로 만들어진 산출물은 그 형식 그대로 두고 형식을 통일한다는 이유로 한글로 바꾸지 않는다. 적용 범위를 좁게 못 박아 두지 않으면 모든 문서를 한 형식으로 몰아가는 정리 작업이 시작되고 그 과정에서 멀쩡하던 산출물의 조판이 깨진다. 조판 결과를 그려 보는 역할은 대개 화면을 쓰는 응용 프로그램을 부른다. 사람이 그 터에 요청하지 않았는데 배경에서 도는 작업이 응용 프로그램을 앞으로 끌어내면, 창이 튀어 오르고 초점이 옮겨 가고 권한을 묻는 대화상자가 뜬다. 사람이 다른 일을 하는 중에 이런 일이 벌어지면 검증이 방해로 바뀌므로 화면을 건드리지 않는 경로로 돌리거나 사람이 요청한 시점에만 실행한다. 새 도구를 도입할 때 물을 것은 여덟 역할 중 무엇을 못 하는지다. 무엇으로도 메울 수 없는 역할이 있으면 그 서류는 이 경로로 처리하지 않는다. 실제로 자주 비는 자리는 시각 확인이고 파일이 열린다는 것과 사람이 볼 때 양식이 맞다는 것은 다른 조건이다.

공식 원본 확인이라는 첫 관문

한글 양식 처리에는 게이트가 순서대로 걸려 있고 첫 번째 게이트는 기관 원본을 확인하기 전에 작성을 시작하지 않는 것이다. 요청 원문과 첨부 목록에서 기관이 요구한 정확한 파일 형식, 파일 이름 규칙, 마감을 먼저 확인한다. 소유 업무의 폴더와 최근 내려받은 파일을 함께 뒤져 공식 원본을 찾고 확장자만 보지 않고 파일이 실제로 그 형식인지 확인한다. 확장자와 내용이 어긋난 파일은 흔하고 그 상태로 작업을 시작하면 마지막에야 문제가 드러난다. 원본은 작업 단위 폴더 안의 원본 자리에 그대로 두고 파일 이름과 받은 시각을 기록한다. 요청 원문이 요구사항의 정보이라는 점도 이 단계에서 고정한다. 파일 이름을 어떻게 붙일지, 어떤 형식으로 몇 건을 언제까지 내야 하는지는 모두 요청한 쪽이 정하며 그 내용은 요청 메일이나 공문에만 있다. 담당자가 전화로 알려 준 조건이나 동료가 기억하는 관행을 요구사항으로 삼으면 나중에 근거가 남지 않는다. 요청 원문에서 뽑은 조건은 접수 기록에 그대로 옮겨 적고 원문 자체도 업무 폴더에 함께 남긴다. 마감 뒤에 조건이 바뀌었다는 연락이 오는 경우가 있으므로 어느 시점의 요청을 근거로 작업했는지가 기록되어 있어야 한다.

kordoc

모두 파싱해버리겠다.

npm v4.7.1 license MIT

대한민국에서 둘째가라면 서러울 문서지옥. 거기서 7년 버틴 공무원이 만들었습니다.

HWP 3.x/5.x, HWPX, HWPML, PDF, XLS, XLSX, DOCX, 이미지(PNG/JPG/WebP) — 관공서에서 쏟아지는 모든 문서를 파싱하고, 비교하고, 분석하고, 생성합니다.

English



Figure 19: Kordoc으로 행정 서류를 읽고 작성하는 예시

1. 양식 하나를 골라 값을 채우지 않고 두 개의 문서만 만든다
2. 최근에 받은 양식이나 공개된 빈 양식을 하나 고른다
3. 원본을 원본 자리에 두고 파일 이름과 받은 시각을 적는다
4. 접수 기록을 한 장 만들어 요청 출처, 기관이 요구한 파일 형식과 이름 규칙, 마감, 이 서류를 소유하는 업무, 원본의 위치를 적는다

마감과 작업 시각은 미리 적는 원장에 남긴다. 원장은 시작과 종료를 짝으로 덧붙이기만 하는 방식이라, 시작만 있고 종료가 없는 줄이 그대로 남아 중간에 끊긴 서류를 드러낸다. 여러 건이 동시에 걸린 주간에는 어느 서류를 손대다 말았는지가 기억으로 복원되지 않는다. 가능하다면 원본부터 한 번 조판해 눈으로 본다. 쪽수, 표의 크기, 사진칸의 위치, 동의 문구가 든 상자, 서명 자리를 미리 봐 두면 작업 사본이 원본과 달라졌을 때 무엇이 달라졌는지 바로 안다. 공식 원본을 끝내 찾지 못했다면 비슷한 양식을 새로 만들거나 다른 형식의 대체본을 쓰지 않고 거기서 멈춘다. 기관에 원본을 다시 요청하는 것이 정답이고 자체 제작한 양식은 내용이 아무리 정확해도 반려된다. 이 관문이 막는 사고는 내용이 비슷한 과거 서류를 현재 기관의 원본으로 잘못 쓰는 일이고 과거 서류는 대개 잘 채워져 있어서 더 위험하다.

도구가 오류를 내지 않았다는 것을 내용이 맞다는 뜻으로 읽으면 이런 종류의 사고가 난다. 같은 작업 환경에서 실제로 있었던 일 하나가 이 형태였다. 텍스트 추출기가 가진 글자 수 상한을 쪽수 제한으로 착각한 채 논문을 넣었더니 본문이 Methods 중간에서 잘렸는데 추출은 성공으로 보고되었다. 잘린 뒤쪽에 있던 내용은 그 뒤의 모든 작업에서 없는 것으로 다뤄졌고 그것이 빠졌다는 사실은 결과물이 이상해진 다음에야 드러났다. 서류 작업에서 잘못된 원본을 붙잡는 일도 같은 구조를 갖는다. 양식이 열리고 조판이 되고 칸이 채워지는 동안 어느 단계도 오류를 내지 않으므로 원본이 옛 판이거나 다른 사업의 것이라는 사실은 제출 직전이나 제출한 뒤에 드러난다. 그래서 원본을 확인하는 일을 작업의 일부로 두지 않고 앞에 별도 관문으로 떼어 놓는다. 받은 원본은 손대지 않은 채로 두고 작업 사본을 따로 만들며 무엇을 원본으로 삼았는지를 원장에 먼저 적는 뒤에 첫 칸을 채운다.

읽기와 빈칸 목록

원본을 확보하면 문서를 칸 단위로 읽는다. 사람은 양식을 읽을 때 문맥으로 빈칸의 의미를 채우지만 처리 계층에는 그 문맥이 없으므로 어느 칸에 무엇이 들어가는지를 명시적으로 적어 주어야 한다. 빈칸 목록에는 각 칸의 label, 같은 label이 문서 전체에서 몇 번 나오는지, 체크박스가 몇 개이고 어떤 선택지인지, 서명 자리가 어디에 몇 개인지, 긴 글이 들어가는 칸이 어느 쪽에 있는지를 적는다. 목록 없이 채우기 시작하면 마지막에 어느 칸을 빠뜨렸는지 확인할 방법이 없다. 빈칸 목록은 제출 직전에 한 번 더 쓰인다. 채우기가 끝난 작업 사본을 목록과 나란히 놓고 칸마다 값이 들어갔는지, 들어간 값이 그 칸의 label과 맞는지를 대조한다. 대조 없이 완성본만 훑어보면 사람은 자기가 방금 채운 순서대로 읽기 때문에 빠진 칸을 건너뛴다. 특히 두 번째 쪽 이후의 체크박스와 표 안쪽의 작은 칸이 자주 비어 있고 비어 있어도 문서는 정상으로 보인다. 읽는 과정에서 특히 조심할 것이 반복되는 label이다. 여러 사람의 정보를 나란히 적는 양식에서는 성명이나 소속 같은 label이 여러 번 나오고 그중 하나만 본인 칸이다. 처리 계층이 label만 보고 모든 칸을 같은 값으로 채우면 다른 사람의 칸이 오염되고 서식은 멀쩡해 보이므로 사람이 검토할 때도 놓치기 쉽다. 병합되거나 중첩된 표를 보기 좋게 만들려고 구조를 평평하게 펴면 그 순간 원래 어느 칸이 어디에 속했는지를 잃는다. 문자 인식으로 얻은 값은 원문과 같은 신뢰도로 취급하지 않고 반드시 원본 화면과 대조한다.

1. 빈칸 목록을 만든다
2. 각 칸의 label, 같은 label의 반복 횟수, 체크박스, 서명 자리, 긴 글이 들어가는 칸을 적는다

값의 출처 결정

빈칸 목록이 끝나면 값을 채우기 전에 각 칸의 값이 어디서 오는지를 먼저 정한다. 기억나는 값부터 채우고 나중에 확인하려 들면, 다음에 같은 양식을 받았을 때 같은 탐색을 처음부터 반복하게 되고 값이 틀렸다는 것이 드러났을 때 어디서 온 값인지 되짚을 수 없다. 값과 출처를 나란히 적은 대응표는 작업 중에는 채우기의 근거가 되고 제출 뒤에는 검증의 근거가 된다. 서류 작업에서 축적되는 것은 이 대응 관계다.

항목 종류	값의 정보	확인 원칙
이름, 직급, 소속, 학력	해당 언어의 이력 정보와 개인정보 정보	두 정보를 모두 열어 대조하고 표기가 다르면 멈추고 사람에게 묻는다
주소, 연락처, 법적 이름, 서명	개인정보 정보	필요한 항목만 읽고 대화, 로그, 위키에 다시 적지 않는다

항목 종류	값의 정보	확인 원칙
계좌, 신분 증빙	보호된 개인 파일 정보	요청 출처와 최소 첨부를 확인하고 값 자체를 본문으로 옮기지 않는다
행사명, 역할, 기간, 지급 기준	요청 원문과 공문, 행사 폴더	과거 행사의 비슷한 문구를 현재 사실로 복사하지 않는다
과제명, 기간, 참여 역할	과제 소유 위키와 현재 협약, 공고 원문	제안 단계와 수행 단계의 값을 구분한다
연구 설명과 실적	해당 프로젝트, 원고, 검증된 문헌 근거	공개 가능 범위와 글자 수에 맞추되 과장하지 않는다

신원 항목은 두 정보를 함께 본다. 이름, 소속, 직급, 학력은 현재 이력 정보와 개인정보 정보에서 각각 확인하고 기억이나 지난번 산출물이나 파일 이름에서 유추한 표기를 쓰지 않는다. 한 출처에 그 항목이 없다는 사실만으로는 확인을 마칠 수 없다. 두 번째 현재 출처를 열어 확인하고 두 출처가 서로 다르게 적고 있거나 표기를 정규화해야 하는 상황이면 거기서 멈추고 사람에게 묻는다. 로마자 표기, 기관 명칭의 개편, 직급의 영문 표기는 그럴듯한 답을 만들기 가장 쉽다. 그럴듯한 답이 한 번 서류에 들어가면 이후의 서류가 그 값을 복사하고 몇 해 뒤에는 기관마다 다른 표기가 등록된 상태가 된다. 대응표를 만드는 비용은 첫 서류에서 가장 크고 그 뒤로 급격히 줄어든다. 세 번째 서류쯤 되면 정보가 없는 항목이 무엇인지가 드러나는데 반복해서 출처를 적을 수 없는 항목은 정보를 새로 만들어야 한다는 신호다. 어떤 항목이 여러 곳에서 조금씩 다르게 나타난다면 정보가 여럿이라는 뜻이므로 하나를 정보원으로 정하고 나머지를 참조로 내린다. 서류를 처리하면서 개인 자료의 정보 구조가 정리되는 것이 이 대응표의 부수 효과다.

1. 각 칸 옆에 값의 출처만 적는다. 값은 적지 않고 어느 정보의 어느 항목까지만 쓴다
2. 실제 개인 항목의 값은 이 과정에서 한 번도 입력하지 않는다
3. 출처를 적을 수 없는 칸에 표시를 남기고 그 칸이 왜 비었는지 한 줄로 적는다

성공 기준은 새 세션을 시작한 에이전트에게 접수 기록과 빈칸 목록만 주었을 때, 지금 채울 수 있는 칸과 채울 수 없는 칸을 구분하고 채울 수 없는 이유를 말할 수 있는가다. 출처를 적을 수 없다고 표시된 칸이 하나도 없다면 대개 목록을 덜 만든 것이다. 대부분의 양식에는 정보가 없는 항목이 최소한 하나는 있고 그 항목을 찾아내는 것이 이 작업의 실제 소득이다. 정보가 없는 항목을 알고 시작하는 것과 채우다가 막히는 것은 같은 서류에서도 걸리는 시간이 크게 다르다. 민감한 항목에는 별도의 경계가 걸린다. 고유 식별 번호, 계좌 정보, 신분 증빙은 광범위 검색의 대상에서 빼고 위키 색인과 작업 로그와 대화 재인용에도 넣지 않는다. 이 파일들이 제자리에 있는지 확인해야 할 때는 내용을 검색하는 대신 파일 권한을 검사한다. 내용을 훑는 확인은 그 자체로 값을 다시 노출하고 검색 기록에 흔적을 남기므로 감사 방법이 감사 대상을 늘리는 셈이 된다. 값을 모은 중간 파일은 제출 package 밖의 제한된 임시 위치에 두고 작업이 끝나면 남기지 않는다. 값을 명령의 인자로 직접 넘기면 그 값이 실행 기록에 남으므로 파일로 넘기거나 값을 가린 채 확인하는 경로를 쓴다. 대화창과 작업 로그에도 값을 적지 않는다. 로그에 남길 것은 어느 정보의 어느 항목을 읽어 어느 칸에 넣었다까지다.

채우기와 고치기와 새로 만들기의 선택

값의 출처가 정해지면 작업 방식을 고른다. 기관 양식에 빈칸이 분명하게 있으면 원본 서식을 유지한 채 지정한 칸에만 값을 넣는다. 원본에 이미 문장이 들어 있고 그중 일부만 바꿔야 하면 문서 구조를 옮기지 않고 해당 문장만 고쳐 반영한다. 기관 원본이 아예 없는 내부 보고서나 회의록은 새로 만드는 쪽이 맞고 이때는 원본 보존이라는 제약이 없으므로 판단이 단순해진다. 작업 방식과 상관없이 지키는 것은 원본 불변이다. 원본 파일은 열어서 보되 고치지 않고 작업은 언제나 사본에서 한다. 원본을 한 번 덮어쓰면 되돌리기 어렵기 때문에 이 규칙을 반드시 지킨다. 되돌릴 때는 동기화 서비스를 남긴 파일 버전 기록을 쓴다. 덮어쓰기 전에 현재 파일을 읽고 무엇을 어떻게 바꿀지 보여 준 뒤 승인을 받는 순서는 여덟 개 작업 공간 전부에 공통으로 적용한다.

제출본의 이름에는 확장자 앞에 본인 제출본임을 나타내는 접미사를 붙인다. 국문 서류와 영문 서류에 각각 정해진 접미사가 있고 그 규칙은 모든 작업 공간에서 같다. 최종이나 진짜최종이나 날짜를 붙인 이름은 이 접미사를 대신하지 못한다. 날짜가 붙은 파일이 세 개면 어느 것이 제출본인지는 여전히 알 수 없고 최종이라는 단어는 사흘이면 두 번 나온다. 접미사가 하는 일은 이 파일이 기관 원본에서 파생된 본인 제출본이라는 사실을 이름만 보고 판정하게 만드는 것이고 그래서 원본 양식에는 어떤 경우에도 이 접미사가 붙지 않는다. 파일이 놓이는 폴더도 같은 원리로 정해져 있다. 받은 원본은 원본 폴더에 두고 지금 제출하는 산출물은 작업 단위 폴더의 루트에 둔다. 최신본을 초안 폴더나 현재 폴더 아래로 내리면 무엇을 제출하는지 확인하는 데 폴더를 세 번 열게 되고 급할 때 그 세 번이 잘못된 파일을 고르게 만든다. 새 판이 나오면 직전 판을 같은 레벨의 지난 판 폴더로 밀어내고 중간 산출물과 미리보기 이미지는 그 아래나 작업 공간 밖의 임시 폴더에 둔다. 작업 폴더

루트에 임시 폴더를 만들면 그 폴더가 다음 주에는 제출 대상처럼 보인다. 반복되는 label을 자동으로 채우지 않게 막는 장치가 이 단계의 안전장치다. 한 label이 여러 칸에 걸릴 때 처리 계층이 자동으로 진행하는 대신 멈추도록 설정해 두면, 다른 사람의 칸을 채우는 사고가 구조로 막힌다. 처리 계층이 멈췄을 때 그것을 빠르게 넘기려고 안전장치를 꺼 버리는 것이 이 단계에서 가장 위험한 선택이다.

별도 단계로서의 서명과 날인

서명은 다른 칸을 채우는 일과 성격이 다르다. 다른 칸의 값은 사실의 전사이지만 서명은 그 문서의 내용에 사람이 동의한다는 표시다. 서명 이미지가 자동으로 배치되었다는 것과 사람이 그 내용에 동의했다는 것은 별개이고 순서도 사람이 전체를 읽은 뒤에 서명이 들어가야 맞다. 서명 이미지는 공식 자산 폴더에서 가져오고 배경이 투명인지 확인한다. 배치 방식에는 정해진 답이 있다. 서명 자리를 나타내는 표시 위에 이미지를 올리고 형식이 그 배치를 지원하면 본문 앞에 뜨는 개체로 둔다. 표시 앞뒤에 이미지를 인라인으로 끼워 넣는 것으로 끝내지 않는데 인라인으로 붙이면 그 줄의 높이가 이미지만큼 늘어나 아래 내용이 통째로 밀리고 쪽 배치가 원본과 달라지기 때문이다. 표시가 문서에 여러 번 나오면 몇 번째 자리인지 지정해서 넣고 중첩된 표나 텍스트 상자 안에 서명 자리가 있을 때 처리 계층이 경고를 내면 배치가 성공했다고 간주하지 않는다.

구조를 훑는 검사만으로 서명된 양식이 완성되었다고 말하지 않는다. 조판 결과를 그려서 서명이 그 영역 위에 놓였는지, 다른 글자를 가리지 않는지, 잘리지 않았는지, 표 배치를 깨지 않았는지를 눈으로 본다. 서명 이미지가 지정된 영역을 조금 벗어나거나 옆 칸의 글자를 덮는 일은 흔하고 파일 구조만 보면 어느 쪽도 오류로 나타나지 않는다. 위치가 어긋났으면 보정한 뒤 다시 그려 보고, 확인할 방법이 없다면 서명을 붙이지 않고 사람이 출력본에 직접 서명하는 경로로 돌린다. 서명을 별도 단계로 떼어 놓는 진짜 이유는 책임의 소재다. 서명한 문서에 사실과 다른 내용이 있으면 그 책임은 서명한 사람에게 있고 문서를 어떤 방식으로 만들었는지는 책임의 크기를 바꾸지 않는다. 에이전트가 채운 값이 정확하다는 것과 그 문서의 진술에 사람이 동의한다는 것은 서로 다른 판정이며 뒤의 판정은 위임할 수 없다. 그래서 서명 직전에 전체를 처음부터 읽는 단계를 절차에 넣어 두고 급할 때도 이 단계는 줄이지 않는다.

서명을 붙이는 동작에는 두 단계가 있고 선은 그 사이에 있다. 앞 단계는 배치이고 뒤 단계는 그 배치를 서명 행위로 인정하는 일이다. 배치는 말긴다. 표시가 있는 자리를 찾아 이미지를 그 위에 올리고 형식이 지원하면 본문 앞에 두어 표시를 덮게 하고 조판 결과를 그려 위치가 맞는지 확인하는 데까지가 여기 들어간다. 인정은 말기지 않는다. 서명된 양식은 기관에 제출되면 사람이 그 내용에 동의했다는 기록이 되고 그 동의를 에이전트가 대신 표시할 자리는 없기 때문이다. 그래서 배치가 끝난 파일은 완성본으로 넘어가지 않고 사람이 열어서 확인할 대상으로 넘어간다. 확인할 방법이 없는 형식이거나 조판 결과를 그려 볼 수 없다면 이미지를 붙이지 않고 사람이 출력본에 직접 서명하는 경로로 돌리는데 확인하지 못한 서명이 붙은 파일은 붙지 않은 파일보다 위험하다.

구조 검증과 시각 검증의 병행

검증은 세 층으로 나눠서 한다. 내용 층에서는 필수 칸이 모두 채워졌는지, 날짜와 금액과 체크박스가 정확한지, 동의 문구가 요구된 그대로인지를 본다. 여기에는 문서에 남으면 안 되는 것을 걷어 내는 확인도 들어간다. 초안을 만드는 과정에서 들어간 지시문, 작업 계획, 도구가 남긴 상태 문구, 조수 어투의 안내 문장이 본문에 섞여 있는 경우가 실제로 있고 서식이 맞으므로 다른 층에서는 걸리지 않는다. 구조 층에서는 파일 묶음과 내부 참조가 규격에 맞는지, 문서가 정상적으로 열리는지를 본다. 배치 층에서는 쪽수가 원본과 같은지, 표의 행과 열이 유지되었는지, 서명이 제자리에 있는지, 글자가 칸을 넘치거나 겹치거나 잘리지 않았는지를 조판 결과로 본다. 구조 오류가 0건이어도 배치는 얼마든지 어긋나 있을 수 있고 기관 담당자가 실제로 보는 것은 조판된 결과다. 시각 확인에서 흔히 생기는 실수는 대표 쪽 한 장만 보는 것이다. 값을 넣은 모든 쪽, 긴 표가 있는 쪽, 서명이 있는 쪽, 그리고 마지막 쪽은 매번 본다. 마지막 쪽을 따로 세는 이유는 내용이 한 줄 늘어났을 때 밀려난 부분이 거기서 드러나기 때문이다.

확인한 결과는 그 업무를 소유한 위키의 로그 문서에 덧붙인다. 작업 폴더 루트에 진행 메모나 변경 기록 파일을 따로 만들지 않는데 폴더 루트에 문서가 늘어나면 무엇이 제출 대상인지가 흐려지기 때문이다. 로그 한 항목의 형식은 고정되어 있다. 무엇이 이 작업을 촉발했는지, 그때 원본이 어떤 상태였는지, 무엇을 실행했는지, 무엇을 확인했는지, 무엇을 발견했는지, 무엇을 결정했는지, 무엇이 남았는지, 어떤 산출물이 나왔는지를 적는다. 확인 결과에는 다섯 가지 상태 라벨을 쓴다. 확인됨, 불일치, 미확인, 보류, 재실행필요다. 라벨을 강제하는 이유는 확인하지 못한 것을 이상 없음으로 적지 않게 만드는 것이고 조판을 그려 보지 못했으면 미확인이며 그 상태로는 제출 게이트를 통과하지 못한다. 값 자체는 적지 않고 칸의 이름과 판정만 적는다. 이 기록이 있으면 같은 양식을 다음에 받았을 때 지난번에 어긋났던 자리부터 확인하게 되고 어긋나는 자리는 양식마다 정해져 있는 편이라 두세 번 처리한 양식에서는 이 기록이 그 양식의 위험 지도가 된다.

개인정보와 제출 package

제출 package는 보내는 파일만 모아 둔 폴더다. 공식 원본, 검증을 통과한 작성본, 기관이 요구한 최소 증빙만 넣고 나머지는 넣지 않는다. 작업용 중간 문서, 값을 모아 둔 파일, 미리보기 이미지, 실패한 변환본은 모두 제외한다. 게이트를 통과하지 못한 파일을 제출 폴더에 남기지 않는 것이 마지막 게이트이고 실패본이 섞여 있으면 언젠가 실수로 그것이 첨부된다. 저장하고 내보내고 첨부하기 전에 삽입한 신원 항목을 하나씩 확인한 출처와 대조한다. 대조하지 못했거나 출처와 어긋난 항목은 사소한 흠으로 보지 않고 진행을 막는 결함으로 다룬다. 파일 이름은 기관이 지정한 규칙이 있으면 그것을 따르고 없으면 언어별 접미사 규칙을 적용한다. 신분 증빙과 금융 정보 사본은 요청한 기관과 담당자와 메일 도메인을 확인한 뒤 요구된 한 종류만 첨부하고 증빙을 첨부해 보내는 일에는 사전 승인이 별도로 걸린다.

발송은 사람이 최종 책임진다. 에이전트는 수신자와 제목과 본문과 첨부 목록을 갖춘 초안까지 만들고 거기서 멈추며 사람이 그 초안을 보고 명시적으로 승인한 뒤에야 발송한다. 메일 보내 달라거나 회신해 달라는 말은 초안을 만들어 보여 주고 기다리라는 뜻으로 읽는다. 초안을 고쳤으면 고친 것을 다시 보여 주고 새 승인을 받으며 앞선 승인은 고쳐진 초안에 적용되지 않는다. 제출 뒤에 남기는 연결이 다음 서류의 출발점이 된다. 발송한 메일과 실제로 보낸 파일을 해당 업무의 폴더에 함께 두면, 나중에 기관이 제출 내용을 물어 왔을 때 무엇을 언제 보냈는지 바로 답할 수 있다. 연구 위키에는 그 서류에서 얻은 재사용 가능한 규칙만 올라가고 파일 자체는 올라가지 않는다. 개인 항목을 어디서 읽었는지는 대응표에 남지만 그 항목의 값은 어느 기록에도 남기지 않는다.

첫 번째 사례인 외부 평가와 자문 서류

외부 기관이 평가 위원을 위촉하면서 서류 여러 건을 한꺼번에 요청한다. 요청 묶음에서 서류 목록과 각각의 마감을 먼저 뽑고 그중 어느 것이 개인 신상을 적는 서류이고 어느 것이 판단을 적는 서류인지 구분한다. 위촉을 수락하는 서류와 비밀 유지를 약속하는 서류와 지급을 위한 서류는 개인 항목을 옮겨 적는 작업이고 평가 의견을 적는 서류는 자료를 읽고 판단하는 작업이다. 원본은 각각 그대로 보존하고 반복되는 개인 항목은 두 정본을 교차 확인해 필요한 항목만 읽어 채운다. 지급 서류에는 가장 민감한 개인 정보가 모인다. 금융 정보와 신분 증빙을 함께 요구하는 경우가 많고 요청하는 쪽도 무엇이 최소한인지 정확히 알려 주지 않는 경우가 있다. 이전에 다른 기관에 보냈던 증빙 묶음을 그대로 재사용하면 이번 기관이 요구하지 않은 정보까지 함께 나가므로 증빙은 서류마다 새로 고른다. 여기서 지킬 경계는 개인 항목 채우기와 내용 판단을 같은 자동화 단계로 합치지 않는 것이다. 개인 항목은 정본과 대조하면 맞았는지 알 수 있지만 평가 의견은 대조할 정본이 없고 읽은 사람의 판단이 근거다. 하나의 흐름으로 합치면 검증 방법이 있는 쪽의 통과가 검증 방법이 없는 쪽의 통과로 오인된다. 서명과 세 층 검증을 마친 뒤 하나의 제출 package와 발송 초안을 만들고 발송 여부는 사람이 결정한다.

두 번째 사례인 과제 수행 중 인력 등록

수행 중인 과제에 인력을 등록하는 서류는 값의 출처가 두 갈래로 갈리는 전형적인 경우다. 한쪽은 등록 대상자의 개인 항목이고 다른 쪽은 과제 자체의 항목이다. 먼저 과제 소유 위키에서 그 과제가 제안 단계인지 수행 단계인지, 현재 참여 역할과 협약 기간이 무엇인지 확인한다. 제안서에 적힌 값과 협약 이후의 값이 다른 경우가 드물지 않고 제안서의 값을 그대로 쓰면 행정에서 되돌아온다. 이 서류의 작성본은 과제 폴더가 소유하고 절차는 문서 규칙 묶음이 소유하는데 서류를 행사 업무 쪽에 만들어 두면 연차 보고나 변경 신청에서 이 등록 건을 다시 찾을 때 아무도 그곳을 떠올리지 못한다. 등록 대상이 본인이 아닌 경우에는 개인 항목의 취급이 한 단계 더 조심스러워진다. 다른 사람의 정보를 대신 채우는 일은 그 사람이 제출한 자료의 범위 안에서만 하고 부족한 항목을 추정해 메우지 않는다. 이전 과제에서 같은 사람의 정보를 다룬 적이 있더라도 그때의 값이 현재도 유효한지는 알 수 없으므로 다시 받는다. 대신 채운 서류는 서명 전에 그 사람이 직접 확인하도록 하고 확인했다는 사실을 기록에 남긴다. 값이 서로 어긋날 때의 처리가 이 서류의 핵심이다. 참여 기간이나 인건비 관련 수치가 과제의 행동 기록과 협약 원문과 행정 요청에서 서로 다르게 나타나는 일이 실제로 생기는데 이때 에이전트가 그럴듯한 하나를 골라 채우면 불일치가 문서 안으로 들어가 사라진다. 불일치는 보고할 사실이므로 세 값을 나란히 적어 사람에게 올리고 어느 것이 맞는지는 담당 부서에 확인한 뒤 채운다.

세 번째 사례인 강의계획서와 교과 행정

수업 관련 서류에서 반복되는 실수는 지난 학기 문서를 현재 학기의 출처로 쓰는 것이다. 먼저 활성 학기와 해당 과목의 소유 폴더를 찾고 지난 학기 강의계획서는 문장 구조를 참고하는 자료로만 쓴다. 현재 학기의 일정, 과목 코드, 평가 비율, 이수 구분은 학사 공지와 학기 정보에서 다시 읽는다. 학기가 바뀌면 조용히 바뀌는 값이 있고 그중 평가 비율과 과목 코드는 틀리면 정정 절차로 이어지므로 비용이 크다. 기관 원본이 있으면 채우거나 부분만 고치고 원본이 없는 내부

계획서에만 새로 만드는 방식을 쓴다. 교과 행정 서류는 값의 양이 많고 대부분이 기계적이라는 점 때문에 자동화 유혹이 가장 큰 영역이기도 하다. 주차별 주제, 교재, 평가 항목처럼 반복되는 칸이 많아 한 번 채워 두면 다음 학기에 그대로 쓰고 싶어지는데 반복되는 칸일수록 조용히 바뀐 값이 섞여 들어가고 대부분 맞기 때문에 검토할 때도 넘어가기 쉽다. 지난 학기 문서를 출발점으로 쓰되 각 칸이 이번 학기 정보와 일치하는지는 칸 단위로 확인한다. 수업 날짜는 서류 안에서 검증되지 않는 값이라 따로 대조한다. 강의 일정 정보와 서류에 적은 주차별 날짜를 나란히 놓고 공휴일과 보강 일정까지 맞춰 본다. 서류만 보면 날짜는 언제나 그럴듯해 보이고 어긋난 것은 학기가 시작한 뒤에야 드러난다. 이 대조는 처리 계층이 대신할 수 없는 종류의 확인인데, 두 값이 서로 다른 정보에 있고 어느 쪽이 맞는지를 사람이 결정해야 하기 때문이다.

수업 업무를 소유한 작업 공간의 배치는 학기 폴더 하나를 작업 단위로 삼는다.

```
- /
  AGENTS.md
  wiki/
    index.md
    layout.md
    conventions.md
    calendar-routing.md
    log.md
  { }{ }-{ }/
  { }/
  textbooks/
  { }/
  , ,
```

폴더 이름은 예시이고 학년도와 학기와 과정을 어떤 순서로 붙일지는 각자의 정렬 방식에 맞춘다. 작업 단위가 학기와 과정의 조합이어서 같은 과목이라도 학기가 다르면 다른 폴더가 되고 학기가 끝나면 지난 학기를 한 폴더로 접는다. 교재와 참고 자료와 반복되는 행정 업무는 학기에 매이지 않으므로 학기 폴더 밖에 따로 있다. 이 위키는 다섯 문서로 끝나 앞선 장들이 보여 준 작업 공간들보다 눈에 띄게 얇다. 규칙의 양은 그 업무가 얼마나 자주 판단을 요구하는지에 따라 정해졌고 서류와 과제를 다루는 공간이 두꺼운 것도 같은 이유에서다. 얇다고 해서 층을 생략하지는 않아서 이 위키도 원자료와 위키와 규칙이라는 같은 세 층을 문서에 선언한다. 원자료는 학기와 과목 폴더에 실제로 놓인 자료이고 위키는 반복해서 다시 쓰는 위치 정보와 배치 기준이며 규칙은 최상위 규칙 파일이 소유한다. 세 층을 선언해 두면 문서를 하나 더 만들지 정할 때 그 내용이 어느 층에 속하는지부터 묻게 되고 어느 층에도 속하지 않는 문서는 만들지 않는다.

따라해보기

아래 블록은 AI 도구의 채팅창에 그대로 붙여 넣는 프롬프트다. 수업 작업 공간을 처음 세울 때 쓰는 프롬프트다. 꺾쇠 안은 자기 경로와 표기로 바꾼다.

```
< > .
- AGENTS.md wiki/
- { }{ }-{ }
- textbooks/

- wiki/ . index.md , layout.md
  , conventions.md ,
  calendar-routing.md , log.md
- layout.md
- AGENTS.md
  1.
  2.
  3.
-
```

처리를 멈춰야 하는 경우

자동화 절차에서 가장 중요한 설계는 어디서 멈추는가다. 다음 상황에서는 진행하지 않고 사람에게 올린다.

- 암호나 보호 설정 때문에 원본을 읽을 수 없다.
- 기관이 배포한 공식 원본을 찾지 못했다.
- 같은 label이 여러 번 나오는데 어느 칸이 누구의 것인지 특정할 수 없다.
- 형식을 바꾼 작업 사본에서 표, 개체, 쪽 배치가 원본과 달라졌다.
- 제출 시스템이 어떤 형식을 받는지 확인되지 않았다.
- 개인 항목의 출처가 서로 충돌하거나 필요한 항목이 현재 정본에 없다.
- 삽입한 신원 항목 중 확인한 출처와 대조하지 못한 것이 남아 있다.
- 서명 위치와 넘침을 조판 결과로 확인할 수 없다.
- 문서 내용이 법적 동의, 예산, 인사, 평가처럼 사람의 최종 판단을 요구한다.

멈췄다는 사실 자체도 기록에 남긴다. 어느 단계에서 왜 멈췄는지, 사람에게 무엇을 물었는지, 그 답이 무엇이었는지를 적어 두면 같은 조건이 다음에 나타났을 때 다시 물을 필요가 없다. 여러 번 반복해서 같은 이유로 멈춘다면 그 조건은 규칙으로 올릴 후보이며 대개 특정 기관이나 특정 사업의 관행에서 온다. 한 번만 나타난 조건은 그 서류의 기록에만 남긴다. 멈춤 기록이 쌓이면 어떤 종류의 서류가 사람의 시간을 실제로 잡아먹는지가 드러나고 그것이 다음 해에 무엇을 준비할지 정하는 근거가 된다. 행정 서류에서 진짜 손실을 만드는 것은 조용히 잘못 채워진 문서가 제출되는 상황이다. 앞의 아홉 항목은 모두 자동으로 그럴듯한 답을 만들 수 있지만 그 답이 맞는지는 문서 안에서 확인할 수 없다는 공통점을 가진다. 확인할 수 없는 자리에서 진행하면 오류가 발견되지 않은 채 기관에 남고 정정에는 작성보다 훨씬 큰 비용이 든다. 멈추는 조건을 미리 문서로 적어 두면 그 판단이 그날의 마감 압박에 좌우되지 않는다.

밀어붙인 쪽과 되돌린 쪽의 경계는 지금 설정 파일에 그대로 남아 있다. 도구를 실행하는 층에는 승인 게이트가 거의 없다. 기본값이 묻지 않는 것이고 샌드박스도 넓게 열려 있으며 도구 하나하나에 사람 승인을 강제해 둔 항목은 전체 설정에서 손에 꼽는다. 읽고 정리하고 초안을 만드는 일에서는 확인을 걸어 봤자 매번 승인만 누르게 되고 그 습관이 붙으면 정작 확인이 필요한 순간에도 승인을 누른다. 그래서 기계적으로 강제한 것은 두 가지로 줄었다. 하나는 이메일 발송이고 다른 하나는 일정 변경이다. 이 둘의 공통점은 밖으로 나가면 되돌릴 수 없다는 것이고 나머지 경계는 지침 문장과 작업별 프롬프트가 지킨다. 각 작업의 프롬프트에는 같은 금지 목록이 반복해서 적혀 있다. 자동으로 메일을 보내지 않고 댓글을 자동으로 달지 않고 메일과 파일과 문서를 지우지 않고 예약 작업의 정의를 만들거나 고치거나 멈추거나 지우지 않는다.

되돌린 자국도 남아 있다. 일정 변경을 코드로 막은 가드는 같은 사고가 반복된 뒤에 만들어졌다. 요청의 단계를 식별하는 키를 두고 완료되었거나 위임되었거나 외부 대기이거나 억제된 상태는 같은 키로 다시 열 수 없게 했다. 사람이 자동 생성된 일정을 지우면 지웠다는 표시가 남아 같은 키가 다시 만들어지지 않는다. 운영 로그에는 아침에 허용되어 있던 동작이 같은 날 저녁에 읽기 전용으로 바뀐 기록도 있다. 권한이 좁아지는 방향으로 움직인 시점은 모두 사고 뒤였고 미리 좁혀 둔 적은 없다. 이 순서를 뒤집을 수 있다고 믿지 않는 편이 낫다는 것이 3년 동안 남은 판단이고 그래서 자동화를 설계할 때 실제로 정하는 것은 잘못이 생겼을 때 그것이 사람에게 도착하는 경로가 몇 개이고 그중 몇 개가 사람이 열어 봐야만 작동하는지다.

30장. 원격으로 에이전트를 관리하기

내가 ChatGPT 앱과 Codex를 사랑하는 가장 큰 이유는 Remote다. 연구실이나 집의 컴퓨터 앞에 앉아 있지 않아도 iPhone에서 같은 프로젝트를 열고 에이전트와 계속 일할 수 있다. 지금도 이동 중에는 ChatGPT 앱의 음성 입력으로 지시하고, 원격으로 연결된 Codex가 집의 컴퓨터에서 파일을 읽고 도구를 사용하도록 한다. 휴대폰에서 새 작업을 시작할 수 있고 이미 진행 중인 작업에 후속 지시를 보낼 수도 있다. 작업이 질문이나 승인 요청에서 멈추면 알림을 확인하고 바로 답한다. 여러 프로젝트와 작업이 동시에 열려 있어도 각각의 대화를 따로 관리할 수 있다. 집의 컴퓨터에 있는 LLM-Wiki는 이 모든 작업이 참고하는 지식의 기반이 된다. 내게 Codex Remote는 이동 중에도 연구와 행정의 흐름을 끊지 않게 해 주는 작업 환경이고, 이 기능 때문에 ChatGPT를 강하게 추천한다.

24시간 돌아가는 컴퓨터 한 대

Codex Remote를 계속 쓰려면 ChatGPT 데스크톱 앱이 실행된 컴퓨터 한 대가 깨어 있고 인터넷에 연결되어 있어야 한다. 나는 집에 둔 MacBook Pro를 전원에 연결한 채 24시간 켜 놓고 호스트로 사용한다. 화면은 꺼 두고 시스템은 잠자기에 들어가지 않도록 설정한다. ChatGPT 앱의 연결 설정에서도 컴퓨터를 깨어 있게 하는 항목을 켜 둔다. MacBook Pro는 저전력 모드로 두어도 문서 검색, 메일 확인, 일정 조회, 위키 편집 같은 작업을 충분히 처리한다. 내 사용 환경에서 이 컴퓨터를 한 달 내내 켜 두었을 때 추가 전기료는 대략 3,000원 정도였다. 무거운 분석을 오래 돌리면 사용 전력은 늘겠지만 평소의 위키와 행정 작업에서는 전력 사용량이 낮았다. 컴퓨터가 잠들거나 네트워크가 끊기거나 ChatGPT 앱이 종료되면 원격 접속도 멈추므로 실제로 가장 자주 확인하는 것은 전원과 잠자기 설정이다.

호스트인 MacBook Pro에는 내가 평소 쓰는 Dropbox 폴더가 동기화되어 있다. LLM-Wiki와 외부 행사 자료를 보관하는 Meetings 폴더도 이 컴퓨터에서 바로 열 수 있다. 원격으로 새 작업을 만들 때 해당 폴더를 프로젝트로 선택하면 Codex가 그 안의 문서와 운영 규칙을 읽는다. “Dropbox의 Meetings 폴더를 열어 줘” 또는 “LLM-Wiki에서 이 교수님과 관련된 기록을 찾아 줘”라고 말하면 Codex가 필요한 폴더를 찾는다. 사용자는 자신이 기억하는 폴더 이름과 자료의 성격만 말해 주면 된다. 검색에 필요한 색인이나 임시 파일은 Codex와 컴퓨터가 관리한다. Gmail이나 Google Calendar 같은 MCP 연결과 필요한 권한도 호스트 컴퓨터의 설정을 그대로 사용한다. 그래서 집의 MacBook Pro 한 대에 작업 환경을 갖춰 두면 iPhone에서는 말로 지시하고 결과를 확인하는 데 집중할 수 있다.

처음 한 번 연결하기

처음 연결할 때는 호스트 컴퓨터의 ChatGPT 데스크톱 앱에서 Set up Remote를 선택한다. 화면에 나온 QR 코드를 iPhone으로 읽으면 ChatGPT 앱이 연결 절차를 연다. 두 기기는 같은 ChatGPT 계정과 같은 작업 공간으로 로그인되어 있어야 한다. 다단계 인증, 통합 로그인, 패스키를 사용하는 계정은 휴대폰에서 해당 인증을 마친다. 연결이 끝나면 ChatGPT 앱의 Remote에 집의 MacBook Pro가 표시된다. 그 컴퓨터에 저장된 프로젝트에서 새 작업을 만들거나 이미 열려 있는 작업을 이어서 사용할 수 있다. 작업 결과, 파일 변경 내용, 테스트 결과, 터미널 출력, 화면 갈무리도 휴대폰에서 확인할 수 있다. 연결된 컴퓨터가 보이지 않으면 데스크톱 앱이 실행 중인지, 컴퓨터가 깨어 있는지, 인터넷이 연결되어 있는지부터 확인한다. 설정 화면의 정확한 이름과 지원 기기는 OpenAI의 Remote 연결 안내에서 확인할 수 있다.

ChatGPT 앱의 Remote 메뉴에서 집에 켜 둔 컴퓨터와 그 컴퓨터의 Codex 작업을 연다.

Remote 화면에는 연결된 컴퓨터의 이름과 그 컴퓨터에서 실행 중인 작업이 함께 나온다. 나는 원고 검토, 메일 확인, 일정 조정, 위키 정리를 각각 별도의 작업으로 열어 둔다. 진행 중인 작업과 내 답을 기다리는 작업을 목록에서 구분할 수 있다. 하나의 작업이 오래 걸리는 동안 다른 프로젝트에서 새 작업을 시작해도 된다. 작업을 누르면 컴퓨터에서 보던 같은 대화가 열리고, 앞선 지시와 결과도 그대로 남아 있다. 이동 중에는 목록을 훑어 질문이나 승인 요청이 온 작업부터 연다. 아래 화면에서는 개인정보와 심사 정보가 들어간 작업 이름을 지웠다. Codex Remote는 여러 작업을 한 화면에서 관리하고 새 작업까지 시작하는 도구로 쓰인다.

연결된 컴퓨터에서 실행 중인 여러 작업을 확인하고 필요한 작업에 바로 후속 지시를 보낸다.

음성으로 하는 원격 작업

나는 iPhone에서 긴 지시를 대부분 음성으로 말한다. ChatGPT Voice를 켜고 “진행 중인 작업을 확인해 줘”라고 말하면 Codex 작업의 상태를 확인하고 필요한 후속 지시를 보낼 수 있다. 새 프로젝트를 열어 별도의 작업을 시작하는 것도 음성으로 요청한다. 몇 분 동안 배경과 원하는 결과를 설명해도 ChatGPT가 내용을 받아 적고 작업 지시로 정리해 주므로 이동 중에 특히 편하다. 에이전트가 중간에 질문하면 그 질문을 음성으로 듣고 바로 답할 수 있다. 내가 기억하는 프로젝트 이름과 자료의 성격을 말하면 호스트 컴퓨터에서 관련 폴더와 파일 후보를 찾아준다. 지금 이 책을 고치는 과정에서도 나는 음성으로 수정할 부분을 설명하고 Codex가 원고 파일을 열어 반영하도록 하고 있다. 음성과 Remote가 결합되면 산책하거나 차로 이동하는 시간에도 여러 작업을 시작하고 방향을 고치고 결과를 확인할 수 있다.

Codex 원격 접속으로 가능한 사례

강의 요청 메일에 답하는 과정은 내가 원격 접속을 자주 쓰는 사례다. 어느 교수님에게 강의 요청 메일을 받으면 iPhone에서 Remote를 열고 Dropbox의 Meetings 폴더를 사용하는 작업을 새로 만든다. 나는 음성으로 교수님 이름과 제안받은 날짜를 말하고 가능한 일정을 확인해 달라고 한다. Codex는 Gmail MCP로 해당 교수님의 원문 메일을 찾아 요청 날짜와 장소와 답변해야 할 내용을 읽는다. Google Calendar MCP로 그 날짜의 강의, 회의, 출장, 이동 전후의 일정을 함께 확인한다. 장소까지 이동하는 데 필요한 시간을 반영해 실제로 비어 있는 날짜를 추천한다. LLM-Wiki와 Meetings 폴더에 같은 교수님이나 행사에 관한 이전 기록이 있으면 그것도 함께 읽는다. 몇 분 뒤에는 가능한 날짜와 충돌하는 일정과 이동 시간을 한 번에 볼 수 있다.

가능한 날짜를 고르면 Codex가 교수님께 보낼 답장을 작성해 Gmail 초안으로 저장한다. 저장한 초안의 받는 사람, 참조, 숨은 참조, 제목, 본문을 휴대폰 화면에 모두 보여 준 뒤 내 승인을 기다린다. 나는 문구와 날짜를 확인하고 필요하면 음성으로 고쳐 달라고 한다. 수정한 초안도 다시 화면에 표시된다. 내용이 맞으면 “이대로 발송하고 가능한 시간은 캘린더에 홀드해 줘”라고 명시적으로 승인한다. Codex는 승인한 메일을 발송하고 Google Calendar에 해당 시간을

타임 블록으로 만든다. 상대방이 날짜를 확정하면 같은 작업에서 확정 메일을 읽고 임시 타임 블록을 최종 일정으로 바꿀 수 있다. 메일 검색, 일정 비교, 답장 작성, 발송 승인, 일정 확보가 하나의 대화 안에서 이어지므로 여러 앱을 번갈아 열며 같은 내용을 다시 입력할 일이 크게 줄었다.

이 과정은 연구 행정에도 그대로 확장된다. 외부 행사에 필요한 서류를 찾고, 이전에 사용한 발표 자료를 확인하고, 새 일정에 맞춰 준비 목록을 갱신할 수 있다. 논문 PDF 배치를 시작하고 중복 후보를 확인한 뒤 ingest를 승인할 수도 있다. 공동연구자의 메일과 프로젝트 기록을 함께 읽어 다음 미팅의 안건을 만들 수 있다. 문서 수정 결과와 변경된 파일도 휴대폰에서 확인한다. Codex Remote는 호스트 컴퓨터의 파일, MCP, 플러그인, 브라우저와 권한을 사용하므로 내가 컴퓨터 앞에서 하던 작업 흐름이 원격에서도 이어진다. 외부로 메일을 보내거나 일정을 만드는 동작은 승인 규칙을 그대로 따른다. 휴대폰은 이 모든 작업을 시작하고 이어 가고 승인하는 인터페이스가 된다.

원격 작업의 지식 기반인 LLM-Wiki

원격 작업이 많아질수록 LLM-Wiki가 더 중요해진다. 나는 논문에서 확인한 내용, 교수와 학생에 관한 기록, 프로젝트별 결정, 메일 작성 규칙을 LLM-Wiki에 남긴다. 새 작업을 열어도 Codex가 해당 프로젝트의 위키와 규칙을 읽으므로 같은 배경을 다시 설명하는 일이 줄어든다. 교수님께 답장을 쓸 때는 이름과 이전 대화와 행사 기록을 확인하고, 논문을 찾을 때는 기존 위키와 PDF를 먼저 검색한다. 작업 중 새로 확인한 사실과 결정은 다시 위키나 작업 로그에 기록한다. 집의 MacBook Pro와 iPhone에서 같은 프로젝트를 열어도 판단의 배경이 유지되는 이유가 여기에 있다. LLM-Wiki에 축적된 지식은 실제 일정과 메일과 연구 작업에 계속 사용된다. Codex Remote는 그 지식을 연구실 밖에서도 꺼내 쓰게 해 준다.

Codex Remote와 Claude Dispatch를 써 본 인상

나는 Codex Remote와 Claude의 Dispatch를 모두 써 봤다. Claude Dispatch에서도 원격으로 작업을 이어 갈 수 있다. 내가 사용한 범위에서는 이미 컴퓨터에서 열어 둔 세션에 들어가는 방식이 중심이어서 이동 중에 여러 새 세션을 만들고 프로젝트를 오가기가 불편했다. 여러 세션의 상태를 한눈에 보고 각각 다른 후속 작업을 거는 과정도 Codex보다 답답했다. Codex Remote에서는 호스트의 프로젝트를 고르고 새 작업을 바로 만들 수 있으며 이미 실행 중인 여러 작업도 같은 목록에서 관리할 수 있다. ChatGPT Voice를 iPhone에서 사용해 새 작업을 시작하고 진행 상태를 묻고 후속 지시를 보내는 흐름도 자연스럽다. 내가 실제 연구와 행정에서 써 본 원격 작업 환경의 성능과 편의성은 Codex가 가장 좋았다. 그래서 누군가 원격으로 LLM-Wiki와 에이전트를 운영할 도구를 묻는다면 나는 ChatGPT와 Codex를 추천한다.

원격 작업의 승인 규칙

Remote로 접속해도 호스트 컴퓨터에 설정된 파일 권한과 승인 규칙은 그대로 적용된다. Codex는 허용된 프로젝트와 연결된 MCP 안에서 작업한다. 새 폴더를 열거나 추가 권한이 필요하면 휴대폰에 승인 요청을 보낸다. Gmail 메일은 초안을 저장하고 받는 사람, 참조, 숨은 참조, 제목, 본문을 보여 준 뒤 발송 승인을 기다린다. Google Calendar에 새 일정을 만들거나 기존 일정을 바꾸는 동작도 무엇이 바뀌는지 확인한 뒤 진행한다. 파일 삭제나 외부 공유처럼 되돌리기 어려운 동작도 같은 방식으로 멈춘다. 이동 중에 승인 요청을 바로 확인할 수 있으므로 안전 규칙을 유지하면서 작업이 몇 시간씩 멈추는 일을 줄일 수 있다. 나는 집에 켜 둔 MacBook Pro와 iPhone의 Codex Remote와 LLM-Wiki를 묶어 사용하면서 연구실 밖에서도 거의 모든 연구와 행정 작업을 같은 방식으로 관리하고 있다.