Prompt chaining

made with ❤️ for “Little ML book club”


analyze a market research report, summarize findings, identify trends with data points, and draft an email
Initial Prompt (Summarization)
Second Prompt (Trend Identification)
Third Prompt (Email Composition):


analyze a market research report, summarize findings, identify trends with data points, and draft an email
Initial Prompt (Summarization)
Second Prompt (Trend Identification)
Third Prompt (Email Composition):

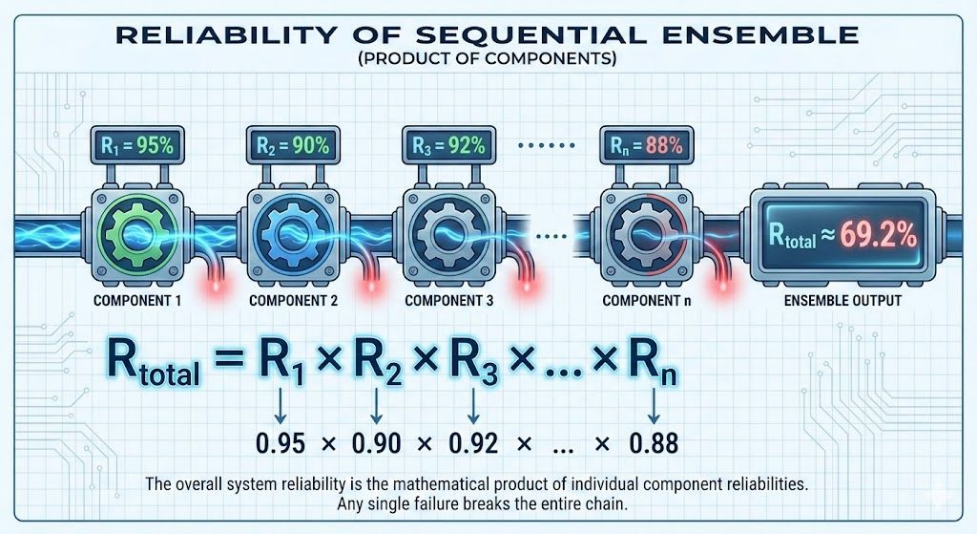
S O L I D

```
{
 "trends": [
   {
     "trend_name": "AI-Powered Personalization",
     "supporting_data": "73% of consumers prefer to do business with
brands that use personal information to make their shopping
experiences more relevant."
   },
   {
     "trend_name": "Sustainable and Ethical Brands",
     "supporting_data": "Sales of products with ESG-related claims
grew 28% over the last five years, compared to 20% for products
without."
   }
 ]
}
```

```
{
 "trends": [
   {
     "trend_name": "AI-Powered Personalization",
     "supporting_data": "73% of consumers prefer to do business with
brands that use personal information to make their shopping
experiences more relevant."
   },
   {
     "trend_name": "Sustainable and Ethical Brands",
     "supporting_data": "Sales of products with ESG-related claims
grew 28% over the last five years, compared to 20% for products
without."
   }
 ]
}
```


- Prompt 1: Extract text content from a given URL or document.
- Prompt 2: Summarize the cleaned text.
- Prompt 3: Extract specific entities (e.g., names, dates, locations) from the summary or original text.
- Prompt 4: Use the entities to search an internal knowledge base.
- Prompt 5: Generate a final report incorporating the summary, entities, and search results.

- Prompt 1: Extract text content from a given URL or document.
- Prompt 2: Summarize the cleaned text.
- Prompt 3: Extract specific entities (e.g., names, dates, locations) from the summary or original text.
- Prompt 4: Use the entities to search an internal knowledge base.
- Prompt 5: Generate a final report incorporating the summary, entities, and search results.



0.95^5 = 0.774
0.98^10 = 0.81

- Prompt 1: Generate 5 topic ideas based on a user's general interest.
- Processing: Allow the user to select one idea or automatically choose the best one.
- Prompt 2: Based on the selected topic, generate a detailed outline.
- Prompt 3: Write a draft section based on the first point in the outline.
- Prompt 4: Write a draft section based on the second point in the outline, providing the previous section for context. Continue this for all outline points.
- Prompt 5: Review and refine the complete draft for coherence, tone, and grammar.

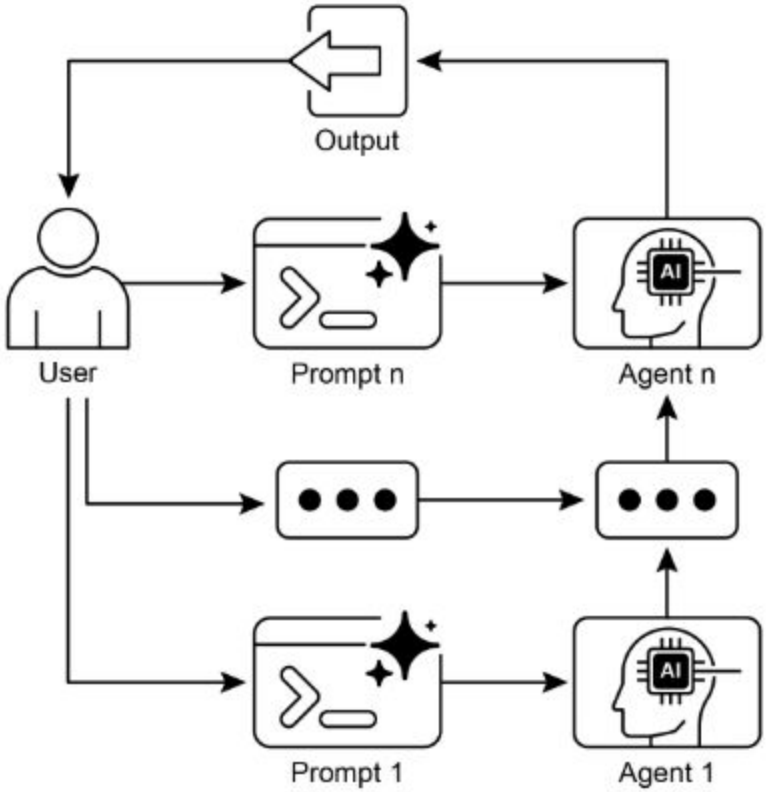
- Prompt 1: Understand the user's request for a code function. Generate pseudocode or an outline.
- Prompt 2: Write the initial code draft based on the outline.
- Prompt 3: Identify potential errors or areas for improvement in the code (perhaps using a static analysis tool or another LLM call).
- Prompt 4: Rewrite or refine the code based on the identified issues.
- Prompt 5: Add documentation or test cases.

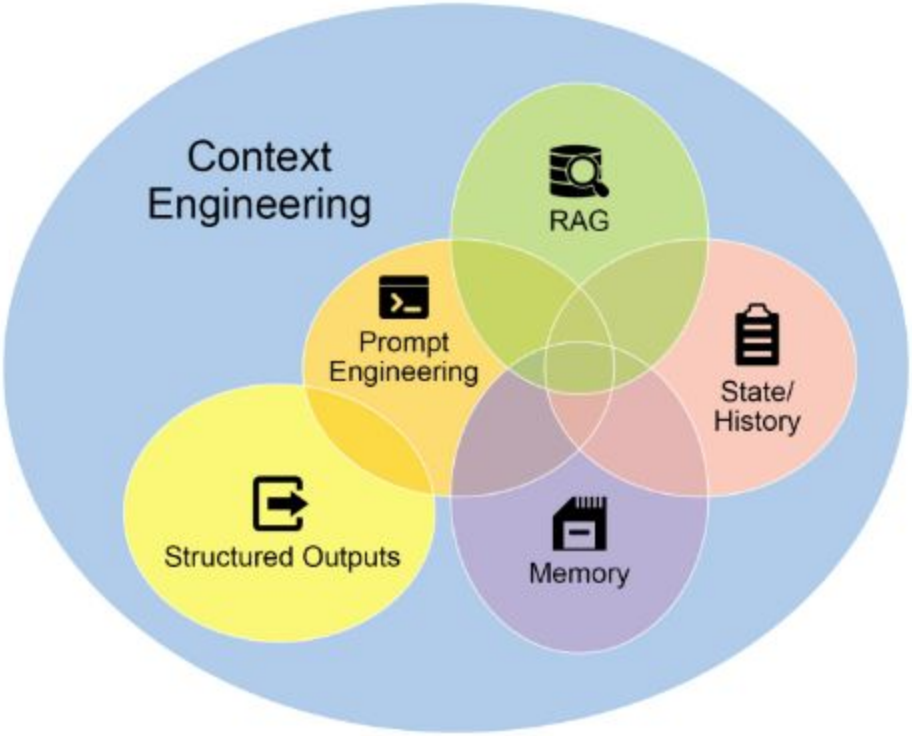
```python
import os
from langchain_openai import ChatOpenAI
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.output_parsers import StrOutputParser

# Initialize the Language Model
llm = ChatOpenAI(temperature=0)

# --- Prompt 1: Extract Information ---
prompt_extract = ChatPromptTemplate.from_template(
"Extract the technical specifications from the following
text:\n\n{text_input}"
)
# --- Prompt 2: Transform to JSON ---
prompt_transform = ChatPromptTemplate.from_template(
"Transform the following specifications into a JSON object with
'cpu', 'memory', and 'storage' as keys:\n\n{specifications}"
)

# --- Build the Chain using LCEL ---
# The StrOutputParser() converts the LLM's message output to a simple string.
extraction_chain = prompt_extract | llm | StrOutputParser()
# The full chain passes the output of the extraction chain into the 'specifications'
# variable for the transformation prompt.
full_chain = (
{"specifications": extraction_chain}
| prompt_transform
| llm
| StrOutputParser()
)
# --- Run the Chain ---
input_text = "The new laptop model features a 3.5 GHz octa-core processor, 16GB of RAM, and a 1TB NVMe
SSD."
# Execute the chain with the input text dictionary.
final_result = full_chain.invoke({"text_input": input_text})
print("\n--- Final JSON Output ---")
print(final_result)
```


Output
User
Prompt n
AI
Agent n
Prompt 1
AI
Agent 1


Context Engineering
RAG
Prompt Engineering
State/ History
Structured Outputs
Memory

"Let Me Speak Freely?" — Tam et al. (EMNLP 2024 Industry, arXiv:2408.02442)
GPT-4o-mini dropped from 94.57% → 86.95% on GSM8K under JSON constraints

"Say What You Mean" — dottxt/Outlines team rebuttal (Nov 2024, blog.dottxt.ai)
Argued the original paper's prompts were not equivalent between structured and unstructured conditions
LLaMA-3-8B-Instruct, structured generation outperformed unstructured on all three reasoning tasks — GSM8K 77.79% vs 77.18%, Last Letter +4pp, Shuffled Objects +3.6pp

Independent replication — Dylan Castillo (Dec 2024, updated May 2025)
Confirmed dottxt's LLaMA-3-8B results, but GPT-4o-mini told a different story: Shuffled Objects with 3-shot JSON-Schema collapsed to 65.85% vs 92.68% for natural languag

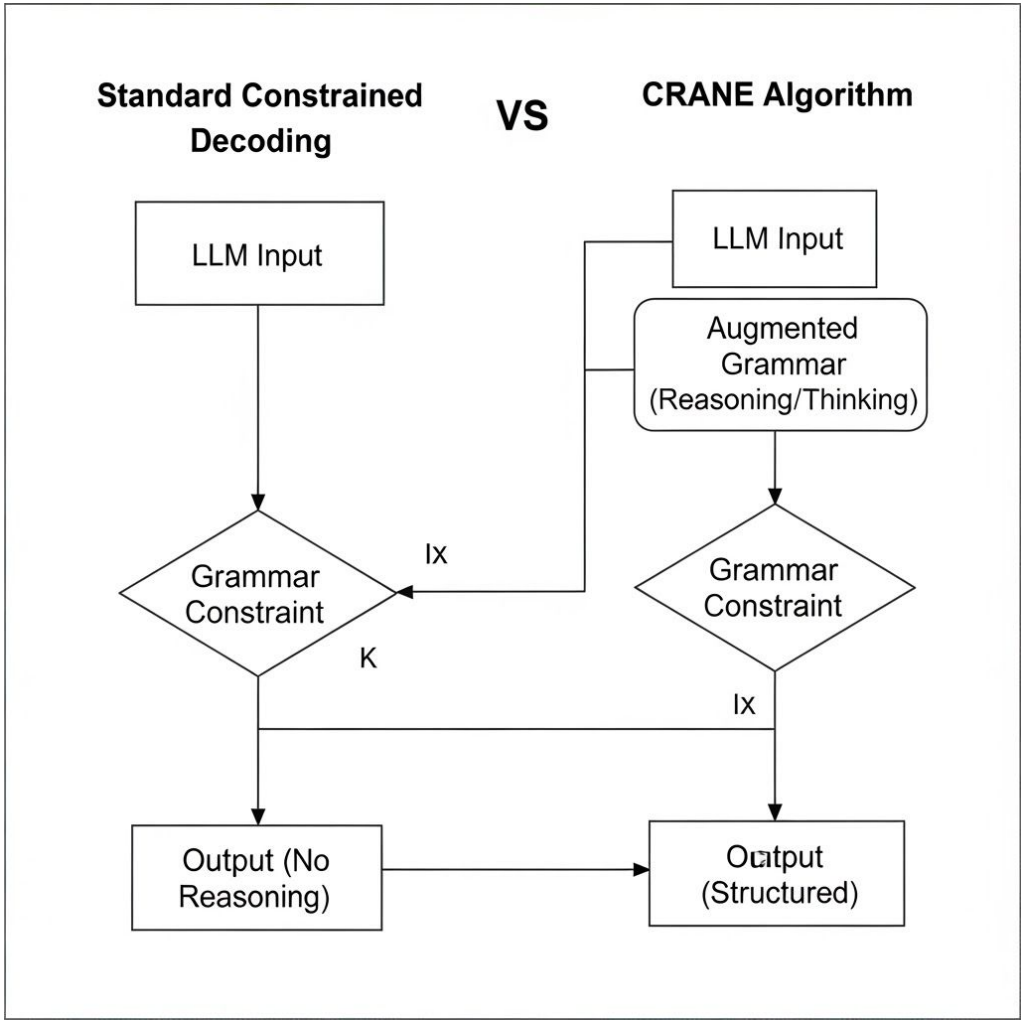
JSONSchemaBench https://arxiv.org/pdf/2501.10868

Table 8: Performance Percentages for Various Models

	Last Letters	Shuffle Objects	GSM8K
LM only	50.7%	52.6%	80.1%
XGrammar	51.2%	52.7%	83.7%
Llamacpp	52.0%	52.6%	82.4%
Outlines	53.3%	53.0%	81.6%
Guidance	**54.0%**	**55.9%**	**83.8**%

Grammar-Aligned Decoding
(NeurIPS 2024, arXiv:2405.21047)

CRANE
(ICML 2025, arXiv:2502.09061)



see you next time
for
“Routing”