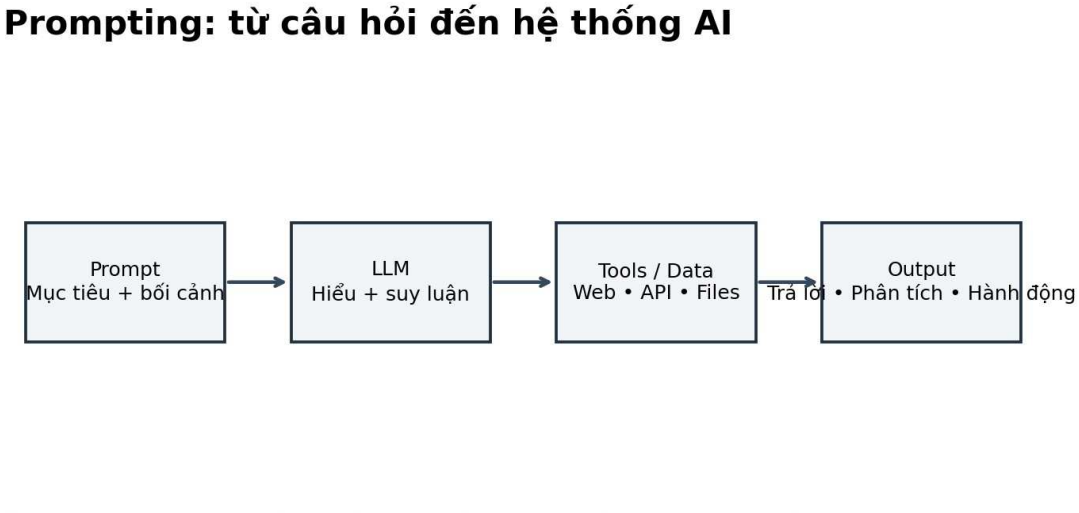


Hình 1. Bức tranh tổng thể: Prompting → LLM → Tools/Data → Output

PROMPT ENGINEERING & LLM APPLICATIONS

TỪ PROMPT ĐẾN HỆ THỐNG AI THỰC CHIẾN

Sổ tay thực hành tổng hợp từ Prompt Engineering Guide

Phiên bản tiếng Việt - thuật ngữ chuyên ngành giữ nguyên English và giải thích trong ngoặc

MỤC LỤC

01. Phần I - Nền tảng và tư duy prompting

02. Phần II - Tối ưu prompt như một quy trình kỹ thuật

03. Phần III - Deep Research và Reasoning LLMs

04. Phần IV - Context Engineering

05. Phần V - Image Generation với 4o

06. Phần VI - Function Calling và Prompt Function

07. Phần VII - Fine-Tuning và dữ liệu tổng hợp

08. Phần VIII - Ứng dụng: coding và workplace case study

09. Phần IX - Context Caching và kiến trúc tối ưu

10. Phần X - Bản đồ triển khai và checklist

Cách đọc tài liệu

Đọc theo thứ tự nếu bạn mới bắt đầu. Nếu đã dùng LLM thường xuyên, có thể đi thẳng tới Phần II, III, IV hoặc các chương ứng dụng. Các ví dụ trong sách được viết lại để dễ hiểu và không phụ thuộc vào việc sao chép nguyên văn ví dụ của nguồn.

PHẦN I - NỀN TẢNG VÀ TƯ DUY PROMPTING

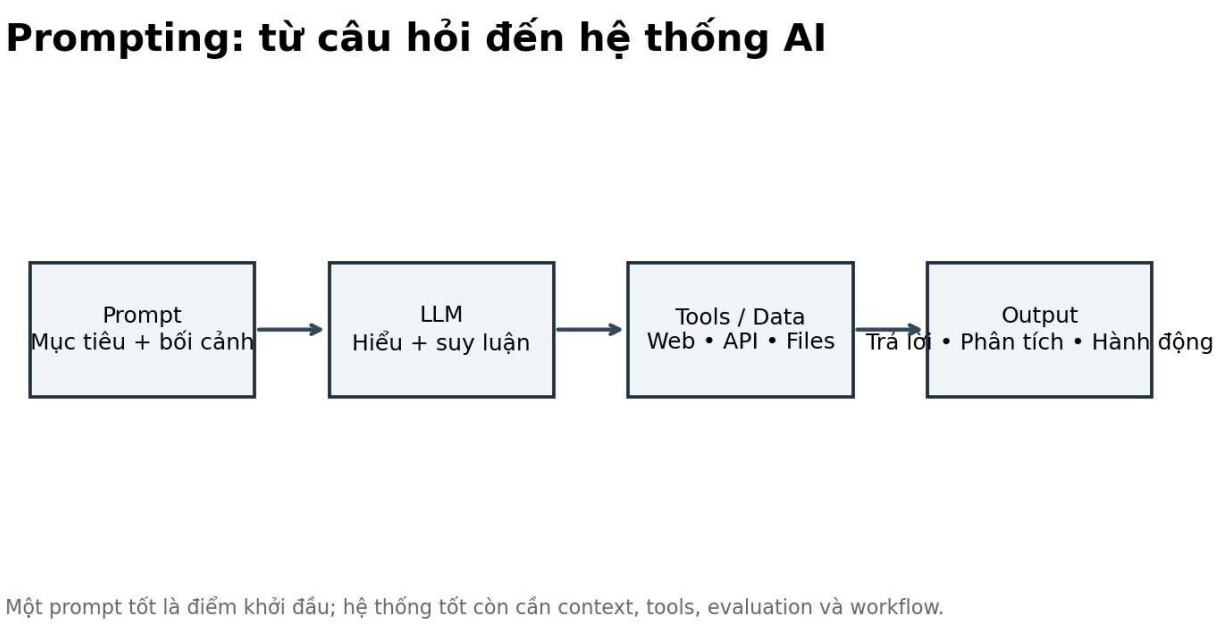
1. Prompt Engineering là gì?

Prompt Engineering (kỹ thuật thiết kế và tối ưu chỉ dẫn cho mô hình ngôn ngữ) là quá trình xây dựng cách giao việc cho LLM sao cho mô hình hiểu đúng mục tiêu, có đủ thông tin cần thiết và tạo ra đầu ra phù hợp. Prompting không chỉ là viết một câu hỏi hay; nó là cách thiết kế giao diện giữa con người và mô hình.

Hiểu trong 10 giây

Prompt giống như bản brief cho một cộng sự rất giỏi nhưng không tự biết bạn muốn gì. Bạn càng nói rõ mục tiêu, bối cảnh, dữ liệu và đầu ra, kết quả càng dễ kiểm soát.

Một prompt tốt thường trả lời bốn câu hỏi: Bạn muốn làm gì? Dựa trên thông tin nào? Điều gì cần tránh hoặc tuân thủ? Kết quả cuối cùng phải trông như thế nào?



Hình 2. Prompt chỉ là điểm bắt đầu của một hệ thống AI.

2. Vì sao “prompt hay” chưa đủ?

Khi ứng dụng thực tế, chất lượng không còn phụ thuộc riêng vào câu chữ. Bạn còn phải quản lý context (ngữ cảnh), dữ liệu, tool, cấu trúc output, lịch sử tương tác và cách đánh giá. Đây là lý do khái niệm Context Engineering (kỹ thuật thiết kế toàn bộ ngữ cảnh) ngày càng quan trọng.

Ví dụ: Một chatbot nội bộ trả lời sai không nhất thiết vì model “kém”. Nó có thể đang nhận tài liệu cũ, truy xuất sai đoạn văn, không biết ưu tiên nguồn chính thức, hoặc bị nhồi quá nhiều lịch sử hội thoại.

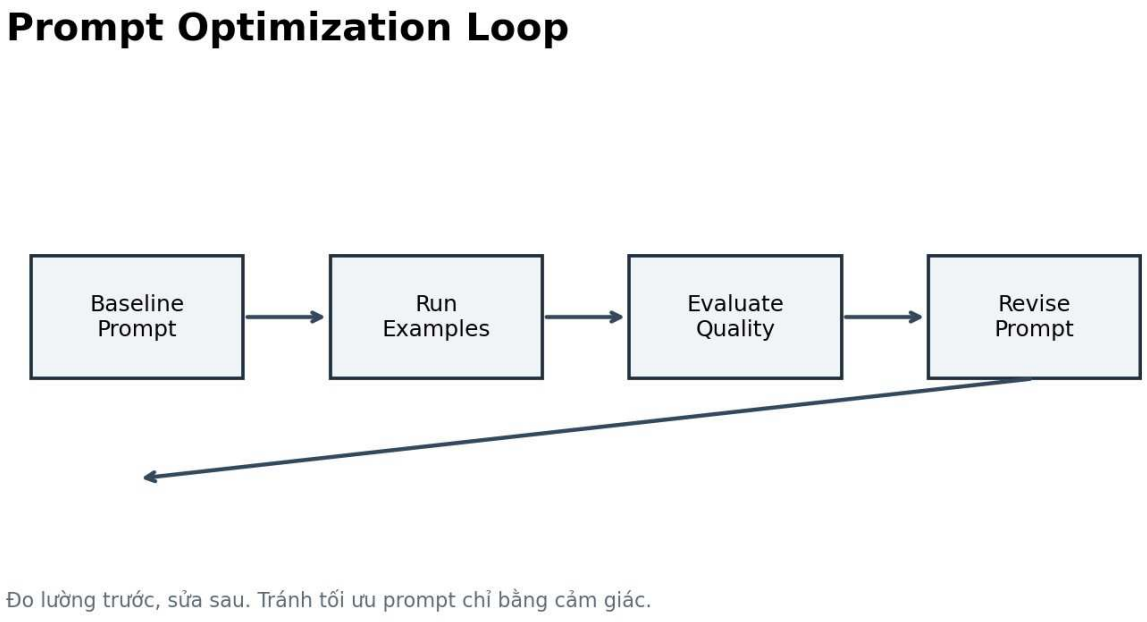
Thành phần	Câu hỏi cần kiểm tra
Instruction	Model cần làm chính xác việc gì?
Context	Thông tin nào thực sự liên quan?
Tools / Data	Model cần đọc, tra cứu hoặc hành động qua đâu?
Output	Kết quả phải có cấu trúc và tiêu chí nào?
Evaluation	Làm sao biết câu trả lời tốt hay chưa?

PHẦN II - TỐI ƯU PROMPT NHƯ MỘT QUY TRÌNH KỸ THUẬT

3. Specificity và Clarity

Specificity (cụ thể) giúp mô hình biết phạm vi. Clarity (rõ ràng) giúp mô hình hiểu nhiệm vụ mà không phải đoán. Hai nguyên tắc này đặc biệt quan trọng khi prompt có nhiều bước hoặc yêu cầu định dạng.

Ví dụ yếu: “Viết cho tôi báo cáo về AI.”
Ví dụ tốt hơn: “Viết bản tóm tắt 1 trang về 5 xu hướng AI trong ngân hàng, tập trung vào doanh thu, CASA và năng suất. Dùng bảng 4 cột: xu hướng, ví dụ thị trường, tác động, việc ngân hàng nên cân nhắc.”



Hình 3. Prompt nên được tối ưu bằng vòng lặp đo lường.

4. Structured Input / Output và Delimiters

Structured Input/Output (đầu vào/đầu ra có cấu trúc) giúp giảm mơ hồ. Delimiters (ký hiệu phân tách) giúp mô hình nhận ra đâu là instruction, đâu là dữ liệu. JSON, XML, Markdown headings hoặc các thẻ như <DOCUMENT>...</DOCUMENT> đều có thể hữu ích, tùy bài toán.

Cách làm	Khi nên dùng	Ví dụ
Markdown	Báo cáo, tài liệu đọc	## Findings / ## Risks / ## Actions
JSON	Tích hợp phần mềm	{"customer":"...","risk":"high"}
XML / tags	Tách instruction và data	<context>...</context>
Delimiters	Dữ liệu dài hoặc dễ nhầm	--- START DATA --- ... --- END DATA ---

5. Task Decomposition

Task Decomposition (phân rã nhiệm vụ) là chia một yêu cầu lớn thành các bước nhỏ. Cách này hữu ích khi một prompt phải vừa đọc dữ liệu, vừa phân tích, vừa tạo kết quả cuối. Thay vì làm tất cả trong một lần, có thể xây một chuỗi bước: extract → analyze → draft → check → finalize.

Ví dụ

Thay vì “đọc 30 tài liệu và đề xuất chiến lược”, hãy tách thành: (1) trích xuất facts; (2) nhóm chủ đề; (3) đối chiếu mâu thuẫn; (4) rút ra implications; (5) viết recommendation có dẫn chứng.

6. Prompt Optimization thực tế

Tối ưu prompt nên được xem là một experiment (thí nghiệm) có đầu vào, tiêu chí đánh giá và vòng lặp. Một cách đơn giản: giữ nguyên dữ liệu thử nghiệm, thay đổi một yếu tố, đo kết quả, rồi ghi nhận version tốt hơn.

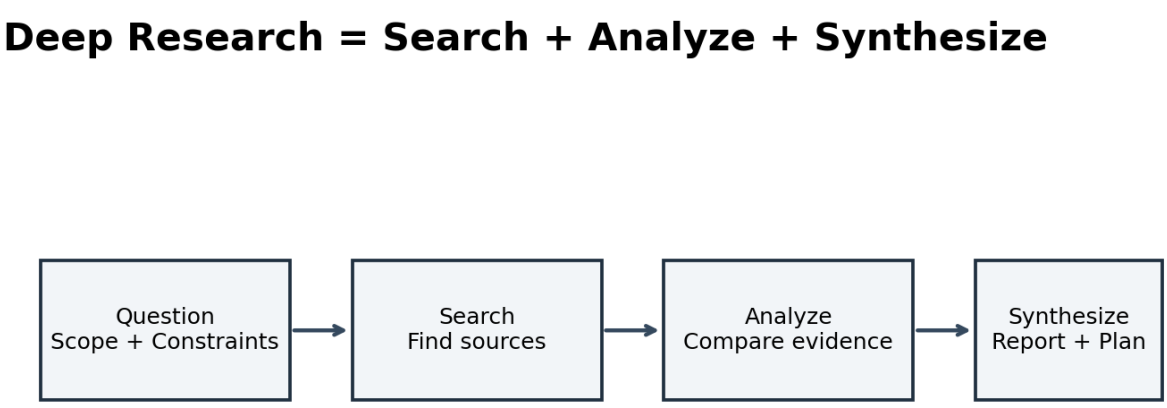
Bước	Làm gì?
Baseline	Chạy prompt hiện tại trên tập ví dụ cố định.
Evaluate	Chọn KPI phù hợp: accuracy, completeness, format adherence, latency, cost.
Change one thing	Sửa instruction hoặc output format, tránh thay quá nhiều biến.
Compare	So sánh trước/sau trên cùng tập.
Version	Lưu prompt + lý do thay đổi + kết quả.

Nguồn tham khảo: https://www.promptingguide.ai/guides/optimizing-prompts

PHẦN III - DEEP RESEARCH VÀ REASONING LLMS

7. Deep Research

Deep Research (nghiên cứu chuyên sâu nhiều bước) phù hợp với câu hỏi cần tìm kiếm nhiều nguồn, phân tích chéo và tổng hợp thành báo cáo. Prompt Engineering Guide mô tả quy trình cốt lõi theo logic Search + Analyze + Synthesize, từ đó tạo Report, Insights và Action Plans. ŅciteÖturn384517view1Ọ



Hình 4. Mô hình Search + Analyze + Synthesize.

Ví dụ: thay vì hỏi “Xu hướng AI ngân hàng 2026 là gì?”, hãy yêu cầu: “Nghiên cứu 3 thị trường Việt Nam, Trung Quốc và Singapore; ưu tiên nguồn ngân hàng, regulator và báo cáo doanh nghiệp; so sánh 5 use case tạo doanh thu; nêu dữ liệu hỗ trợ, điểm chưa chắc chắn và đề xuất thử nghiệm.”

Deep Research đặc biệt hữu ích cho market analysis, literature review, policy/regulatory research, feasibility study, phân tích tài liệu tải lên kết hợp nguồn web và tổng hợp nhiều nguồn. ŅciteÖturn770396view0Ọ

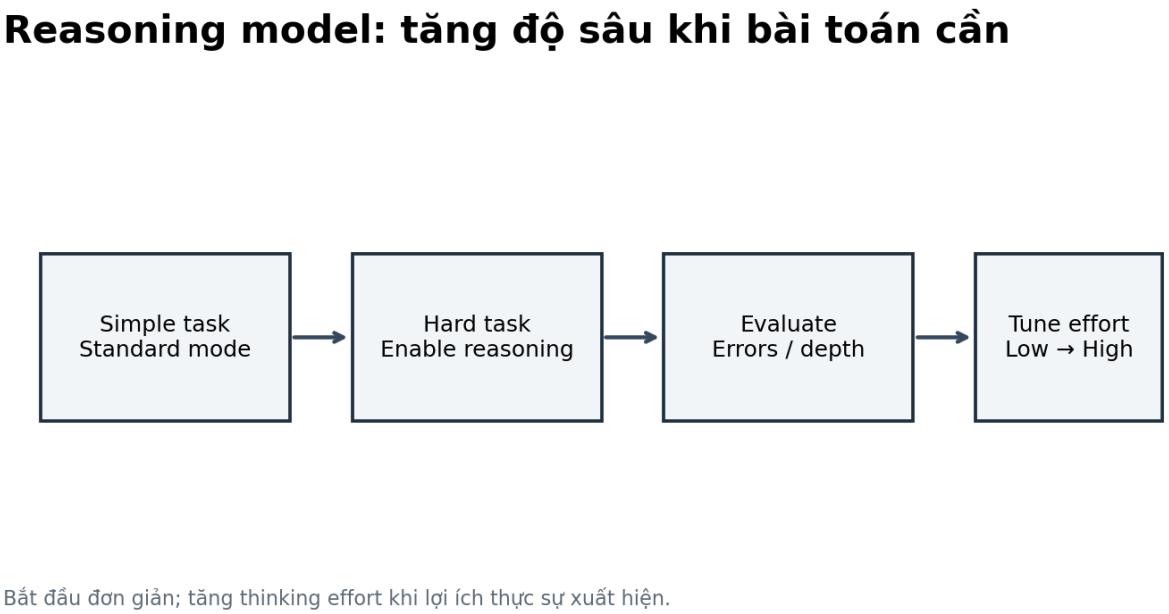
Ghi nhớ

Deep Research không thay thế việc kiểm chứng. Với vấn đề có tác động lớn, hãy kiểm tra các nguồn quan trọng và phân biệt rõ fact, interpretation và recommendation.

Nguồn tham khảo: https://www.promptingguide.ai/guides/deep-research

8. Reasoning LLMs

Reasoning LLMs (mô hình được tối ưu cho suy luận) được thiết kế để xử lý các nhiệm vụ cần phân tích sâu hơn. Guide hiện nhấn mạnh một cách làm thực dụng: bắt đầu bằng chế độ chuẩn, đánh giá; chỉ bật reasoning khi thấy task thực sự hưởng lợi; sau đó tăng thinking effort từ thấp lên cao nếu cần. ŅciteÖturn205268view3Ô



Hình 5. Tăng độ sâu suy luận theo nhu cầu, không mặc định tối đa.

Ví dụ: với “hãy viết lại câu này cho tự nhiên”, reasoning cao thường không cần. Nhưng với “phân tích 3 phương án đầu tư, kiểm tra giả định, tìm mâu thuẫn và lập bảng nhạy cảm”, suy luận sâu có thể mang lại lợi ích rõ hơn.

Điểm cần lưu ý

Guide cũng cảnh báo reasoning models có thể gặp overthinking, underthinking, chi phí và latency cao hơn; explicit Chain-of-Thought prompting đôi khi còn làm giảm instruction-following. Vì vậy nên ưu tiên instruction rõ và lựa chọn effort phù hợp task. ŅciteÖturn239752view4Ô

Nguồn tham khảo: https://www.promptingguide.ai/guides/reasoning-llms

PHẦN IV - CONTEXT ENGINEERING

9. Context Engineering

Context Engineering (kỹ thuật thiết kế và tối ưu toàn bộ ngữ cảnh) mở rộng Prompt Engineering. Thay vì chỉ sửa câu lệnh, người xây hệ thống phải quyết định thông tin nào đi vào context window, theo thứ tự nào, nguồn nào đáng tin, tool nào được phép dùng, memory nào cần giữ và phần nào nên loại bỏ.