

AI와의 소통, ‘명령’에서 ‘설계’로

프롬프트 엔지니어링을 넘어, 컨텍스트 엔지니어링의 시대로



“우리는 완벽한 ‘문장’을 만드는 기술을 넘어,
AI가 최상의 결과를 내도록 ‘세계’를 구축하는 법을 이야기해야 합니다.”

패러다임의 전환: 왜 지금 컨텍스트 엔지니어링인가?

**“Prompt Engineering is over.
Context Engineering is in.”**

— Andrej Karpathy (전 Tesla AI 디렉터)

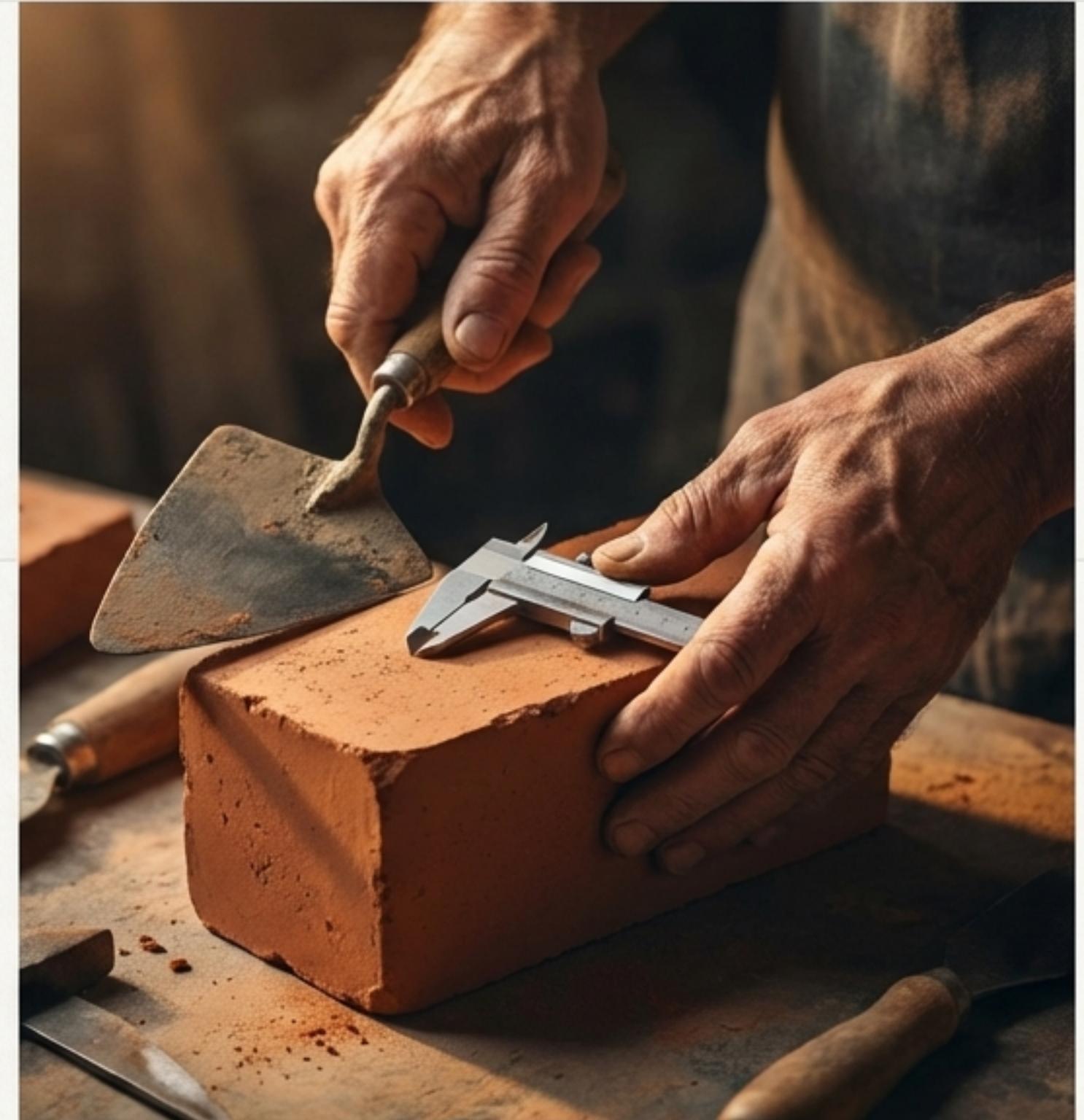
- 이 선언은 프롬프트 엔지니어링이 불필해졌다는 의미가 아닙니다.
- AI와의 소통 방식이 ‘단일 질문’을 잘 다듬는 기술에서, AI가 최상의 결과를 내도록 ‘환경 전체’를 설계하는 시스템 공학으로 확장되고 있음을 의미하는 중요한 신호입니다.
- 프롬프트 엔지니어링은 이제 더 큰 그림, 즉 컨텍스트 엔지니어링의 핵심 구성 요소가 되었습니다.

Part 1. 완벽한 벽돌 만들기: 프롬프트 엔지니어링의 정수

AI가 내 말을 가장 정확하게 이해하게 만들려면,
우리는 무엇부터 알아야 할까요?

훌륭한 건축은 최상의 자재에서 시작됩니다.
AI와의 소통에서 '프롬프트'는 가장 기본이 되는 '벽돌'입니다.

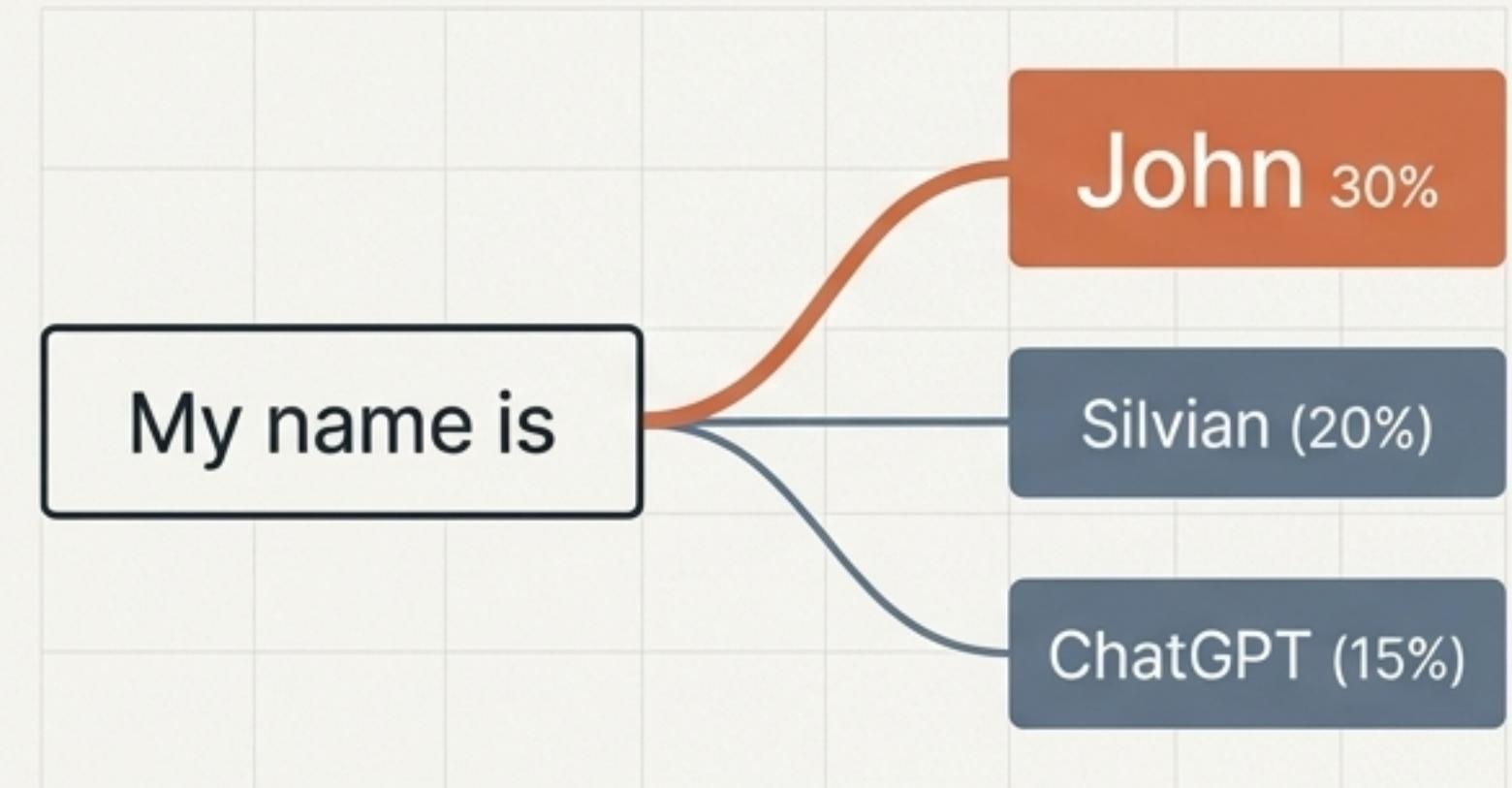
프롬프트 엔지니어링은 바로 이 벽돌의 규격, 강도, 질감을
완벽하게 제어하는 정교한 기술입니다.



AI는 세상을 이해하는 것이 아니라, 다음 단어를 예측할 뿐입니다

거대 언어 모델(LLM)은 문장을 통째로 생성하는 것이 아니라, '토큰(Token)'이라는 단위로 다음 단어를 확률적으로 예측하고 생성합니다.

예시: 'My name is' 다음에 올 단어로 'John'(30%), 'Silvian'(20%), 'ChatGPT'(15%) 등의 확률값을 계산하고, 이 중 가능성 높은 단어를 선택하거나 특정 파라미터(Temperature 등)에 따라 샘플링합니다.



- LLM의 답변은 데이터베이스에 존재하는 텍스트들을 재조합(Paraphrasing)하여 가장 그럴듯한 문장을 만드는 과정입니다.
- **한국어의 특수성:** 한국어는 글자 단위로 토큰이 쪼개지는 경우가 많아, 영어에 비해 동일한 의미 전달에 더 많은 토큰이 소모되어 API 비용 부담이 클 수 있습니다.

확률적 생성의 치명적 약점: 환각 (Hallucination)

환각이란, LLM이 사실이 아닌 정보를 마치 실제 있었던 것처럼
그럴듯하게 지어내는 현상을 의미합니다.

이는 모델의 작동 원리 때문입니다. LLM은 사실 관계를 확인하는
것이 아니라, 주어진 질문과 가장 확률적으로 연관성이 높은
것이 아니라, 주어진 질문과 가장 확률적으로 연관성이 높은
단어들을 조합하여 문장을 생성합니다. 데이터베이스 안의 소설,
인터넷 게시글 등 허구의 정보도 '그럴듯하게' 조합될 수 있습니다.

질문:

세종대왕 맥북 던짐 사건에 대해 알려줘.

AI의 환각 답변 (예시):

세종대왕 맥북 던짐 사건은 조선왕조실록에 기록된 일화로,
세종대왕이 훈민정음 창제 과정에서 스트레스를 받아 어좌에
있던 맥북 프로를 신하들에게 던졌다는 이야기입니다...



환각을 줄이는 정교한 벽돌 제작술: PE의 3대 원칙



1. 구체적인 명령 (Detailed Instruction)

AI는 명령이 모호하면 스스로 추측하여 답변하므로, 환각의 가능성이 높아집니다. 원하는 결과물의 형식, 톤앤매너, 길이, 포함할 내용 등을 최대한 구체적으로 명시해야 합니다.

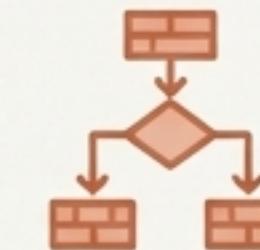
예시: '요약해 줘' (X) -> '이 글을 초등학생도 이해할 수 있도록 3문단으로 요약하고, 친절한 말투를 사용해 줘' (O)



2. 데이터와 문맥 제공 (In-Context Learning)

AI는 입력된 데이터를 기반으로 답변을 생성합니다. 따라서 질문과 관련된 정확하고 양질의 정보를 함께 제공하면, 원하는 답변을 출력할 확률이 극적으로 올라갑니다.

기법: **Few-Shot Learning** - AI에게 원하는 결과물의 예시(Exemplars)를 몇 개 보여주어 패턴을 학습시키는 방식.



3. 알고리즘적 구조 (Algorithmic Structure)

줄글로 길게 설명하기보다, AI가 이해하기 쉽도록 단계별(Step-by-step) 지침이나 마크다운(Markdown)을 활용하여 구조화된 형태로 명령을 전달하는 것이 효과적입니다.

기법: **Chain-of-Thought (CoT)** - 복잡한 문제에 대해 결론만 요구하지 않고, 문제를 해결하는 '사고 과정'을 단계별로 제시하도록 유도하는 기법.

하지만 완벽한 벽돌만으로는 위대한 건축물이 되지 않습니다.



1. 프롬프트의 취약성 (Prompt Fragility)

단어 하나, 조사 하나만 바뀌어도 결과가 완전히 달라질 수 있습니다.

LLM 모델이 조금만 업데이트되어도 어제까지 완벽했던 프롬프트가 오늘은 작동하지 않는 문제가 발생하여, 안정적인 서비스 구축을 어렵게 합니다.

2. 지식의 단절 (Knowledge Cut-off)

AI는 특정 시점까지의 데이터로만 학습되어 최신 정보나 기업 내부 데이터 같은 비공개 정보는 알지 못합니다.

결국, '세종대왕 맥북 던짐 사건'과 같은 환각 문제는 외부 지식 없이는 근본적으로 해결하기 어렵습니다.

완벽한 '질문 하나'를 만드는 것만으로는 밑 빠진 독에 물붓기일 수 있습니다.

이제 질문을 넘어, AI에게 답을 찾는데 필요한 '환경'을 시스템적으로 제공해야 합니다.



Part 2. 견고한 세계의 설계: 컨텍스트 엔지니어링의 모든 것

AI가 스스로 정보를 찾고, 도구를 사용하며,
신뢰할 수 있는 전문가가 되게 하려면,
우리는 무엇을 제공해야 할까요?

컨텍스트 엔지니어링은 개별 벽돌(프롬프트)을 넘어,
건물 전체의 '청사진'을 그리는 일입니다.

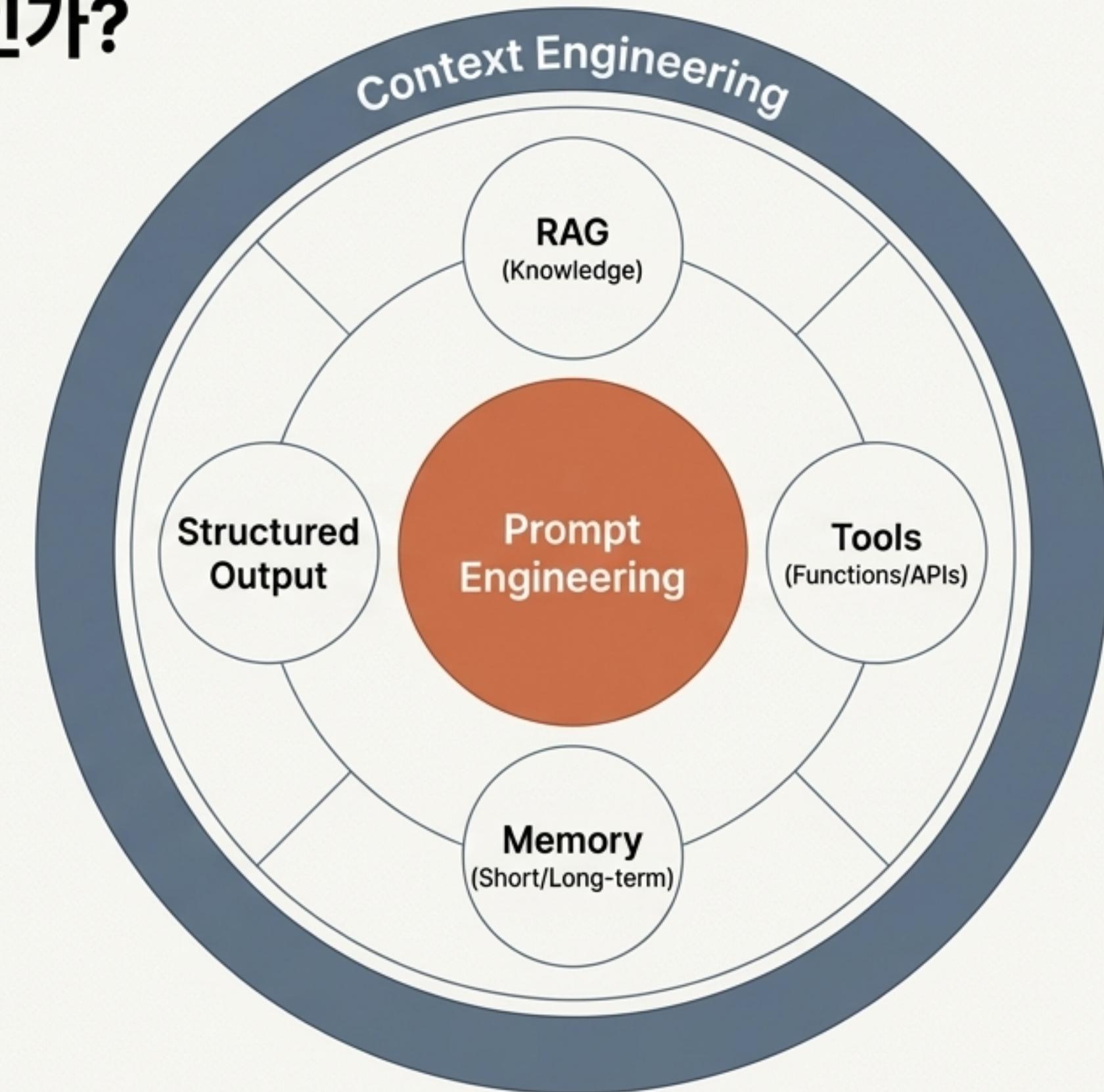
단순히 명령하는 것을 넘어, AI가 작업을 성공적으로
수행하는 데 필요한 모든 환경을 설계하는 시스템 공학입니다.

컨텍스트 엔지니어링(CE)이란 무엇인가?

LLM이 주어진 과업을 안정적이고 신뢰성 있게 해결하도록, 필요한 정보(Information), 도구(Tools), 지침(Instructions), 기억(Memory)을 하나의 뮤음(Context)으로 구성하여 동적으로 제공하는 시스템을 설계하는 모든 활동.

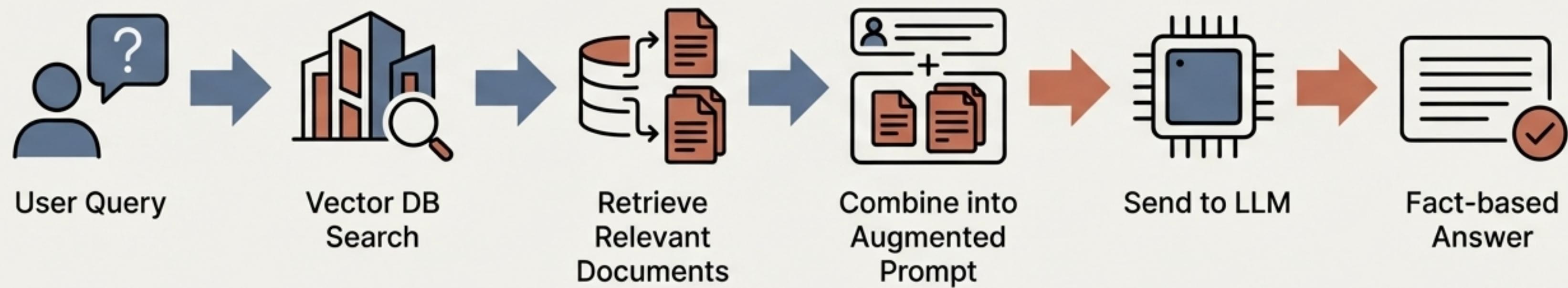
Mindset Shift

- Prompt Engineering:
“어떻게 질문할까?” (How to ask?)
- Context Engineering:
“AI가 성공하려면 무엇을 알고, 무엇을 할 수 있어야 할까?”
(What does the AI need to know and do?)



CE 핵심 기술 1: RAG - AI에게 최신 도서관을 지어주다

기술명 : RAG (Retrieval-Augmented Generation, 검색 증강 생성)
역할 : '지식 단절'과 '환각' 문제를 해결하는 가장 대표적인 기술.



효과: '세종대왕 맥북 던짐 사건'에 대해 질문 시, RAG 시스템은 '세종대왕' 관련 역사적 사실을 먼저 검색하여 '세종대왕은 15세기 인물이며, 맥북은 20세기에 발명되었습니다. 따라서 해당 사건은 역사적 사실이 아닙니다.'와 같은 정확한 답변을 생성합니다.

CE 핵심 기술 2: Tools & Agents

- AI에게 손과 발을 달아주다

- LLM이 텍스트 생성 이상의 '행동'을 할 수 있도록 외부 도구(Tools)를 호출하고 사용하는 능력.
- 도구의 예시: 웹 검색 API, 데이터베이스 쿼리, 항공권 예약 API, 캘린더 연동 등
- **Agentic AI**: 단순히 답변하는 것을 넘어, 목표를 설정하고, 계획을 세우고, 필요한 도구를 사용하며, 스스로 피드백을 통해 결과를 수정해 나가는 자율적인 시스템.
- **핵심 프레임워크: ReAct** (Reason + Act) - AI가 '생각(Thought)'과 '행동(Action)'을 순차적으로 생성하며 작업을 수행하는 방식.



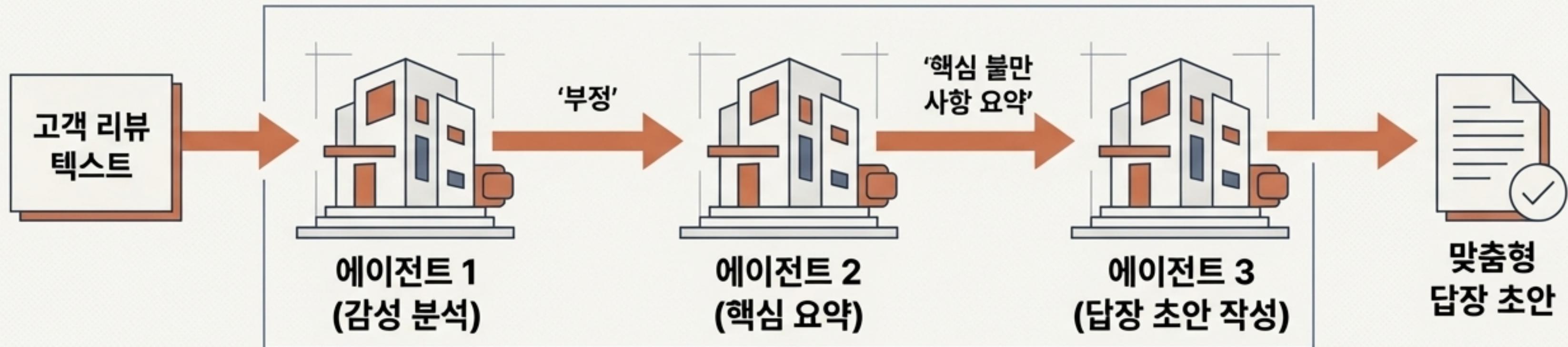
RAG가 건물에 '도서관'을 만들어주는 것이라면, Tool 사용은 '전기, 수도, 인터넷'과 같은 기반 시설을 연결하여 건물이 실제로 기능하게 만드는 것과 같습니다.

CE 고급 기술: 프롬프트 체이닝

- 전문가 팀을 구성하다

Definition: 하나의 복잡한 작업을 여러 개의 간단한 작업으로 분해하고, 각 작업을 전문화된 별도의 LLM(또는 프롬프트)이 순차적으로 처리하도록 연결하는 기법.

Core Idea: 한 작업의 결과물(Output)이 다음 작업의 입력(Input)으로 사용됩니다.



- 각 에이전트는 하나의 작은 임무에만 집중하므로 더 높은 정확성과 신뢰성을 보입니다.
- 문제 발생 시 디버깅이 용이하며, 전체 시스템의 유지보수성이 향상됩니다.

많이 넣는다고 무조건 좋을까? 거대 컨텍스트의 4가지 함정

'최신 모델들은 컨텍스트 윈도우가 100만 토큰 이상으로 커졌습니다. 그렇다면 그냥 모든 정보를 다 넣으면 되지 않을까요?'
... 아닙니다. 관리되지 않은 방대한 컨텍스트는 오히려 AI의 성능을 저하시킵니다.

1. 컨텍스트 오염 (Context Poisoning)

대화 중 발생한 작은 오류나 환각이 컨텍스트에 포함되어, 이후의 모든 추론을 오염시키는 현상.



2. 컨텍스트 분산 (Context Distraction)

컨텍스트가 너무 길어지면, AI가 새로운 계획을 세우기보다 과거 기록에만 과도하게 집중하여 반복적인 행동을 보이는 현상.
(Needle-in-a-Haystack 문제)



3. 컨텍스트 혼란 (Context Confusion)

작업과 무관한 도구나 정보가 컨텍스트에 포함될 경우, AI가 주의를 잃고 엉뚱한 도구를 호출하거나 관련 없는 답변을 생성하는 현상.



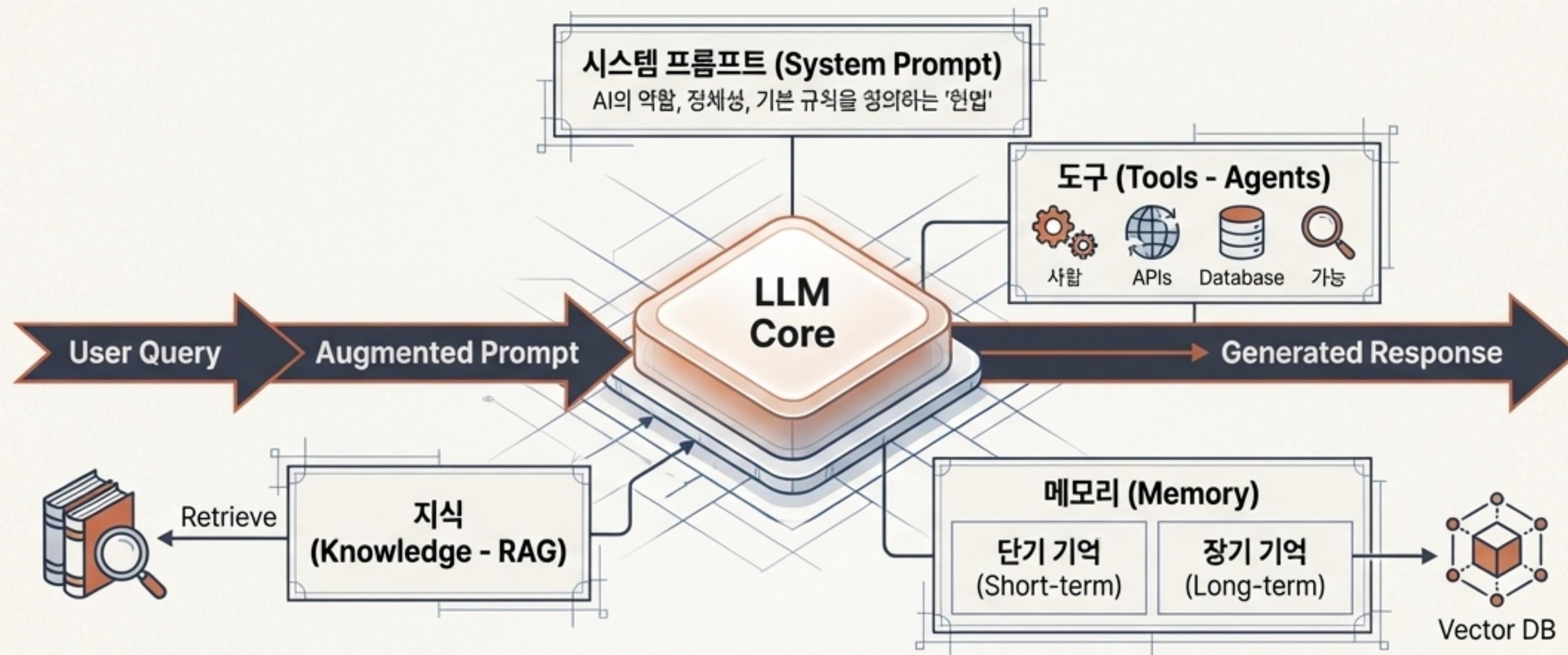
4. 컨텍스트 충돌 (Context Clash)

대화 과정에서 서로 모순되거나 상충하는 정보가 컨텍스트에 쌓여 AI의 추론을 방해하는 현상.



컨텍스트 엔지니어링의 핵심은 쓰레기 더미를 봇는 것이 아니라,
AI가 소화할 수 있도록 필요한 정보를 '선별'하고 '구조화'하여 제공하는 능력입니다.

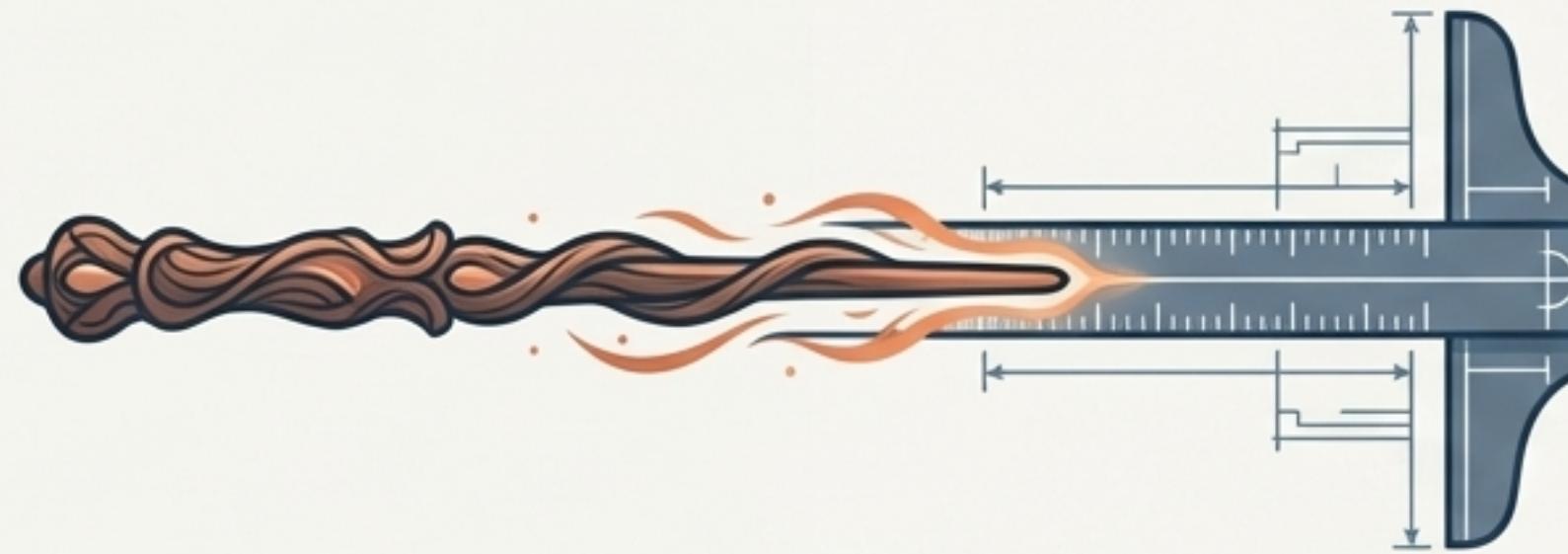
최종 설계도: 현대적 AI 시스템 아키텍처



이 아키텍처 안에서 **프롬프트 엔지니어링**은 각 구성요소(시스템 프롬프트, RAG 쿼리, 툴 사용 지침 등)에 들어가는 '명령어'를 정교하게 작성하는 기술로 작동합니다.

컨텍스트 엔지니어링은 이 모든 구성요소를 유기적으로 연결하고 데이터 흐름을 설계하여, 시스템 전체가 안정적으로 목표를 달성하도록 만드는 상위 레벨의 엔지니어링입니다.

당신은 AI의 마법사입니까, 건축가입니까?



- 우리는 단 하나의 완벽한 '**프롬프트**'를 찾는 여정에서 시작했습니다. (**마법사의 주문**)
- 그러나 이제 우리는 AI가 최고의 성능을 발휘할 수 있는 견고한 '**시스템**'을 구축하는 단계에 이르렀습니다. (**건축가의 설계**)

- AI의 미래는 무엇을 '**묻느냐**'가 아니라, AI가 당신의 질문에 답하기 위해 무엇을 '**알고 있느냐**'에 달려있습니다.
- 가장 중요한 능력은 화려한 언어적 기교가 아니라, 시스템 전체를 설계하는 **논리적 기획력**과 **엔지니어링 원칙**이 될 것입니다.