Less is More! On the Composition Pattern of Prompts

Anonymous ACL submission

Abstract

Prompting is the primary interface through which users interact with large language models. Although individual prompting strategies have been extensively studied and shown to be effective, it remains unclear whether combining multiple strategies can yield additive benefits. We present a systematic empirical study of prompt pattern composition across multiple LLMs, tasks, and composition mechanisms, evaluating 18 prompt pattern categories over 7 benchmarks spanning 6 task types. Surprisingly, prompt composition is highly non-additive. Fewer than 10% of pairwise compositions outperform their stronger constituent prompt. We further find that prompt accumulation provides only limited early-stage gains, while additive predictions increasingly overestimate actual performance as compositions grow longer. Moreover, selection-based mechanisms consistently outperform direct prompt accumulation. Overall, our findings show that effective prompting should prioritize strong and compatible strategies over simply adding more prompts, suggesting a *less-is-more* principle for prompt engineering.

1 Introduction

As large language models (LLMs) are increasingly integrated into everyday work and information-seeking (Bommasani et al., 2021; Achiam et al., 2023), prompts have become the primary means through which users express intent and steer model behavior (Liu et al., 2023). Rather than updating model parameters or building task-specific systems, users adapt LLMs through natural-language instructions. A growing body of work has shown that prompting strategies such as few-shot prompting (Brown et al., 2020), chain-of-thought reasoning (Wei et al., 2022), self-consistency (Wang et al., 2022), reflection (Shinn et al., 2023), persona assignment (Kong et al., 2024), and structured output templates (Schick and Schütze, 2021) can substantially improve model performance across many tasks.

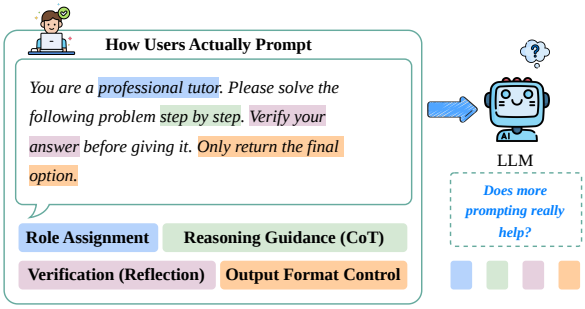

Figure 1: When interacting with LLMs, users often combine multiple prompts in a single query, raising a natural question: does more prompting really help?

While individual prompting strategies have been extensively studied, their behavior under composition remains poorly understood. As illustrated in Figure 1, a user may ask an LLM to "act as a professional tutor," "solve the problem step by step," "verify the answer," and "return only the final option" within a single instruction. Such a prompt is not a single prompting technique, but a composition of role assignment, reasoning guidance, answer verification, and output-format control. Combining these strategies may appear to be a natural way to obtain stronger responses, but it can also introduce redundant instructions, conflicting behavioral constraints, or competing instruction priorities. This motivates our central research question: *Can different prompt patterns be reliably combined to improve LLM performance, and does increasing the number of composed prompt patterns yield scaling-like benefits?*

To answer this question, we conduct a systematic empirical study of prompt pattern composition across multiple LLMs, tasks, and composition mechanisms. Specifically, we evaluate 18 prompt pattern categories, ranging from demonstrations and chain-of-thought reasoning to role assignment,

verification, reflection, output formatting, question refinement, and context control (White et al., 2023). We test these patterns over seven benchmarks spanning six task types, including reasoning, code generation, database querying, table-to-text generation, question answering, and summarization. This setup allows us to examine prompt composition as a general interaction phenomenon rather than an artifact of a single task or prompting style.

We first analyze pairwise compositions to test whether combining two prompting strategies improves over using either strategy alone. For each pair of prompt patterns, we evaluate both prompt orders and compare the composed prompt against its individual components. Surprisingly, we find that prompt composition is highly non-additive. Across models and tasks, fewer than 10% of pairwise compositions outperform their stronger constituent prompt. Even prompt patterns that are effective in isolation often lose performance when paired with other strong patterns, suggesting that prompt quality does not imply compositional compatibility.

We then study larger prompt compositions to examine whether prompt accumulation exhibits scaling-like behavior. Starting from the best-performing single prompt pattern, we incrementally construct compositions through greedy extension and compare the resulting trajectories against local pairwise additive predictions. We find that accumulation yields only limited early-stage gains: performance typically saturates after 4–5 prompt patterns, and additive predictions increasingly overestimate actual performance as compositions grow larger. This suggests that higher-order interference and diminishing marginal effects constrain the benefits of simply adding more prompts.

Finally, we ask how multiple prompt strategies should be utilized if direct accumulation is unreliable. We compare four representative mechanisms: direct concatenation, multi-turn composition, choice-based composition, and best-of-N Composition. We find that selection-based mechanisms consistently outperform direct accumulation methods. This suggests that effective multi-prompt usage depends more on selecting compatible prompting behaviors than on accumulating increasingly complex prompt compositions.

Overall, our findings challenge the common intuition that more prompting guidance naturally leads to better LLM performance. Instead, prompt composition often follows a *less-is-more* principle: performance depends less on adding instructions than on selecting compatible ones. This work provides empirical guidance for designing more effective prompting practices in real-world LLM applications.

Pattern Category	Prompt Pattern
	Vanilla
Input Semantics	*Meta Language Creation*
Output Customization	*Output Automater* *Persona* *Visualization Generator* *Recipe* *Template*
Error Identification	*Fact Check List* *Reflection*
Prompt Improvement	*Question Refinement* *Alternative Approaches* *Cognitive Verifier* *Refusal Breaker*
Interaction	*Flipped Interaction* *Game Play* *Infinite Generation*
Context Control	*Context Manager*
Example Instruction	*Demonstration*
Reasoning	*Chain of Thought*

Table 1: Categories of prompt patterns.

2 Problem Formulation

2.1 Prompt Patterns

We define a *prompt pattern* as a reusable behavioral instruction schema that specifies how an LLM should reason, interact, or format outputs. Unlike a concrete prompt string, a prompt pattern may have multiple surface realizations sharing the same functional role.

We build on the prompt-pattern taxonomy of White et al. (2023), which organizes prompt patterns into six high-level categories and 16 subcategories spanning input semantics, output customization, error identification, prompt improvement, interaction control, and context management, as shown in Table 1. Please see Appendix A for detailed descriptions. We further extend this taxonomy with three widely used prompting behaviors commonly observed in practical LLM usage: **Vanilla**, which uses direct task instructions without additional behavioral constraints; **Demonstration (Demo)**, which uses task example-based few-shot prompting; and **Chain-of-Thought (CoT)**, which provides reasoning guidance through step-by-step decomposition. Full experiment setting can be found in Appendix B.

Let $\mathcal{P} = \{p_1, \ldots, p_K\}$ denote the set of prompt

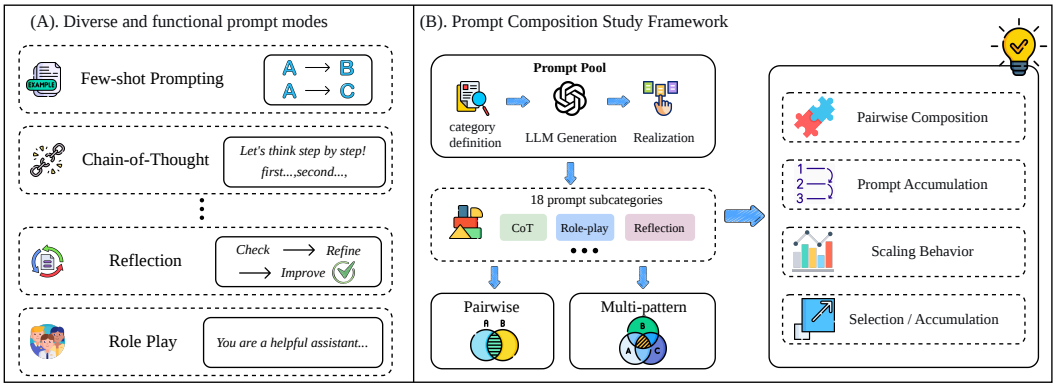

Figure 2: An overview of our study. We investigate how prompt patterns interact when composed, and how such composition affects LLM performance across tasks, models, and composition strategies.

patterns considered in our study. Each pattern $p_i \in \mathcal{P}$ is associated with a set of concrete realizations $\mathcal{R}(p_i)$. A concrete realization r_i is sampled as

$$r_i \sim \mathcal{R}(p_i). \tag{1}$$

For each prompt pattern, we construct multiple realizations and average results across realizations to reduce sensitivity to specific wording choices.

2.2 Prompt Composition

Given a task input x, a prompt composition is defined as an ordered sequence of prompt patterns:

$$C = (p_1, p_2, \ldots, p_k), \tag{2}$$

where k denotes the number of composed prompt patterns.

Each pattern p_i is instantiated by a realization $r_i \sim \mathcal{R}(p_i)$, and the realized prompt corresponding to composition C is constructed as

$$R(C) = r_1 \oplus r_2 \oplus \cdots \oplus r_k, \tag{3}$$

where $\oplus$ denotes textual concatenation.

Given a language model M, the model output under composition C is

$$\hat{y} = M(R(C) \oplus x). \tag{4}$$

We define the task performance of a composition C on task t as

$$f_t(C), \tag{5}$$

where f_t denotes the expected evaluation score under the task-specific metric. Single-pattern prompting is treated as a special case with $|C| = 1$, while pairwise and multi-pattern compositions correspond to $|C| = 2$ and $|C| > 2$, respectively.

Our goal is not only to identify high-performing prompt patterns, but to characterize how prompt behaviors interact under composition. In particular, we study whether prompt composition exhibits additive gains, order sensitivity, interference effects, and scaling behavior as more prompt patterns are accumulated.

3 Can Effective Prompt Strategies Be Reliably Combined?

A common assumption behind practical prompt composition is that beneficial prompting strategies should accumulate their advantages: if two prompt patterns each improve model performance individually, then combining them should produce an even stronger prompt. In this section, we test this compositionality assumption through controlled pairwise prompt composition experiments.

Following Section 2, each prompt pattern p denotes a reusable prompting behavior, and $f_t(C)$ denotes the task performance obtained by applying composition C on task t. For readability, we write $f_t(p)$ for single-pattern prompting and $f_t(p, q)$ for ordered pairwise composition.

We conduct controlled pairwise composition experiments by evaluating both ordered prompt pairs (p, q) and (q, p). Concrete prompt realizations are concatenated into the system prompt, while the task instance is provided through the user prompt. Aggregated pairwise results for additional models are reported in Appendix C, and the full ordered pairwise interaction matrices for additional main-text models are provided in Appendix D.

For a single prompt pattern, we define its gain over vanilla as

$$\Delta_t^{\text{single}}(p) = f_t(p) - f_t(\text{Vanilla}). \tag{6}$$

	Vanilla	CoT	Persona	CtxMgr	Recipe	Demo	Gamify	AltApp	Reflect	RefBrk	InfGen	Verify	MetaLan	VisGen	FactChk	OutputAuto	Template	Q-Ref	Flip
Vanilla	51.67± 0.95	52.83± 0.73	52.70± 0.70	52.69± 1.09	52.60± 1.23	51.89± 0.92	51.12± 3.22	50.36± 1.13	50.12± 1.16	49.85± 0.74	48.91± 1.43	47.35± 0.64	47.18± 2.28	46.49± 1.20	44.60± 2.38	43.97± 2.35	42.52± 2.52	41.97± 7.25	24.23± 11.34
CoT	52.83± 0.73	52.83± 0.73	52.49± 1.73	54.35± 1.43	53.07± 1.42	53.38± 1.97	48.14± 1.25	48.01± 0.40	49.60± 0.88	50.56± 0.61	49.39± 0.41	48.22± 0.39	49.57± 0.87	47.43± 2.34	42.83± 1.87	45.56± 0.46	43.47± 3.96	46.87± 2.00	29.07± 1.95
Persona	52.70± 0.70	49.79± 2.08	52.70± 0.70	54.27± 1.29	53.68± 1.06	52.54± 1.01	51.58± 2.96	52.00± 1.11	44.22± 1.84	51.19± 1.28	48.04± 1.37	48.08± 1.78	49.10± 1.82	46.35± 1.16	42.09± 2.11	45.70± 2.07	39.67± 1.85	42.61± 3.40	32.86± 2.14
CtxMgr	52.69± 1.09	51.34± 2.52	54.27± 0.15	52.69± 1.09	54.51± 1.02	54.39± 0.93	52.10± 3.16	48.34± 0.34	49.05± 1.27	53.44± 1.64	48.62± 0.72	51.97± 0.78	50.05± 2.40	48.59± 1.28	46.87± 1.94	46.36± 1.89	38.99± 1.20	47.62± 1.05	30.20± 4.16
Recipe	52.60± 1.23	54.42± 0.81	53.33± 2.11	55.32± 1.83	52.60± 1.23	53.86± 1.03	51.10± 2.84	50.48± 1.56	48.89± 1.89	51.03± 0.52	48.62± 1.06	47.58± 1.77	52.11± 3.52	47.21± 2.82	42.57± 1.45	45.50± 0.57	45.23± 2.39	46.48± 1.48	34.27± 0.35
Demo	51.89± 0.92	54.08± 0.17	53.19± 1.10	54.59± 1.15	53.09± 0.64	51.89± 0.92	51.05± 3.82	50.64± 1.39	49.21± 0.53	50.89± 2.01	48.29± 2.44	50.58± 0.73	50.83± 0.27	45.58± 1.71	42.94± 1.65	45.29± 1.75	51.40± 1.13	44.30± 2.26	27.98± 2.54
Gamify	51.12± 3.22	53.52± 3.02	50.47± 3.55	53.42± 4.35	53.11± 2.36	49.74± 3.16	51.12± 3.22	48.29± 3.28	44.55± 3.15	50.56± 2.65	48.61± 0.44	49.11± 2.10	48.24± 3.85	45.35± 1.26	41.00± 4.28	45.65± 1.71	40.87± 3.30	40.74± 3.48	30.55± 1.92
AltApp	50.36± 1.13	50.95± 0.58	48.55± 0.92	49.48± 1.61	51.17± 0.71	51.13± 0.16	50.93± 4.82	50.36± 1.13	47.28± 0.45	48.62± 1.35	47.76± 0.33	47.20± 0.99	45.48± 4.17	48.55± 0.88	42.68± 2.02	41.95± 3.20	38.99± 1.99	43.07± 2.00	26.23± 5.64
Reflect	50.12± 1.16	50.02± 1.72	49.09± 1.79	50.51± 1.25	47.78± 1.37	50.11± 1.31	45.74± 0.28	47.85± 1.37	50.12± 1.16	48.33± 0.07	47.57± 0.66	47.92± 1.08	44.57± 1.30	44.15± 1.45	43.47± 3.21	41.83± 1.56	42.84± 3.93	44.19± 2.93	31.32± 1.53
RefBrk	49.85± 0.74	49.69± 0.90	48.77± 0.27	52.44± 0.95	49.54± 1.02	50.86± 2.03	48.17± 4.29	47.72± 1.18	46.92± 2.70	49.85± 0.74	47.25± 1.87	48.17± 1.93	44.53± 2.83	48.13± 1.22	40.79± 2.00	46.13± 2.52	39.50± 2.61	44.68± 1.67	28.91± 1.64
InfGen	48.91± 1.43	48.12± 1.02	49.62± 1.98	48.97± 0.90	48.56± 2.22	48.52± 1.05	47.50± 1.10	47.56± 1.18	45.24± 2.52	47.48± 1.88	48.91± 1.43	45.89± 2.63	41.65± 5.66	45.05± 2.76	39.89± 1.86	44.27± 1.74	38.30± 2.94	44.31± 3.14	30.06± 3.99
Verify	47.35± 0.64	49.79± 1.37	49.17± 2.04	51.54± 1.77	47.25± 0.54	52.00± 1.17	47.58± 3.23	47.70± 0.41	47.31± 0.69	47.76± 0.65	47.51± 0.43	47.35± 0.64	46.65± 1.27	45.95± 0.70	42.56± 1.66	44.13± 2.49	39.52± 1.59	45.65± 1.97	21.18± 4.01
MetaLan	47.18± 2.28	45.26± 2.12	49.31± 1.10	46.37± 2.91	47.82± 4.23	48.72± 1.28	47.98± 4.65	45.78± 4.26	42.32± 4.23	46.69± 2.22	42.88± 5.06	40.88± 4.93	47.18± 2.28	41.91± 4.71	42.29± 3.02	39.71± 6.63	38.08± 1.84	35.66± 6.47	22.55± 4.75
VisGen	46.49± 1.20	45.73± 0.93	47.16± 0.82	47.87± 1.95	45.07± 2.80	49.00± 3.56	46.82± 2.21	45.03± 0.79	44.43± 0.28	45.61± 2.93	42.18± 1.76	45.84± 1.79	43.69± 2.46	46.49± 1.20	40.65± 0.94	40.69± 0.54	45.24± 1.64	40.45± 2.78	31.61± 2.13
FactChk	44.60± 2.38	43.45± 1.73	43.47± 2.55	46.11± 2.85	43.25± 2.21	49.47± 1.41	40.99± 1.81	39.31± 2.45	42.72± 4.11	44.98± 1.82	47.55± 0.64	42.03± 2.12	38.42± 0.59	42.85± 3.42	44.60± 2.38	40.36± 3.31	39.90± 2.02	42.52± 1.83	34.13± 5.29
OutputAuto	43.97± 2.35	43.36± 2.85	45.38± 0.26	46.96± 0.86	45.15± 1.51	47.88± 2.05	42.18± 2.80	46.43± 2.12	47.18± 1.17	45.49± 1.91	44.30± 1.31	42.58± 1.27	40.87± 0.95	43.02± 2.54	36.97± 2.90	43.97± 2.35	34.97± 2.01	37.67± 5.94	24.90± 6.29
Template	42.52± 2.52	43.32± 3.76	46.85± 0.54	47.48± 1.62	42.00± 3.32	47.84± 1.18	41.13± 2.27	43.61± 4.05	41.37± 1.26	43.94± 2.46	44.25± 4.35	41.27± 3.81	35.39± 1.22	43.81± 2.99	37.64± 1.51	40.61± 3.37	42.52± 2.52	41.90± 2.49	30.36± 0.72
Q-Ref	41.97± 7.25	37.46± 5.16	46.53± 0.55	50.06± 1.94	45.70± 2.04	46.32± 4.06	39.68± 2.33	42.61± 3.00	42.73± 2.60	40.43± 0.66	39.77± 3.06	44.69± 2.60	40.44± 4.19	39.15± 0.71	41.83± 4.18	36.34± 4.17	37.96± 2.76	41.97± 7.25	15.49± 3.03
Flip	24.23± 11.34	27.73± 7.87	36.60± 1.95	33.83± 6.30	16.49± 1.97	44.22± 4.56	35.43± 6.62	28.48± 1.99	31.24± 3.82	33.27± 4.59	29.39± 3.52	15.61± 6.14	32.49± 1.19	30.10± 1.28	24.74± 3.84	27.32± 2.37	19.36± 5.96	27.38± 7.71	24.23± 11.34

Table 2: Averaged pairwise score of Llama-3.1-8B across 6 tasks. Diagonal entries correspond to single-prompt runs; Off-diagonal entries correspond to ordered two-prompt compositions. Red indicates the performance degradation.

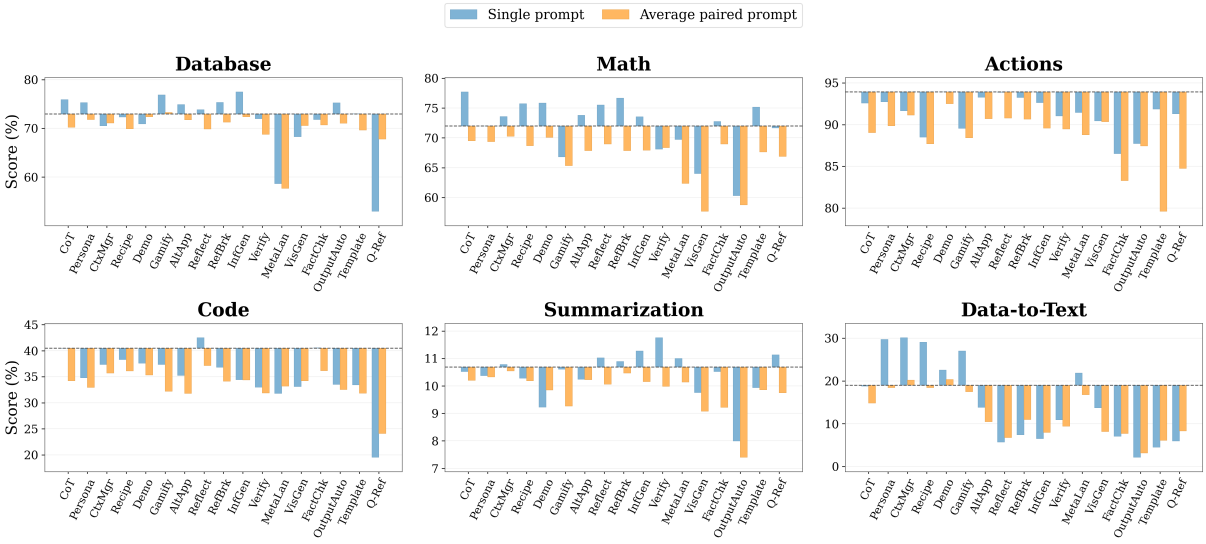

Figure 3: Single-prompt and average pairwise-composition performance over Vanilla on Llama-3.1-8B-Instruct across six task categories. Blue bars show the single prompt pattern score, orange bars show the averaged pairwise prompt score.

To measure how p behaves under pairwise composition, we first average over the two possible prompt orders:

$$s_t(p, q) = \frac{1}{2}\left[f_t(p, q) + f_t(q, p)\right]. \quad (7)$$

For a candidate prompt set $\mathcal{Q}$ without prompt p. We then define the average paired-prompt score of p as

$$F_t^{\text{pair}}(p; \mathcal{Q}) = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} s_t(p, q). \quad (8)$$

This score averages both prompt orders, so it does not rely on placing p only in its favorable position. We further define the composition interference score:

$$\Delta_t(p; \mathcal{Q}) = F_t^{\text{pair}}(p; \mathcal{Q}) - f_t(p). \quad (9)$$

Observation 1: Prompt effectiveness is highly task-dependent. Figure 3 shows that prompt effectiveness is highly task-dependent. While some prompt patterns substantially improve performance on specific tasks, the same patterns are often neutral or harmful on others. This suggests that prompt patterns do not behave as universally beneficial modules, but instead introduce task-specific behavioral biases that may or may not align with a given task.

Observation 2: Pairwise prompt composition usually reduces performance. Across models and tasks, fewer than 10% of pairwise prompt compositions outperform their stronger constituent prompt. Across most prompt patterns, the averaged pairwise score is lower than the corresponding single-prompt score, indicating $\Delta_t(p; \mathcal{P}) < 0$. Full results of Llama-3.1-8B are shown in Table 2.

These results suggest that prompt composition

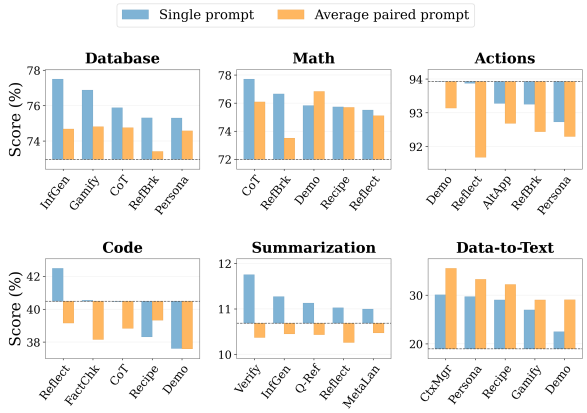

Figure 4: Pairwise composition among the top-5 single-prompt strategies for each task.

is highly non-additive: adding a second prompt pattern often introduces competing behavioral constraints, redundancy, or instruction interference that dilute the useful behavior of the original prompt.

Observation 3: Individually strong strategies can still be compositionally incompatible. One concern is that the negative average in Observation 2 may be driven by individually weak prompt patterns. To test this possibility, for each task we select the top-5 non-vanilla single-prompt patterns according to $f_t(p)$, forming a task-specific set $\mathcal{P}_t^{\text{top}}$, and recompute the paired score using only top-performance candidate set $\mathcal{Q}_t^{\text{top}} = \mathcal{P}_t^{\text{top}} \setminus \{p\}$:

$$F_t^{\text{pair}}(p; \mathcal{Q}_t^{\text{top}}) = \frac{1}{|\mathcal{Q}_t^{\text{top}}|} \sum_{q \in \mathcal{Q}_t^{\text{top}}} s_t(p, q). \quad (10)$$

Figure 4 reports this top-5 composition setting.

Even in this favorable setting, composition often fails to preserve single-prompt gains. In Database, Actions, and Summarization, every top prompt pattern becomes worse on average after being paired with other top prompt patterns. In Code and Math, most top prompt patterns also lose performance.

Data-to-Text is the main exception, suggesting that prompt composition can be beneficial when prompt patterns impose complementary rather than competing behavioral constraints.

Thus, the strong single-prompt pattern does not imply compositional compatibility. Even individually strong prompt patterns frequently fail to combine constructively, suggesting that prompt quality itself is fundamentally non-compositional.

Takeaway. Overall, pairwise prompt composition is highly non-additive. Even individually strong prompt patterns frequently become compositionally incompatible when combined, suggesting that prompt behaviors interact through redundancy, behavioral competition, and instruction interference rather than simple capability accumulation.

4 Does Prompt Accumulation Scale?

Although pairwise prompt composition rarely yields strong additive improvements, larger prompt compositions may still exhibit emergent behaviors beyond local pairwise interactions. We therefore investigate whether prompt accumulation exhibits scaling-like behavior as more prompt patterns are progressively composed.

To study this question, we incrementally construct prompt chains through greedy extension and compare the resulting trajectories against a local pairwise additive prediction.

Greedy extension starting from the best-performing single prompt pattern, iteratively appending the next prompt pattern that yields the largest pairwise gain with the current chain.

Algorithm 1 Greedy Prompt Chain Construction

Require: Prompt pool $\mathcal{P}$, composition score $f_t(C)$, maximum chain length K
Ensure: Prompt chain S_K, actual trajectory $\mathcal{A}$

```
p_1 ← arg max_{p∈P} f_t(p)
S_1 ← [p_1]
A ← [f_t(S_1)]
P ← P \ {p_1}
for k = 1 to K − 1 do
    p_{k+1} ← arg max_{p∈P} f_t(S_k ⊕ p)
    S_{k+1} ← S_k ⊕ p_{k+1}
    A ← A ∪ {f_t(S_{k+1})}
    P ← P \ {p_{k+1}}
end for
return S_K, A
```

To examine whether higher-order composition behavior can be approximated through local interactions, we further introduce a local pairwise additive approximation. Instead of evaluating all higher-order interactions directly, this approximation predicts the scaling trajectory using only adjacent pairwise gains within the constructed prompt chain.

We evaluate these accumulation trajectories across three model families and repeat each chain construction three times with independently sampled prompt realizations. We report averaged tra-

Algorithm 2 Local Pairwise Additive Prediction

Require: Prompt chain $S_K = (p_1, \ldots, p_K)$
Ensure: Additive prediction trajectory $\hat{\mathcal{A}}$

$\hat{a}_1 \leftarrow f_t(p_1)$
$\hat{\mathcal{A}} \leftarrow [\hat{a}_1]$
for $k = 1$ to $K - 1$ **do**
  $g_k \leftarrow f_t(p_k, p_{k+1}) - f_t(p_k)$
  $\hat{a}_{k+1} \leftarrow \hat{a}_k + g_k$
  $\hat{\mathcal{A}} \leftarrow \hat{\mathcal{A}} \cup \{\hat{a}_{k+1}\}$
end for
return $\hat{\mathcal{A}}$

jectories to reduce sensitivity to prompt wording and sampling noise.

Observation 1: Prompt accumulation yields only limited early-stage gains. Figure 5 shows that greedy prompt accumulation can produce modest early-stage gains, but performance quickly saturates. Most trajectories reach a local peak after composing only 4–5 prompt patterns, after which additional accumulation yields only weak fluctuations or gradual degradation.

Even in this favorable greedy setting, sustained scaling-like gains fail to emerge, suggesting that prompt accumulation is structurally constrained rather than merely caused by prompt selection.

Observation 2: Local pairwise gains fail to explain long prompt compositions. During the early accumulation stages, pairwise additive predictions closely match the actual trajectories, indicating that prompt composition initially behaves in an approximately additive manner. However, the agreement weakens as compositions grow longer, and additive estimates increasingly overestimate actual performance after approximately 4–5 prompt patterns.

These results suggest that prompt accumulation cannot be fully explained through local pairwise interactions alone. As prompt chains grow longer, higher-order interference and diminishing marginal effects gradually dominate accumulation behavior.

We further test whether accumulation becomes more effective when restricted to repeatedly composing only the strongest-performing prompt patterns. Specifically, we construct repeated strong-prompt accumulation by greedily composing only the top-performing prompt patterns while avoiding adjacent duplicates.

Even in this favorable setting, sustained scaling-like gains still fail to emerge. Some models exhibit small early-stage improvements before saturating, while others remain nearly flat throughout accumulation. Interestingly, close agreement between additive estimates and actual trajectories does not necessarily imply successful prompt scaling. In some cases, the curves remain aligned primarily because pairwise compositions themselves already provide little improvement, causing both trajectories to stay nearly flat.

Takeaway. Overall, prompt accumulation exhibits only limited local additivity and fails to produce sustained scaling-like gains. Although greedy extension and strong-prompt reuse can yield modest early improvements, performance quickly saturates as prompt chains grow longer, while pairwise additive predictions increasingly diverge from actual behavior. These results suggest that prompt accumulation is fundamentally constrained by higher-order interference, redundancy, and diminishing marginal behavioral influence.

We therefore next investigate whether multiple prompt patterns can be utilized more effectively through alternative composition mechanisms.

5 How Should Multiple Prompt Strategies Be Utilized?

The previous sections show that directly accumulating prompt patterns often introduces interference and quickly saturates. We therefore investigate whether multiple prompt strategies can be utilized more effectively through alternative composition mechanisms.

To study this question, we compare four representative prompting mechanisms that differ in how prompt patterns are incorporated, coordinated, and selected during inference. Some mechanisms directly accumulate multiple prompt patterns into a shared context, while others treat prompt patterns as competing candidate strategies and defer selection until inference time.

- **Concatenation.** Concatenation directly merges multiple prompt patterns into a single instruction context.

- **Multi-turn Composition.** Multi-turn composition injects prompt patterns sequentially across conversational turns before presenting the final task.

Both mechanisms rely on direct prompt accumulation and therefore assume that multiple prompt-

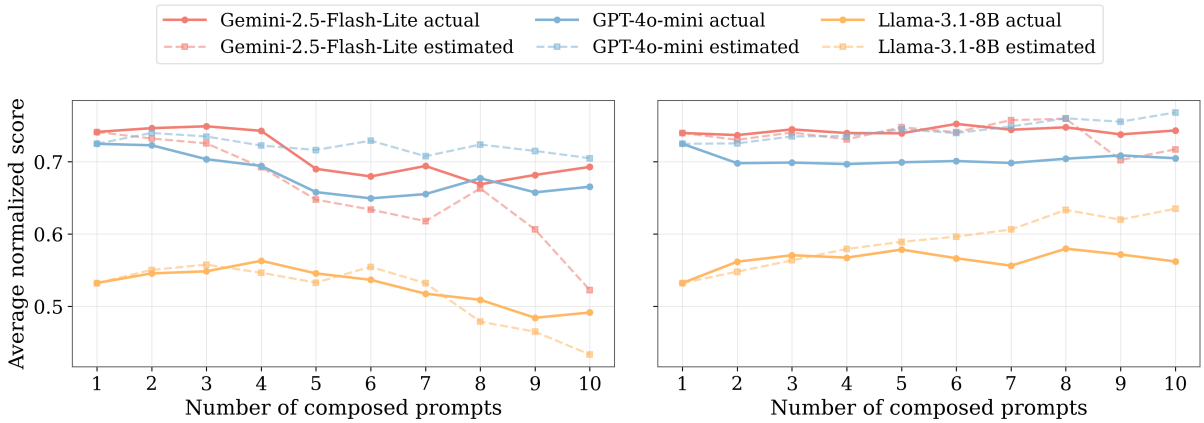

Figure 5: Scaling behavior of prompt accumulation under diverse, non-duplicated composition (left) and repeated strong-prompt composition (right). Solid lines denote actual greedy composition trajectories, while dashed lines denote local pairwise additive approximations.

	CONCATENATION						CHOICE-BASED						BEST-OF-N						MULTI-TURN					
Model	Act	Code	D2T	DB	Math	Sum	Act	Code	D2T	DB	Math	Sum	Act	Code	D2T	DB	Math	Sum	Act	Code	D2T	DB	Math	Sum
Llama-3.1-8B	94.3	31.0	**41.5**	72.9	73.8	10.6	92.4	32.3	38.2	67.0	76.7	10.6	**96.0**	**46.2**	29.6	**93.4**	**82.5**	**11.8**	91.2	35.5	8.6	72.1	75.2	6.8
Gemini-2.5-Flash-Lite	**100.0**	89.9	34.6	88.8	90.3	26.3	94.1	90.0	**49.9**	89.7	**91.3**	27.5	88.9	91.8	49.8	**93.5**	88.2	**28.7**	95.7	**96.0**	47.3	93.5	89.3	27.7
Gemini-2.5-Pro	**100.0**	**100.0**	55.6	93.5	97.1	32.8	98.7	98.6	**57.6**	93.5	98.0	**35.7**	97.3	100.0	54.8	98.1	90.1	34.4	99.0	100.0	53.0	**99.1**	**98.1**	28.4
GPT-4o-mini	93.3	75.0	48.0	86.9	90.3	14.9	90.4	**79.0**	**48.4**	**91.6**	87.3	15.8	**93.5**	78.0	43.0	90.7	88.1	**16.8**	90.4	77.6	45.1	89.5	**90.6**	14.5
GPT-4o	97.5	85.0	51.4	93.5	**99.0**	25.1	96.9	**91.0**	**54.3**	92.5	90.3	24.7	96.1	88.8	42.6	**96.3**	90.3	**25.7**	**100.0**	61.0	43.1	92.5	87.4	8.8
Claude-Sonnet-4.5	**97.0**	97.0	51.7	94.4	98.1	31.7	96.2	97.0	**56.1**	97.2	98.1	36.2	96.6	**98.0**	53.9	95.3	96.1	**36.8**	95.6	96.0	53.4	**98.1**	**100.0**	30.0
Grok-4.1-Fast	**95.2**	92.0	56.7	95.3	98.0	30.1	91.5	92.0	56.3	94.4	**99.0**	29.4	92.1	**93.9**	56.2	**96.3**	96.1	29.8	94.9	92.0	**57.8**	95.3	96.1	**30.3**
DeepSeek-v4-flash	96.0	71.1	**58.2**	92.5	85.4	19.8	97.0	77.3	57.4	91.6	**92.2**	23.9	**98.1**	**85.4**	51.7	**95.3**	92.2	**32.6**	96.6	76.0	57.0	91.6	87.3	6.9

Table 3: Comparison of different prompt utilization mechanisms across tasks and model families. Selection-based mechanisms consistently outperform direct accumulation mechanisms.

ing behaviors can be jointly maintained within the same interaction context.

- **Choice-based Composition.** Choice-based composition treats prompt patterns as alternative candidate strategies. Rather than enforcing all prompt patterns simultaneously, the model first selects the most suitable prompting strategy for the current task and then generates the response conditioned on the selected prompt pattern.

- **Best-of-N Composition.** Best-of-N composition independently generates responses from different prompt patterns and uses a judge model to select the final answer. Unlike direct accumulation, prompt patterns compete through response-level selection rather than coexistence within a shared prompt context.

Unlike accumulation-based mechanisms, these approaches avoid simultaneously enforcing multiple prompt behaviors within the same interaction context.

Observation 1: Selection-based mechanisms outperform direct accumulation. Across tasks and models, selection-based approaches consistently outperform accumulation-based mechanisms. Best-of-N achieves the highest overall win rate (39.2%), followed by Choice-based composition (25.05%), while Multi-turn composition and Concatenation achieve only 20.5% and 15.3%, respectively.

This trend aligns with the previous sections: directly accumulating multiple prompt patterns often introduces redundancy and instruction interference, while selection-based methods avoid forcing multiple prompting behaviors to coexist within the same context.

Interestingly, Choice-based composition already provides substantial improvements despite requiring only a single final generation step, suggesting that avoiding simultaneous prompt coexistence itself is highly beneficial.

Observation 2: Multi-turn accumulation does not eliminate interference. Although Multi-turn composition generally improves over naive Concatenation, it still consistently underperforms selection-based mechanisms. This suggests that distributing prompt accumulation across conversational turns may alleviate some interference, but is insufficient to fully resolve conflicts between prompting behaviors.

Takeaway. Overall, mechanisms that externalize or defer prompt selection are substantially more robust than mechanisms that directly accumulate multiple prompt patterns into a shared context. Taken together with the previous sections, these results suggest that effective multi-prompt usage depends more on selecting compatible prompting behaviors than on accumulating increasingly complex prompt compositions.

6 Related Work

6.1 Prompt Patterns and Behavioral Abstractions

Prompt engineering has become a widely used interface for adapting large language models without updating model parameters. Prior work has shown that prompting strategies such as few-shot prompting (Brown et al., 2020), reasoning guidance (Wei et al., 2022), reflection (Shinn et al., 2023), verification, and structured output control (Schick and Schütze, 2021) can substantially affect model behavior and task performance. Beyond studying individual prompts, several works organize prompting practices into reusable behavioral abstractions. White et al. (2023) introduce a prompt-pattern catalog that abstracts recurring prompting practices such as persona assignment, template formatting, reflection, and question refinement into reusable prompt patterns. Schulhoff et al. (2024) further systematize prompting techniques and terminology across a broad range of LLM applications. These works motivate treating prompts not merely as isolated strings, but as reusable behavioral patterns.

Rather than evaluating prompt patterns in isolation, we investigate how prompting behaviors interact under direct composition and whether their benefits reliably accumulate.

6.2 Prompt Robustness and Prompt Interaction

Recent work suggests that prompt effectiveness is highly sensitive to wording, formatting, and evaluation setup. Sclar et al. (2024) show that seemingly minor prompt-formatting variations can substantially affect LLM performance, while Mizrahi et al. (2024) demonstrate that conclusions drawn from single-prompt evaluation may be unstable and advocate multi-prompt evaluation protocols.

However, prior work primarily studies robustness under prompt variation rather than interactions between multiple functional prompting behaviors. Our work instead focuses on prompt composition as an interaction problem, examining whether multiple prompt patterns preserve, amplify, or interfere with one another when accumulated within the same context.

6.3 Multi-Prompt Utilization and Instruction Following

Prior work shows that multiple prompts can be beneficial when used as alternative candidates rather than accumulated instructions. Dipper constructs diverse prompt ensembles for reasoning tasks (Hu et al., 2025), while APE searches over candidate natural-language instructions to optimize task performance (Zhou et al., 2023). Few-shot prompting is also sensitive to prompt format, example choice, and example order (Zhao et al., 2021), suggesting that prompt effectiveness depends on selection and configuration rather than simply adding more instructions.

Our selection-based mechanisms are also related to LLM-as-a-judge methods, where a model compares or selects among candidate generations (Zheng et al., 2023). Another related direction is complex instruction following: ComplexBench evaluates whether LLMs can satisfy instructions composed of multiple constraints (Wen et al., 2024). In contrast, we study whether reusable prompt patterns yield additive task-performance gains under composition. Overall, prior work has studied prompt design, calibration, optimization, judging, and multi-prompt inference, but has not systematically examined whether prompt patterns support scalable accumulation.

7 Conclusion

In this work, we study prompt composition as an empirical interaction problem rather than a simple accumulation process. Across models and tasks, we found that prompt composition is highly non-additive: pairwise prompt compositions rarely outperform their stronger constituent prompts, prompt accumulation quickly saturates as compositions grow longer, and selection-based mechanisms consistently outperform direct accumulation strategies.

Prompt engineering behaves more like constrained strategy selection. Effective multi-prompt usage is therefore achieved not by simply stacking more instructions, but by selecting compatible prompting behaviors while avoiding instruction interference.

Limitations

First, although we evaluate multiple model families and tasks, the experiments are conducted on a limited set of instruction-tuned general-purpose LLMs. We therefore avoid making strong claims about model scale or universally consistent composition behavior across architectures. In addition, our experiments cover only six task categories, and prompt composition may exhibit substantially different behaviors on other tasks or domains.

Second, our prompt compositions are constructed from predefined prompt patterns and manually designed realizations. While this controlled setting enables systematic analysis, real-world prompting behavior can be considerably more diverse and adaptive.

Third, our analysis primarily focuses on performance-based composition effects rather than internal mechanistic explanations. Although the observed saturation and interference patterns suggest higher-order behavioral interactions, we do not directly analyze the underlying attention dynamics or representation conflicts responsible for these effects.

Finally, our work studies prompt composition mainly in static prompting settings. More advanced paradigms, such as tool-augmented agents, retrieval-based systems, or adaptive multi-agent prompting, may exhibit different composition behaviors and remain important directions for future research.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2025. System card: Claude Sonnet 4.5. https://www.anthropic.com/claude-sonnet-4-5-system-card. Anthropic System Card.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, and 1 others. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems, 2021. *URL https://arxiv.org/abs/2110.14168*, 9.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

DeepSeek-AI. 2026. DeepSeek V4 preview release. https://api-docs.deepseek.com/news/news260424. Official release note.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Wenyang Hu, Gregory Kang Ruey Lau, Liu Diwen, Chen Jizhuo, See Kiong Ng, and Bryan Kian Hsiang Low. 2025. Dipper: Diversity in prompts for producing large language model ensembles in reasoning tasks. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 35546–35560.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2025. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pages 58791–58831.

Aobo Kong, Shiwan Zhao, Hao Chen, Qicheng Li, Yong Qin, Ruiqi Sun, Xin Zhou, Enzhi Wang, and Xiaohang Dong. 2024. Better zero-shot reasoning with role-play prompting. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4099–4113.

Philippe Laban, Alexander Richard Fabbri, Caiming Xiong, and Chien-Sheng Wu. 2024. Summary of a haystack: A challenge to long-context llms and rag systems. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9885–9903.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2025. Llms get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys*, 55(9):1–35.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. 2024. State of what art? a call for multi-prompt llm evaluation. *Transactions of the Association for Computational Linguistics*, 12:933–949.

OpenAI. 2024. GPT-4o mini: Advancing cost-efficient intelligence. OpenAI Blog.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. Totto: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. 2025. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th conference of the European chapter of the association for computational linguistics: main volume*, pages 255–269.

Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, and 1 others. 2024. The prompt report: A systematic survey of prompt engineering techniques. *arXiv preprint arXiv:2406.06608*.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *International Conference on Learning Representations*, volume 2024, pages 25055–25083.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, and 1 others. 2024. Benchmarking complex instruction-following with multiple constraints composition. *Advances in Neural Information Processing Systems*, 37:137610–137645.

Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. 2023. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*.

xAI. 2025. Grok 4 Fast model card. https://data.x.ai/2025-09-19-grok-4-fast-model-card.pdf. XAI Model Card.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, and 1 others. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 3911–3921.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. In *International Conference on Learning Representations*.

A Example Prompts for each prompt pattern

The purpose of `General Assistance`, referred to as `Vanilla` in this study, is to serve as the default system prompt for handling general-purpose tasks.

Vanilla

You are a helpful assistant.

`Provide Examples`, commonly known as `few-shot prompting`, involves supplying concrete examples based on the task context. In this study, we use three examples (i.e., three-shot prompting).

Demo

{Example1}

{Example2}

{Example3}

`Meta Language Generation`, abbreviated as `MetaLan`, involves first defining a meta-language that may carry special meanings for large language models (LLMs).

MetaLan

When I denote functions with 'f[x] -> y', it implies that I am stating a function f that maps x to y. For example, 'f[3] -> 9' means 'f maps 3 to 9', indicating a functional relationship.

`Output Automator`, abbreviated as `OutputAuto`, have the LLM generate a script or other automation artifact that can automatically perform any steps it recommends taking as part of its output.

OutputAuto

Create an executable script that can automatically derive mathematical expressions and evaluate them for different variable values. Ensure the script includes functionality for both symbolic and numerical computation.

`Persona` is used to simulate a specific role or character within the conversation, enabling the model to respond in a manner consistent with that persona.

Persona

From now on, act as a mathematics professor. Focus on explaining complex mathematical concepts in a clear and detailed manner, providing outputs that a mathematics professor would generate to facilitate learning.

`Visualization Generator`, abbreviated as `VisGen`, is used to encourage LLMs to produce visual representations of their outputs, serving as a prompt signal for the model to take physical reasoning into account.

VisGen

Please create a Graphviz Dot file that visually represents the prime factorization tree of the number 120. This should show the breakdown of 120 into its simplest prime factors.

`Recipe` provides constraints to ultimately output a sequence of steps given some partially provided "ingredients" that must be configured in a sequence of steps to achieve a stated goal.

Recipe

I am trying to solve a complex integration problem. I know that I need to identify the type of integration and apply integration by parts. Please provide a complete sequence of steps. Please fill in any missing steps. Please identify any unnecessary steps.

`Template` is used to ensure that the LLM's output adheres to a specific structure or format.

Template

I am going to provide a template for your output. Fit your calculation and reasoning into the placeholders that I list. Please preserve the formatting and overall template that I provide: Solve the equation X and write the answer in the form Solved: SOLUTION, with steps detailed at STEPS.

`Fact Check List`, abbreviated as `FactChk`, ensures that the LLM outputs a list of key facts contained in its response, highlighting the factual elements that form the basis of its statements.

FactChk

When solving a complex geometry problem, list the assumptions and theorems used and verify if each of these is correct in the context of the problem.

`Reflection`, abbreviated as `Reflect`, ask the model to automatically explain the rationale behind given answers to the user.

Reflect

When you solve a math problem, please explain the reasoning and assumptions behind each step you take to reach the solution. If possible, use specific examples or mathematical theorem to support your strategy. Additionally, please highlight any potential ambiguities in your solution, to provide a comprehensive response.

`Question Refinement`, abbreviated as `Q-Ref`, involves engaging the LLM in the prompt engineering process by encouraging it to suggest improved or more precise versions of the user's original question.

Q-Ref

Whenever I ask a question about solving differential equations, suggest a better version of the question that incorporates specific methods suitable for the type of differential equation I am dealing with and ask me if I would like to use your refined question instead.

`Alternative Approaches`, abbreviated as `AltApp`, ensures that the LLM always presents multiple ways to accomplish a task, encouraging users to consider options beyond those they are already familiar with.

AltApp

When asked to solve a system of linear equations, provide alternative methods such as substitution, elimination, and matrix inversion. Compare the pros and cons of each method with respect to ease of computation and real-world applicability. Include the initial method requested, and then ask which method the user would like to choose.

`Cognitive Verifier`, abbreviated as `Verify`, force the LLM to always subdivide questions into additional questions that can be used to provide a better answer to the original question.

Verify

When I present a math problem to you, please generate three additional sub-questions that will help in solving the problem accurately. Once I provide answers to these sub-questions, use them to construct the solution to the original math problem.

`Refusal Breaker`, abbreviated as `RefBrk`, prompts the LLM to automatically assist users in rephrasing their questions when it initially refuses to respond.

RefBrk

If you are unable to solve a math problem, please explain why and suggest one or more alternative ways to phrase the problem that might lead to a solution.

`Flipped Interaction`, abbreviated as `Flip`, allows the LLM to ask questions in order to gather the necessary information for completing a task.

Flip

I'd like you to ask me questions to determine the steps needed to solve a quadratic equation. Once you have enough information, provide a step-by-step solution.

`Game Play`, abbreviated as `Gamify`, involves creating a game centered around a specific topic.

Gamify

We are going to play a mathematical adventure game. You will act as a math genie trapped in an ancient book of arithmetic. My task as a player is to solve math riddles you present to help free you from the book. The game will follow simple rules: I must successfully solve a sequence of math riddles where each correct answer brings us closer to your release, with each riddle focusing on a different mathematical concept such as addition, subtraction, multiplication, division, and basic algebra. To start the game, provide the first riddle and a clue related to ancient wisdom or a famous mathematician.

`Infinite Generation`, abbreviated as `InfGen`, automatically generates a continuous series of outputs—seemingly infinite—without the need to re-enter the generator prompt after each iteration.

InfGen

From now on, generate mathematical sequences based on the input formula I provide until I say stop. Use the format: SEQUENCE_START={start}, FORMULA={formula}, NEXT_TERMS=[terms]. Each sequence start and formula are placeholders, and you should fill in these whenever you receive a new input.

`Context Manager`, abbreviated as `CtxMgr`, enable users to specify or remove context for a conversation with an LLM.

CtxMgr

While solving the following mathematical problems, only consider theorems and formulas relevant to calculus.

B Experimental Setup and Evaluation Details

Objective of Prompt Composition. Given a model, a task, and a prompt composition, our objective is to evaluate how the composed prompt affects model performance. For each task instance, the model receives the composed prompt together with the task input, and generates a response that is evaluated by the task-specific metric.

We average instance-level scores to compare individual prompt patterns, pairwise compositions, and multi-pattern compositions. Our goal is not only to identify which prompt performs best, but also to characterize how performance changes with pattern selection, prompt ordering, model family, task type, and composition length. Accordingly, we treat prompt composition as the primary object of evaluation.

Datasets and Tasks. We follow the benchmark suite and evaluation protocol introduced by Laban et al. (2025), which consists of seven datasets spanning six task categories. Specifically, the bench-

Prompt generation guided by the structure of prompt patterns

You are assisting in prompt engineering for the {area} field, focusing on applications within the {benchmark} benchmark. Your task is to generate prompts based on the provided pattern template.

Pattern Details:
{prompt_pattern}
Requirements:
1. Generate **5 diverse yet related prompts** that align with the given pattern.
2. Ensure each prompt is:
- **Practical**: Ready for immediate use.
- **Specific**: Tailored to the area field.
- **Consistent**: Maintains the intent and structure of the provided pattern.
3. Format the prompts in JSON as shown below:
```json
{{
"prompts": [
"prompt1",
"prompt2",
"prompt3",
"prompt4",
"prompt5"
]
}}
```

Figure 6: Prompt generation guided by the structure of prompt patterns

Task category	Dataset	Metric
Code	HumanEval	pass@1
Code	LiveCodeBench	pass@1
Database	Spider	Execution accuracy
Actions	BFCL	Function-call accuracy
Math	GSM8K	Exact Match
Data-to-text	ToTTo	BLEU
Summarization	Haystack	Joint Score

Table 4: Datasets and evaluation metrics used in our experiments.

mark includes HumanEval (Chen et al., 2021) and LiveCodeBench (Jain et al., 2025) for code generation, Spider (Yu et al., 2018) for database querying, BFCL (Patil et al., 2025) for function calling, GSM8K (Cobbe et al., 2021) for mathematical reasoning, ToTTo (Parikh et al., 2020) for data-to-text generation, and Summary of a Haystack (Laban et al., 2024) for summarization. This task suite allows us to evaluate prompt-composition behavior across heterogeneous task formats, including structured prediction, executable generation, numerical reasoning, and long-context generation.

Following Laban et al. (2025), all task scores are mapped onto a common $[0, 100]$ scale. For binary-evaluated tasks, including code generation, database querying, function calling, and mathematical reasoning, a correct response receives a score of 100 and an incorrect response receives a score of 0. For generation tasks, we use the benchmark-standard continuous scores, namely BLEU for ToTTo and Joint Score for Summary of a Haystack.

Models. We evaluate eight LLMs spanning open-weight and proprietary model families: Meta-Llama-3.1-8B-Instruct (Grattafiori et al., 2024), GPT-4o-mini (OpenAI, 2024), GPT-4o (Hurst et al., 2024), Claude-Sonnet-4.5 (Anthropic, 2025), DeepSeek-V4-Flash (DeepSeek-AI, 2026), Gemini-2.5-Flash-Lite (Comanici et al., 2025), Gemini-2.5-Pro (Comanici et al., 2025), and Grok-4-Fast (xAI, 2025). This model suite covers both smaller instruction-tuned models and stronger proprietary models, allowing us to test whether prompt-composition behavior is specific to a single model family or persists across different deployment settings.

For reproducibility, Table 5 reports the exact model names used in our experiments, together with coarse model-size categories, weight availability, and whether a dedicated reasoning mode was enabled.

Model	Scale	Public	Reasoning
Llama-3.1-8B	Small	✓	×
GPT-4o-mini	Small	×	×
GPT-4o	Large	×	×
Claude-Sonnet-4.5	Large	×	×
DeepSeek-V4-Flash	Medium	✓	×
Gemini-2.5-Flash-Lite	Small	×	×
Gemini-2.5-Pro	Large	×	×
Grok-4.1-Fast	Large	×	✓

Table 5: Model suite used in our experiments. "Scale" denotes a coarse model-size class. "Public" indicates publicly available weights. "Reasoning Mode" indicates whether a dedicated reasoning or thinking mode was enabled in our runs.

Prompt Composition Protocol. For single-pattern prompting, we instantiate one prompt pattern and evaluate it together with the task input. For pairwise composition, we evaluate ordered pairs (p_i, p_j) with $i \neq j$, where the realization of p_i is placed before the realization of p_j. Both prompt orders are evaluated for each unordered pair. For multi-pattern composition, prompt patterns are appended according to the construction strategy described in the main text.

In all settings, the composed prompt realizations are concatenated into the system message, while the task instance is placed in the user message. Vanilla is used as the direct-task baseline and is not counted as a functional prompt pattern when reporting statistics over prompt-pattern compositions.

Evaluation Metrics and Score Aggregation. We adopt the task-specific evaluation metrics used by Laban et al. (2025). Let $\mathcal{T}$ denote the set of task categories and $\mathcal{D}_t$ denote the set of datasets under task category t. For a prompt composition C, we first compute the normalized dataset-level score $s_d(C)$ on each dataset d. For task categories containing multiple datasets, such as code generation, we average over datasets within the same category:

$$s_t(C) = \frac{1}{|\mathcal{D}_t|} \sum_{d \in \mathcal{D}_t} s_d(C).$$

We then compute the overall score by macro-averaging across task categories:

$$S(C) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} s_t(C).$$

This aggregation prevents task categories with more datasets or different metric scales from dominating the overall evaluation.

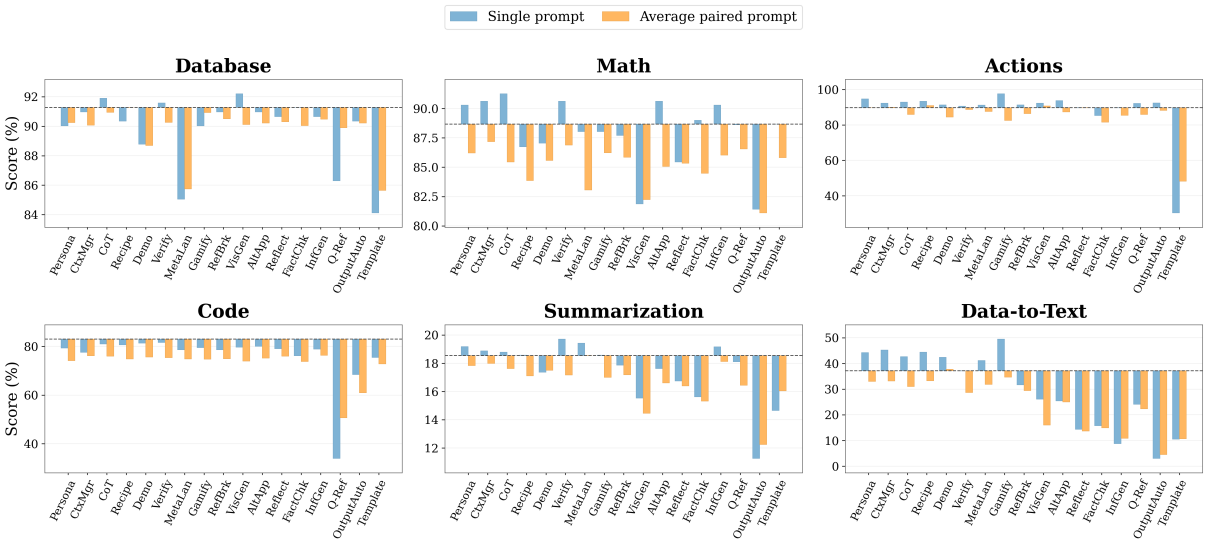

Figure 7: Pairwise prompt composition results on GPT-4o-mini across six task categories.

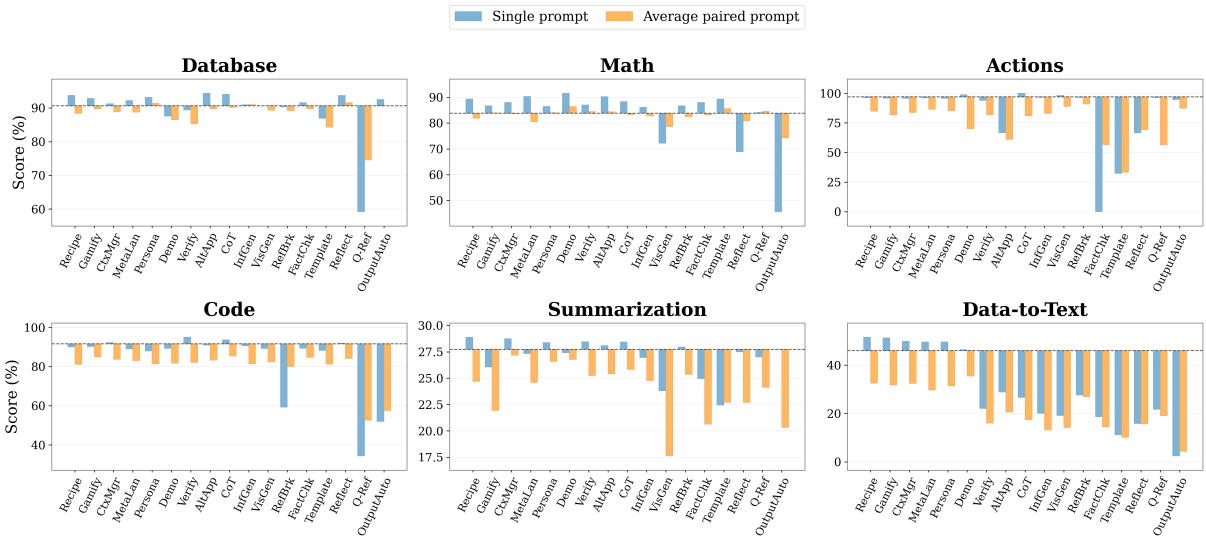

Figure 8: Pairwise prompt composition results on Gemini-2.5-Flash-Lite across six task categories.

C Additional Pairwise Composition Results Across Models

To examine whether the pairwise composition patterns observed in Section 3 are specific to Llama-3.1-8B-Instruct, we further evaluate the same single-prompt and pairwise-composition setting on GPT-4o-mini and Gemini-2.5-Flash-Lite. For each model, we report the single-prompt performance of each prompt pattern and its average paired-prompt performance over all valid partner patterns and both prompt orders, following the definition of $F_t^{\text{pair}}(p; \mathcal{P})$ in Eq. 8.

Figures 7 and 8 show that the main trend observed in Figure 3 remains consistent across additional models. Although the magnitude of the degradation varies across tasks and model families, average pairwise-composition performance is often lower than the corresponding single-prompt performance. This indicates that non-additive behavior observed on Llama-3.1-8B is not an artifact of a single model, but also appears in other models.

These results further support our conclusion that prompt patterns should not be treated as independently composable modules. Even when a prompt pattern is effective in isolation, combining it with another pattern can introduce redundancy, conflicting behavioral constraints, or instruction interference.

D Detailed Ordered Pairwise Interaction Matrices

Tables 7, 8, and 9 provide the full ordered pairwise interaction matrices for the additional models studied in the main text. These tables complement the aggregated results in Appendix C, where we report bar-plot summaries of single-prompt and average pairwise-composition performance.

Each matrix follows the same pairwise-composition protocol as Section 3. Rows denote the first prompt pattern in the ordered composition, and columns denote the second prompt pattern. Diagonal entries correspond to single-prompt runs, while off-diagonal entries correspond to ordered two-prompt compositions. Thus, the entry in row p and column q gives the performance of the ordered composition (p, q) under the corresponding model.

These full matrices expose interaction patterns that are averaged out in the bar-plot summaries. They allow us to inspect whether a prompt pattern consistently degrades when paired with others, whether certain prompt orders are more favorable, and whether the observed gains are concentrated in a small number of compatible pairs. Together with the main-text results, these matrices show that prompt-pattern interference is not specific to Llama-3.1-8B-Instruct, but also appears across the additional model families evaluated in our study.

E The Use of Large Language Models

Using an LLM to help with paper writing During the preparation of this work, the authors utilized Large Language Models for language polishing, improving the structural clarity of the manuscript, and refining the formal expression of individual sentences. The use of Large Language Models did not influence the substantive content of the study and served solely as a writing aid.

F Potential Risks

This work studies how different prompting strategies interact when they are composed. It does not introduce a new model, dataset, or deployment system, and all experiments are conducted on standard benchmark tasks. We therefore expect the direct risks of this work to be limited. A potential risk is that users may overgeneralize our findings and apply prompt-composition recommendations in high-stakes settings without task-specific validation. We emphasize that our results are empirical and benchmark-dependent, and that prompt strategies should be carefully evaluated before use in sensitive domains. In addition, while more effective prompting practices could potentially improve both benign and harmful uses of LLMs, our work focuses on characterizing composition effects and does not provide methods for bypassing safety mechanisms.

	Vanilla	Persona	CtxMgr	CoT	Recipe	Demo	Verify	MetaLan	Gamify	RefBrk	VisGen	AltApp	Reflect	FactChk	InfGen	Q-Ref	OutputAuto	Template	Flip
Vanilla	68.22± 1.07	69.81± 1.87	69.46± 1.14	69.37± 1.37	69.37± 2.28	68.20± 1.77	67.56± 1.24	67.40± 1.82	67.09± 2.17	65.66± 1.13	64.54± 1.97	64.26± 3.24	60.91± 2.65	59.87± 1.47	59.54± 1.46	57.20± 2.16	57.11± 1.85	54.10± 1.43	38.23± 6.02
Persona	69.81± 1.87	69.81± 1.87	70.78± 0.79	69.59± 1.72	70.71± 1.13	69.27± 1.45	67.51± 1.06	70.48± 0.98	68.56± 1.42	67.98± 1.10	62.18± 0.16	66.97± 2.94	60.73± 2.17	60.24± 1.03	59.74± 1.28	58.33± 6.69	60.12± 1.85	56.72± 0.34	51.75± 8.88
CtxMgr	69.46± 1.14	69.52± 1.55	69.46± 1.14	68.99± 1.13	70.96± 0.38	69.27± 2.13	68.81± 1.24	68.32± 0.37	68.41± 1.16	69.06± 0.56	62.59± 0.42	64.32± 1.84	60.43± 0.52	58.62± 0.74	60.50± 2.51	64.23± 1.67	58.15± 2.82	55.72± 0.45	49.50± 0.89
CoT	69.37± 1.37	70.76± 0.98	70.66± 1.21	69.37± 1.37	70.22± 0.98	69.05± 2.78	66.54± 0.52	69.14± 0.88	68.77± 1.09	67.22± 0.87	63.21± 2.20	64.65± 1.56	58.62± 1.22	60.61± 1.24	62.41± 1.05	61.04± 5.36	56.66± 3.86	57.61± 0.73	43.41± 5.23
Recipe	69.37± 2.28	68.87± 0.34	70.50± 0.19	70.69± 0.74	69.37± 2.28	68.78± 1.56	66.81± 1.06	70.55± 0.95	69.89± 2.24	69.24± 1.04	63.58± 0.24	68.96± 1.46	64.24± 1.35	59.22± 1.48	60.72± 2.98	59.76± 2.80	54.98± 1.35	56.95± 1.49	47.07± 4.98
Demo	68.20± 1.77	68.45± 0.93	69.45± 2.38	69.48± 2.29	69.60± 0.49	68.20± 1.77	66.80± 2.07	69.24± 0.94	65.92± 2.16	68.18± 1.62	65.09± 1.06	68.23± 2.18	65.32± 2.98	59.91± 3.09	62.53± 0.37	59.11± 2.15	55.59± 1.47	59.32± 2.72	53.02± 6.58
Verify	67.56± 1.24	68.64± 1.06	69.52± 0.52	66.85± 2.01	67.08± 1.44	66.77± 2.08	67.56± 1.24	66.71± 2.37	67.53± 1.72	66.01± 0.44	60.16± 4.11	63.38± 2.11	60.09± 2.91	60.44± 0.48	58.43± 1.88	59.96± 1.85	56.49± 2.52	55.42± 1.19	42.04± 4.86
MetaLan	67.40± 1.82	65.11± 1.41	66.09± 4.39	68.49± 1.31	69.63± 1.68	68.77± 2.07	65.36± 2.27	67.40± 1.82	68.13± 2.23	68.71± 2.26	62.10± 1.27	66.51± 3.28	58.63± 3.02	56.33± 4.06	59.79± 1.82	58.26± 1.73	57.15± 2.75	52.70± 1.98	41.88± 10.20
Gamify	67.09± 2.17	68.74± 0.74	69.71± 1.73	69.63± 0.23	70.09± 0.89	66.48± 1.88	69.53± 1.72	68.75± 1.37	67.09± 2.17	71.16± 1.54	62.48± 0.99	63.77± 4.97	62.30± 1.05	58.94± 1.87	60.04± 2.85	58.39± 3.71	57.01± 2.46	55.84± 1.36	41.41± 3.08
RefBrk	65.66± 1.13	68.35± 0.70	68.38± 0.92	68.69± 1.09	66.83± 1.21	66.18± 1.06	66.05± 1.01	67.28± 1.41	68.85± 2.26	65.66± 1.13	63.89± 2.05	66.31± 1.86	61.45± 1.67	58.28± 1.77	60.91± 0.83	62.03± 2.75	56.13± 3.46	55.37± 0.75	40.46± 8.93
VisGen	64.54± 1.97	65.28± 1.94	65.44± 0.60	65.27± 2.07	64.22± 2.07	64.92± 0.69	62.93± 1.57	63.11± 1.89	65.27± 0.80	65.82± 1.29	64.54± 1.97	63.72± 1.32	62.26± 0.31	60.47± 0.84	62.18± 1.43	52.63± 7.65	55.42± 0.79	58.09± 1.87	39.76± 1.96
AltApp	64.26± 3.24	68.67± 1.65	67.06± 3.80	64.32± 1.30	69.22± 0.61	67.25± 1.50	63.25± 1.68	67.24± 0.27	69.62± 1.91	65.33± 2.29	61.98± 1.06	64.26± 3.24	60.18± 2.58	58.41± 1.48	62.44± 1.74	57.71± 5.18	56.76± 0.89	56.11± 1.60	35.50± 5.82
Reflect	60.91± 2.65	63.58± 1.28	63.08± 0.84	61.63± 1.17	64.06± 0.81	68.14± 1.71	59.55± 2.54	63.72± 1.83	61.69± 2.07	62.65± 0.91	61.64± 1.28	61.05± 1.71	60.91± 2.65	57.40± 2.16	60.86± 1.11	58.79± 1.67	56.62± 0.52	53.69± 1.02	40.48± 2.46
FactChk	59.87± 1.47	60.90± 1.79	61.84± 2.84	61.25± 3.14	58.37± 1.20	64.07± 0.59	61.71± 1.69	61.63± 1.67	60.72± 2.44	59.08± 1.95	60.87± 1.17	56.79± 0.87	57.67± 2.07	59.87± 1.47	62.68± 0.40	60.36± 1.66	53.66± 1.86	53.75± 0.85	45.16± 5.26
InfGen	59.54± 1.46	59.98± 2.91	62.34± 0.25	62.54± 0.38	59.80± 0.85	62.35± 1.44	59.91± 2.26	60.65± 1.75	57.87± 1.09	60.20± 0.60	61.39± 0.92	57.79± 1.69	59.09± 3.07	57.84± 2.07	59.54± 1.46	58.09± 2.65	52.36± 3.27	57.16± 1.35	44.92± 2.20
Q-Ref	57.20± 2.16	58.93± 0.94	66.81± 1.70	56.65± 3.45	56.99± 5.10	54.20± 2.21	56.88± 4.32	57.43± 5.10	58.20± 0.37	57.94± 2.18	50.96± 6.87	57.48± 6.13	58.51± 1.97	56.56± 0.94	54.04± 3.93	57.20± 2.16	50.35± 3.46	46.66± 4.51	32.62± 5.64
OutputAuto	57.11± 1.85	58.40± 1.46	58.83± 1.10	55.49± 0.57	56.34± 1.74	53.77± 2.73	54.44± 4.17	54.58± 3.34	51.40± 5.15	54.97± 2.17	54.99± 0.75	56.17± 1.01	57.67± 1.32	56.82± 2.66	53.93± 1.94	48.54± 5.30	57.11± 1.85	50.15± 2.83	41.37± 4.84
Template	54.10± 1.43	54.25± 2.63	55.75± 2.04	56.55± 0.44	55.16± 0.86	61.11± 2.68	54.14± 0.97	54.16± 1.34	53.18± 0.12	56.43± 1.47	54.74± 2.14	57.16± 1.42	55.60± 0.76	52.36± 2.78	55.60± 0.95	53.25± 1.83	51.91± 0.65	54.10± 1.43	36.56± 5.88
Flip	38.23± 6.02	49.65± 4.39	55.55± 5.70	41.79± 2.40	46.52± 3.62	49.05± 7.26	47.84± 3.46	47.42± 4.82	56.89± 6.42	41.36± 5.92	31.28± 4.68	40.79± 2.67	40.91± 6.31	35.50± 1.63	40.06± 2.77	37.21± 7.10	35.87± 4.69	34.34± 7.26	38.23± 6.02

Table 7: Averaged pairwise score of GPT-4o-mini across 6 tasks. Diagonal entries correspond to single-prompt runs; Off-diagonal entries correspond to ordered two-prompt compositions. Red indicates the performance degradation.

	Vanilla	CtxMgr	Persona	CoT	Demo	Recipe	Verify	AltApp	MetaLan	RefBrk	Gamify	Q-Ref	FactChk	Reflect	VisGen	InfGen	Flip	OutputAuto	Template
Vanilla	74.13	75.57	74.51	73.91	73.21	72.88	72.64	72.32	72.25	71.14	71.14	69.53	67.17	67.06	64.89	64.33	60.38	60.33	60.17
CtxMgr	75.57	75.57	72.45	72.96	71.20	75.54	74.98	67.56	76.21	75.92	71.01	70.50	68.25	64.20	66.12	65.49	57.32	51.07	61.68
Persona	74.51	74.34	74.51	67.54	74.86	73.78	74.72	74.87	75.61	76.01	73.20	56.30	64.16	66.80	67.57	63.68	57.24	54.80	53.38
CoT	73.91	76.48	74.94	73.91	71.08	75.55	73.42	74.06	75.95	74.71	71.20	63.34	65.63	66.90	66.21	64.46	60.57	51.09	62.60
Demo	73.21	74.52	74.58	71.03	73.21	71.45	68.03	69.02	74.74	71.72	71.88	64.64	62.77	65.43	70.36	66.91	51.45	67.75	65.58
Recipe	72.88	76.48	74.09	76.15	72.23	72.88	73.08	75.54	75.45	74.14	73.11	69.14	67.50	67.06	66.90	61.37	56.08	55.66	61.93
Verify	72.64	75.32	73.80	71.95	71.81	75.30	72.64	73.98	76.14	73.51	69.67	73.43	68.39	64.41	73.34	61.37	58.05	62.30	62.09
AltApp	72.32	74.21	72.54	72.00	70.00	74.03	70.33	72.32	75.74	73.42	73.13	51.84	64.88	63.79	72.19	63.55	41.69	53.36	64.08
MetaLan	72.25	75.79	75.32	74.53	71.84	73.64	74.78	72.58	72.25	72.80	71.63	73.66	62.55	68.70	67.13	67.01	49.89	66.48	61.36
RefBrk	71.14	74.73	74.78	74.70	68.53	72.33	72.49	72.12	76.12	71.14	68.35	66.02	64.73	67.77	70.48	61.51	53.87	48.47	51.45
Gamify	71.14	73.89	69.44	73.69	71.88	73.17	70.25	63.65	74.55	74.32	71.14	56.54	62.89	64.66	63.23	62.18	50.90	58.63	61.82
Q-Ref	69.53	75.69	68.39	60.92	70.13	66.15	69.70	66.04	71.23	70.49	69.34	69.53	66.27	66.80	53.06	63.43	56.85	57.91	50.93
FactChk	67.17	64.20	66.80	61.12	63.08	64.71	65.72	66.95	68.28	67.22	60.80	62.91	67.17	64.27	65.08	63.76	54.02	53.13	60.54
Reflect	67.06	69.30	68.28	67.77	71.06	67.96	65.85	64.80	67.92	67.87	62.31	64.88	64.92	67.06	60.59	63.69	47.80	54.81	58.69
VisGen	64.89	66.81	74.77	68.25	74.29	74.59	68.75	71.49	70.57	66.72	75.46	59.08	65.45	67.49	64.89	62.21	55.24	60.21	61.35
InfGen	64.33	64.33	65.07	63.98	70.34	61.10	62.31	61.70	64.21	64.34	63.04	63.86	62.54	61.29	61.77	64.33	51.50	54.96	61.54
Flip	60.38	66.17	69.66	49.53	57.93	58.83	60.86	53.94	69.80	54.88	56.03	31.20	55.32	50.84	42.46	47.51	60.38	44.92	33.73
OutputAuto	60.33	57.21	59.51	56.16	70.55	59.51	54.23	56.40	64.59	57.55	52.98	51.99	60.52	52.45	49.79	58.14	33.54	60.33	57.80
Template	60.17	63.42	47.80	62.17	59.23	46.72	61.82	62.67	60.98	61.22	47.90	59.87	61.66	61.45	61.12	61.15	18.97	55.59	60.17

Table 8: Averaged pairwise score of GPT-4o across 6 tasks. Diagonal entries correspond to single-prompt runs; Off-diagonal entries correspond to ordered two-prompt compositions. Red indicates the performance degradation.

	Vanilla	Demo	MetaLan	OutputAuto	Persona	VisGen	Recipe	Template	FactChk	Reflect	Q-Ref	AltApp	Verify	RefBrk	Flip	Gamify	InfGen	CtxMgr	CoT
Vanilla	43.05	42.96	42.29	40.77	44.27	45.00	42.27	42.27	38.86	42.61	42.67	41.04	42.90	44.75	44.14	44.02	41.46	45.37	43.05
Demo	41.34	40.16	42.36	41.05	40.02	41.81	41.95	41.61	40.51	41.73	38.76	41.06	40.24	44.47	39.91	43.09	41.13	41.60	40.48
MetaLan	44.29	41.83	43.16	40.35	42.59	41.64	43.47	41.39	39.09	42.05	42.15	43.46	43.40	43.37	44.35	41.33	40.45	44.28	43.84
OutputAuto	41.47	41.45	41.93	41.92	40.32	41.46	40.72	40.93	38.24	44.64	40.84	40.76	38.70	41.93	42.73	40.48	41.61	41.23	40.92
Persona	43.84	41.93	43.09	36.75	44.46	46.00	41.19	40.36	39.43	44.22	43.02	42.33	42.50	42.59	44.55	41.35	38.76	44.59	44.79
VisGen	43.97	43.74	43.18	41.39	40.76	43.77	44.07	43.09	40.68	40.09	40.89	45.03	40.00	44.36	43.76	43.28	42.89	43.39	44.35
Recipe	42.39	38.86	40.45	37.91	45.19	42.64	42.92	41.69	39.70	42.41	42.35	40.78	43.01	44.40	42.52	38.57	39.99	43.11	43.87
Template	40.97	44.27	39.28	40.10	39.14	42.82	40.58	41.99	40.40	40.35	42.31	41.01	42.04	40.38	41.42	41.81	43.19	41.58	41.29
FactChk	41.38	41.37	41.85	41.32	42.00	40.12	44.26	42.12	41.96	41.84	43.15	41.69	41.63	42.81	43.05	43.49	41.33	44.04	44.19
Reflect	44.09	38.99	39.54	42.26	40.13	38.86	44.17	42.05	40.70	42.21	42.51	44.41	41.98	43.80	43.37	42.53	40.50	44.90	43.82
Q-Ref	44.41	42.06	42.62	41.00	42.56	43.73	42.76	38.27	41.34	40.98	40.78	42.28	40.16	44.61	43.04	41.33	44.47	43.20	43.30
AltApp	42.25	39.65	42.47	38.80	39.33	40.14	44.68	41.10	41.76	39.37	42.75	42.27	44.98	43.66	44.34	41.65	41.95	46.00	39.77
Verify	44.23	39.99	42.91	43.13	42.02	42.23	40.25	39.63	40.14	42.32	40.60	45.03	42.61	44.82	43.27	39.51	42.64	43.86	40.56
RefBrk	47.43	41.91	46.01	39.38	42.60	42.38	43.61	41.18	41.46	42.60	46.36	41.44	44.16	43.13	43.82	41.57	41.31	46.43	44.16
Flip	46.47	42.60	41.13	43.06	44.62	45.12	41.91	41.34	40.85	41.04	42.44	40.74	44.28	43.74	46.26	43.28	40.05	44.30	44.08
Gamify	42.56	43.05	42.66	41.27	41.80	43.67	44.20	40.65	42.85	41.35	39.29	39.03	38.69	45.24	41.62	41.84	42.45	42.08	41.24
InfGen	38.48	41.66	41.39	39.99	40.31	43.01	41.21	43.33	40.69	42.97	42.07	42.11	39.96	43.12	42.16	42.35	41.37	42.97	41.48
CtxMgr	44.62	42.01	44.65	39.77	46.03	46.63	44.19	41.39	44.71	45.76	45.09	43.46	44.78	41.96	45.30	40.60	44.09	46.77	45.05
CoT	46.27	41.56	42.20	40.66	45.04	44.82	43.44	39.85	42.10	45.23	43.42	46.15	41.26	44.48	41.24	44.45	42.46	45.39	42.20

Table 9: Averaged pairwise score of DeepSeek-R1 across 6 tasks. Diagonal entries correspond to single-prompt runs; Off-diagonal entries correspond to ordered two-prompt compositions. Red indicates the performance degradation.