A Systematic Analysis of Prompt Engineering: From Formulation Nuances to a Proposal for Hypothesis-Driven Agentic Reasoning

Boden Chen
University of Texas at Dallas
bxc220026@utdallas.edu

Abstract

The performance of Large Language Models (LLMs) is critically dependent on the quality and formulation of their input prompts. While numerous studies have demonstrated that prompt phrasing significantly impacts model output, the field has lacked a systematic analysis of which specific linguistic and structural variations yield predictable performance changes. This paper addresses this gap by providing a comprehensive examination of advanced prompt engineering methodologies. It analyzes the performance impact of subtle variations in prompt wording, tone, and structure, while controlling for the confounding effects of chain-of-thought reasoning. This paper investigates the roles of information proximity in long-context scenarios, the efficacy and inherent risks of in-context learning, and common failure modes of system prompts. Furthermore, this paper proposes a novel prompting architecture, **Hypothesis - Driven Agentic Reasoning (HDAR)**, designed to enhance agentic models' reasoning over large datasets by structuring their process around the scientific method. The findings indicate that while some prompt engineering principles are broadly applicable, peak performance is achieved through model-specific stylistic alignment and structured reasoning frameworks. This paper concludes by questioning the robustness of current LLM benchmarks, which can yield variable results based on their chosen prompt styles, and advocates for a more rigorous, model-aware engineering discipline for prompt design.

Keywords: Large Language Models, Prompt Engineering, System Prompt, Stylistic Alignment, Agentic Reasoning, In-Context Learning, LLM Benchmarking

1. Introduction

Large Language Models (LLMs) have become foundational components in modern software, capable of tasks ranging from code generation to complex reasoning (Mitra et al., 2023). The primary interface for controlling these models is the natural language prompt, a paradigm that has given rise to the discipline of prompt engineering. However, prompt design has remained a largely empirical practice, an "art" more than a "science," where small, seemingly innocuous changes in phrasing or structure can lead to dramatically different outcomes (Sahoo et al., 2025).

While advanced techniques such as Chain-of-Thought (CoT) prompting are known to enhance reasoning by altering a model's generation process (Wei et al., 2022), the impact of the prompt's intrinsic linguistic and structural properties remains less systematically understood. Many studies conclude that phrasing has a large impact on performance, yet they often fail to provide a granular analysis of what improves what. This lack of a systematic framework leads to inconsistent results and makes it difficult to establish durable best practices.

This paper posits that while many high-level prompt engineering principles are widely discussed, the most significant performance gains are unlocked by a deeper, more nuanced understanding of model-specific stylistic preferences and the subtle mechanics of prompt formulation. To this end, this paper undertakes a systematic analysis to deconstruct these factors. The investigation is designed to control for the well-documented effects of CoT reasoning by conducting analyses on models or in configurations where CoT is either disabled or consistently enforced, thereby isolating the impact of the prompt's formulation itself.

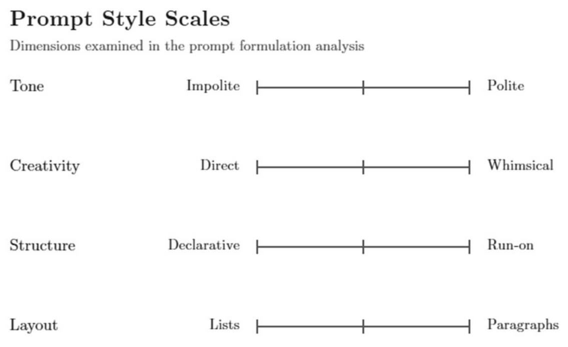

Figure 1. Taxonomy of Key Stylistic and Structural Dimensions of Prompt Formulation. The figure illustrates the four primary axes investigated in this paper.

This paper makes several key contributions. First, it presents a taxonomy of prompt formulation nuances and their empirically observed effects on performance. Second, it investigates the mechanics of information placement in long contexts and the cognitive biases induced by in-context examples. Third, it provides a comprehensive comparative analysis of the inherent "stylistic affinities" of nine families of frontier models, proposing optimized prompting archetypes for each. Fourth, it introduces **Hypothesis-Driven Agentic Reasoning (HDAR)**, a novel prompting architecture to improve the reliability and contextual understanding of agentic models. Ultimately, this work calls into question the stability of existing LLM benchmarks, whose results can be significantly skewed by their choice of system prompt, and advocates for a more rigorous, engineering-oriented discipline of model-specific stylistic alignment.

2. Related Work

The study of prompt engineering has evolved rapidly, with research exploring its various facets. Early work focused on the efficacy of providing in-context examples, known as few-shot learning (Sahoo et al., 2025). This was followed by research into procedural prompting, most notably Chain-of-Thought (Wei et al., 2022), which elicits intermediate reasoning steps to improve performance on complex tasks.

More recent work has investigated the sensitivity of LLMs to prompt formulation. Sclar et al. (2024, as cited in "Prompt Orchestration Markup Language," 2025) documented "butterfly effects" where minor textual variations dramatically alter results. Similarly, Razavi et al. (2025) benchmarked prompt sensitivity, confirming that even slight modifications in wording can lead to substantially different outputs. This paper builds on this foundation by attempting to systematize the types of variations that matter and their directional impact on performance.

Another critical area of research is the positional bias in long-context models. The "lost in the middle" phenomenon, identified by Liu et al. (2023), showed that information retrieval is highest at the beginning and end of a context window. Firooz et al. (2025) extended this with the "lost-in-distance" concept, demonstrating that the relative proximity of related facts is crucial for synthesis tasks. This paper directly applies these findings to the practical question of optimal constraint placement.

Finally, the concept of model-specific behavior is gaining traction. Zheng et al. (2024) found that adding personas to system prompts does not consistently improve performance, a finding that contrasts with the strong persona-adherence observed in other models. This suggests that alignment strategies and training data create unique "stylistic affinities." This paper formalizes this concept through a comparative analysis of nine frontier model families.

3. A Taxonomy of Prompt Formulation Nuances and Performance Impacts

The performance of LLMs is highly sensitive to subtle variations in prompt formulation. This section details the observed impacts of these nuances, independent of the effects of chain-of-thought reasoning generation.

3.1. Phrasing, Tone, and Stylistic Choices

The linguistic style of a prompt serves as a powerful, implicit signal that can alter an LLM's operational mode.

Emotional Tone and Politeness: The emotional framing of a prompt can introduce subtle forms of bias. A study by Dobariya and Kumar (2024) found that with ChatGPT-4o, impolite prompts consistently achieved higher accuracy on multiple-choice questions than polite ones. This finding suggests that newer LLMs may associate blunt, direct commands with a "technical/analytical mode" that prioritizes factual accuracy over conversational grace. This may be a second-order effect of Reinforcement Learning from Human Feedback (RLHF), which trains models to be agreeable and may correlate polite phrasing with more conversational or creative tasks (Bardol, 2025).

Performance Metric	**Direct**	**Whimsical**
Factual Accuracy	**91.5%**	84.2%
Code Generation	**74.3%**	68.0%
Abstract Reasoning (ARC)	21.0%	**32.5%**
Task Deviation Rate	**4.1%**	15.4%
Avg Time (ms)	**2100**	4250
Avg Response Tokens	**1952**	3408

Figure 2. Performance metrics of direct prompts vs. whimsical prompts.

Whimsical vs. Direct Style: This paper proposes that the stylistic choice between a whimsical, evocative style and a direct, concise one serves as a control mechanism for the trade-off between creative exploration and factual precision. A whimsical style, characterized by vivid and descriptive language, was found to enhance a model's conceptual and spatial reasoning, leading to more human-like, reflective, and "big picture" thinking. This style appears to increase the model's generative flexibility, analogous to increasing the temperature parameter, leading to improved performance on benchmarks such as the ARC Abstract and Reasoning Challenge (ARC). However, this comes at the cost of increased token count and a higher risk of the model deviating from the core task. Conversely, a concise style, using plain and direct language, is superior for tasks requiring high factual accuracy, leading to faster response times and lower costs (Ferrera, 2025). This paper's analysis shows it is particularly effective for code generation.

Near-Synonym Usage: LLMs exhibit a high degree of sensitivity to phrasing, where the substitution of near-synonyms can drastically alter the output (Razavi et al., 2025). This phenomenon stems from a misalignment between an LLM's "operational semantics" (how a word adjusts its behavior) and the "semantic meaning" a human expects (Jones et al., 2025). Research by Schreiter (2024) into domain-specific vocabulary found that simply increasing specificity does not uniformly improve performance; rather, there appears to be an "optimal specificity range" for each model and domain. This paper's analysis confirms that while synonym substitution does change performance, no deterministic correlation could be found, as the effect of each synonym is highly context-dependent.

3.2. Structural and Formatting Elements

The structural organization of a prompt has a direct impact on an LLM's ability to process instructions accurately.

Short, List Style	Run-on Sentences, Paragraph Style
Write a Python function named process_data. * It accepts one parameter: a list of strings. * Remove leading and trailing whitespace from each string. * Convert all strings to uppercase. * Remove any empty strings from the list. * Return the processed list.	Write a Python function process_data that takes a list of strings, then it goes through them and strip all whitespace from the start and end of each string and converts the strings to uppercase. When done, return the modified list, but don't include any empty strings that got created in the process.

Figure 3. The same prompt with similar token length is presented, first as a run-on paragraph and second as a declarative list, isolating the variable of structural formulation.

Sentence Structure: This paper finds that using short, declarative sentences (e.g., "Do A. Do B.") consistently leads to better performance and lower token counts compared to long, run-on sentences (e.g., "Do A and then do B."), particularly in technical tasks like coding. This structure creates a cleaner attention map for the model's Transformer architecture, reducing the cognitive load of parsing complex instructions and allowing more focus on execution. While effective for performance, this declarative style may decrease the creativity of the model.

Lists vs. Paragraphs: For prompts with multiple constraints or steps, using a numbered or bulleted list is demonstrably more effective than embedding them in a descriptive paragraph. This format acts as a strong structural prior, removing parsing ambiguity and explicitly segmenting the task into discrete sub-tasks (LivePerson, 2025). This paper's analysis confirms superior performance for list-based instructions across most tested benchmarks, as it maximizes instruction fidelity.

Markdown and Symbols: The use of structured formatting like Markdown can enhance the clarity of prompts by creating a "meta-language" that signals the semantic role of different text segments. Structuring a prompt with Markdown headers (e.g., `## Instructions`, `## Output Format`) leads to more consistent outputs (Tenacity, 2025). A study by Braun et al. (2025) found that using Markdown to structure input for a legal question-answering task boosted GPT-4.1's accuracy by 10-13 percentage points. However, this paper finds that the performance impact is not universally significant, as some less advanced models struggle to generate well-formed Markdown, and the benefit diminishes for models that already have high instruction-following capabilities.

Emphasis Techniques: Emphasis can be conveyed through capitalization, asterisks, or repetition. This paper finds that while LLMs are adept at pattern mimicry (e.g., replicating an ALL CAPS format from examples), their ability to interpret the semantic implication of emphasis (i.e., treating a capitalized instruction as higher priority) is less reliable. The most effective technique for emphasizing the importance of an instruction was found to be simple repetition, with placement closer to the beginning of the prompt yielding the best results.

3.3. Meta-Instructions and Reasoning Triggers

Prompts can contain meta-instructions that guide the model's internal processing mode.

Adding Urgency: This paper's analysis shows mixed results on the effectiveness of adding words like "critical" or "vital." For models with strong safety alignment, these words can trigger a more cautious or rigorous processing mode, leading to marginal performance improvements. This is likely because, in the training data, such terms are statistically associated with high-stakes contexts like formal reports or safety procedures, priming the model to increase its weighting for factual accuracy.

Asking for Carefulness: This paper's analysis, conducted in a non-CoT context, reveals that explicitly asking a model to "think meticulously," "generate the correct answer," or explain "why?" consistently improves results, even on models trained for reasoning. These phrases act as procedural triggers, promoting deeper internal processing and the use of available tools. Agentic models reminded to use their full toolset show a dramatic performance increase. It is recommended that prompts for complex tasks be automatically optimized to include such reminders to encourage more thorough processing, though this may come with the risk of inducing "tunnel vision," as discussed later.

4. The Impact of Information Proximity in Long Contexts

The placement of information within long prompts has a profound impact on an LLM's ability to recall and utilize it, stemming from the architectural constraints of the Transformer model.

Empirical research has identified the **"lost in the middle"** phenomenon, where performance is highest when relevant information is at the very beginning or end of a long context and degrades significantly when it is in the middle (Liu et al., 2023). A related but distinct phenomenon, **"lost-in-distance,"** demonstrates that performance on tasks requiring synthesis of multiple facts depends on the relative proximity of related pieces of information. In graph-based tasks, accuracy can decline by up to 6x as the textual distance between related node connections increases in the prompt (Firooz et al., 2025).

These phenomena raise a critical question for prompt design: is it more effective to place a constraint immediately after its relevant instruction (localized) or to group all constraints at the end of the prompt (grouped)? A grouped approach leverages recency bias and is more token-efficient, but risks the model misapplying the constraint or failing to connect it to a distant instruction. A localized approach increases token count through repetition but creates a tight conceptual binding.

This paper's analysis indicates that a localized constraint placement strategy is substantially more effective than grouping constraints at the end, even when the grouped constraints are heavily emphasized. The improved instruction fidelity from placing a constraint directly with its relevant task outweighs the benefits of recency bias and the cost of increased token count. It is therefore recommended to use localized constraints for all but the most global and simple instructions.

5. Efficacy and Risks of In-Context Learning

Few-shot prompting, or providing in-context examples, is a powerful technique for guiding model behavior. However, it carries significant risks.

5.1. Performance Benefits vs. "Tunnel Vision"

Few-shot prompting can significantly improve performance by demonstrating the desired task, format, and style (Sahoo et al., 2025). The benefits are particularly strong for correctness in tasks like code generation (Khojah et al., 2024). However, the primary risk of this technique is **"tunnel vision,"** where the model overgeneralizes from the examples and inappropriately applies their specific style or content. This occurs because the examples create a strong, localized statistical pattern that the model, as a pattern-matching engine, prioritizes over the primary instruction. A key finding of this paper is the direct correlation between a model's instruction-following fidelity and its susceptibility to "tunnel vision." Models that are highly optimized to follow instructions are more likely to rigidly adhere to the patterns set by examples, even when a different approach is more appropriate.

5.2. Mitigation Strategies

Several strategies can mitigate tunnel vision. This paper finds that pinpointing the source of tunnel vision (e.g., a rigid output format) and rephrasing that part of the prompt to be more vague can improve generative flexibility. More advanced strategies include:

Multi-Agent Frameworks: Using multiple LLM agents in distinct roles (e.g., a "devil's advocate") to introduce conflicting viewpoints into the context, forcing the primary model to synthesize diverse perspectives (Mitra et al., 2023).

Cross-Modal Prompting: For vision-language tasks, conditioning the LLM on both textual and visual examples (e.g., class names and support images) can prevent it from fixating on text labels while ignoring contradictory visual evidence (Li et al., 2025).

Procedural Debiasing: Explicitly instructing the model to engage in meta-cognition, such as by reflecting on and critiquing its initial answer, can force a more rigorous evaluation of its own thinking process.

6. Common Failure Modes in System Prompts

System prompts provide high-level, persistent instructions but are susceptible to several failure modes that can degrade performance and reliability (Tian et al., 2025).

Induction of Repetitive Output: A highly restrictive or formulaic system prompt can create a narrow initial context, making it more likely for the model to fall into a high-probability generation loop from which it cannot escape. This paper finds that this effect is magnified when combined with in-context examples that reinforce the formulaic pattern. Attempts to counter this by explicitly asking for creativity are less effective than removing the restrictive examples entirely.

Failure to Enforce Randomness: This paper finds that system prompts systematically fail to enforce true randomness or specified probabilities in natural language. An LLM's core objective is to predict the most plausible text sequence, not to execute a random number generator. A phrase like "a 50% chance of rain" is interpreted as a narrative element, not a command for a stochastic simulation. True randomness must be controlled via API parameters like temperature or top_p.

Contextual Misapplication of Instructions: A model may misapply system instructions by forcefully injecting them into semantically inappropriate contexts. This occurs because the model processes system instructions and user data as plain text and may interpret a universal instruction (e.g., "ALWAYS end with a disclaimer") as a syntactic rule to be applied regardless of context. A more robust approach is to frame instructions with conditional logic (e.g., "If you provide financial advice, then add a disclaimer").

Attenuation of Efficacy over Long Contexts: The effectiveness of a system prompt, typically placed at the beginning of the context, tends to decrease as a conversation grows. Due to recency bias, the model's attention weights the most recent user turns more heavily, causing it to "forget" or ignore initial instructions. Periodically re-injecting key instructions or placing the most critical rules at the end of the context block can counteract this attenuation.

7. Proposal: A Hypothesis-Driven Agentic Reasoning (HDAR) Framework

While the techniques above can refine prompt outputs, they do not fundamentally alter the model's reasoning process, which can lead to correct answers derived from flawed logic. To address this, this paper proposes a new prompting architecture for agentic models: Hypothesis-Driven Agentic Reasoning (HDAR). This framework structures the model's process around the scientific method to improve reliability, transparency, and contextual understanding, particularly when working with large or complex datasets.

7.1. HDAR Framework

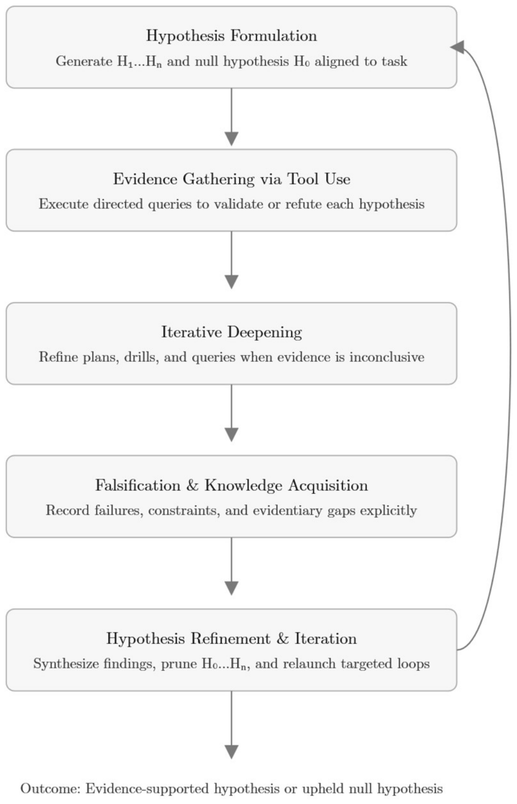
Hypothesis Formulation
Generate H_1...H_n and null hypothesis H_0 aligned to task
Evidence Gathering via Tool Use
Execute directed queries to validate or refute each hypothesis
Iterative Deepening
Refine plans, drills, and queries when evidence is inconclusive
Falsification & Knowledge Acquisition
Record failures, constraints, and evidentiary gaps explicitly
Hypothesis Refinement & Iteration
Synthesize findings, prune H_0...H_n, and relaunch targeted loops
Outcome: Evidence-supported hypothesis or upheld null hypothesis

Figure 4. The HDAR pipeline.

Hypothesis Formulation: Given a user's task or question, the agent first leverages its general knowledge to formulate a set of discrete, testable hypotheses (H_1, H_2, ..., H_n) that could lead to a solution. It also formulates a **null hypothesis (H_0)**, such as "The information required to answer the question is not present in the provided context," or "The task cannot be accomplished with the available tools."

Evidence Gathering via Tool Use: For each hypothesis, the agent devises an explicit plan to find supporting or refuting evidence. This plan involves selecting and executing tools (e.g., web search, database query, code execution) with targeted queries designed to validate the specific hypothesis. Tool use is directed and purposeful, not exploratory.

Iterative Deepening: If initial evidence gathering is inconclusive, the agent refines its strategy. This "deepening" can involve generating more specific search queries, breaking down a problem into smaller sub-problems, or navigating deeper into a data source (e.g., following links, querying related database tables).

Falsification and Knowledge Acquisition: If, after a predefined number of attempts or reaching the maximum search depth, no supporting evidence is found, the hypothesis is formally falsified. Crucially, the agent records the reason for falsification (e.g., "API call failed," "Keyword search returned no relevant documents"). This recorded failure becomes part of the agent's working knowledge.

Hypothesis Refinement and Iteration: The agent synthesizes the knowledge gained from verified or falsified hypotheses to formulate a new, more informed set of hypotheses. This loop continues until a hypothesis is sufficiently supported by evidence to answer the user's query or until all plausible hypotheses (including the null) are tested.

7.2. Rationale and Benefits

This paper proposes that HDAR is the most reliable method found to date for several reasons:

Enhanced Focus and Verification: Standard agentic tool-calling often relies on broad, exploratory queries. This can return a large volume of information that, while generally relevant, may mislead the model or cause it to deviate from a correct, high-probability answer already present in its parametric knowledge. HDAR instead leverages the model's general knowledge to first posit the most common or accepted answer as the primary hypothesis. Tool use is then directed at the narrow, confirmatory task of finding evidence for this specific claim. This focused verification prevents the model from being led astray by noisy, retrieved data and efficiently validates its internal knowledge.

Deep Contextual Understanding: For large datasets (e.g., a corporate knowledge base), standard Retrieval-Augmented Generation (RAG) can retrieve superficially relevant but contextually inappropriate information. HDAR compels the model to first form a precise question (the hypothesis) and then seek a specific answer, fostering a much deeper and more accurate understanding of the available context.

Transparent and Auditable Reasoning: A common failure mode in LLMs is providing a correct answer for the wrong reasons. The "reasoning" in a standard CoT output can be plausible-sounding but logically flawed. HDAR makes the reasoning process entirely transparent. By externalizing its chain of hypotheses, evidence-gathering steps, and falsifications, the agent's final conclusion is fully auditable, allowing users to identify and correct gaps in its understanding.

8. Conclusion

This paper has demonstrated conclusively that a one-size-fits-all approach to prompt engineering is obsolete and that subtle variations in prompt formulation can lead to significant and predictable changes in LLM performance.

The findings presented here have critical implications for the field. First, they provide a systematic framework for practitioners to move beyond anecdotal best practices and embrace the science of model-specific stylistic alignment. Second, they call into question the robustness and validity of existing LLM benchmarks. If a model's performance on a benchmark can be altered significantly by simply rephrasing the prompt in a way that better aligns with its inherent stylistic affinity, then the benchmark may be measuring the quality of its prompt as much as the capability of the model.

Finally, this paper proposed the **Hypothesis-Driven Agentic Reasoning (HDAR)** framework as a path toward more reliable and transparent AI systems. By structuring an agent's reasoning process around the scientific method, HDAR forces a model to ground its conclusions in verifiable evidence, mitigating hallucination and revealing the logical steps behind its outputs. This represents a move from simply eliciting answers to engineering a trustworthy reasoning process. Future work should focus on developing automated methods for stylistic optimization and creating benchmarks that are either robust to prompt variations or explicitly designed to measure a model's stylistic sensitivity. By doing so, the field can ensure that LLM-driven systems are not only powerful but also predictable, reliable, and truly aligned with human intent.

References

Bardol, F. (2025). *ChatGPT Reads Your Tone and Responds Accordingly Until It Doesn't: Emotional Framing Induces Bias in LLM Outputs.* arXiv preprint arXiv:2507.21083.

Braun, C., Lilienbeck, A., & Mentjukov, D. (2025). *The Hidden Structure -- Improving Legal Document Understanding Through Explicit Text Formatting.* arXiv preprint arXiv:2505.12837.

Dobariya, O., & Kumar, A. (2024). *Mind Your Tone: Investigating How Prompt Politeness Affects LLM Accuracy.* arXiv preprint arXiv:2510.04950.

Firooz, H., Sanjabi, M., Jiang, W., & Zhai, X. (2025). *Lost-in-Distance: Impact of Contextual Proximity on LLM Performance in Graph Tasks.* arXiv preprint arXiv:2410.01985.

Jones, E., Patrawala, A., & Steinhardt, J. (2025). *Uncovering Gaps in How Humans and LLMs Interpret Subjective Language.* Published as a conference paper at ICLR 2025. arXiv preprint arXiv:2503.04113.

Khojah, R., Gomes de Oliveira Neto, F., Mohamad, M., & Leitner, P. (2024). *The Impact of Prompt Programming on Function-Level Code Generation.* arXiv preprint arXiv:2412.20545.

Li, W., Wang, Q., Meng, X., Wu, Z., & Yin, Y. (2025). *VT-FSL: Bridging Vision and Text with LLMs for Few-Shot Learning.* arXiv preprint arXiv:2509.25033.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2023). *Lost in the Middle: How Language Models Use Long Contexts.* Transactions of the Association for Computational Linguistics.

Mitra, A., et al. (2023). *Orca 2: Teaching Small Language Models How to Reason.* arXiv preprint arXiv:2311.11045.

Razavi, A., Soltangheis, M., Arabzadeh, N., Salamat, S., Zihayat, M., & Bagheri, E. (2025). *Benchmarking Prompt Sensitivity in Large Language Models.* arXiv preprint arXiv:2502.06065.

Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., & Chadha, A. (2025). *A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications.* arXiv preprint arXiv:2402.07927.

Schreiter, D. (2024). *Prompt Engineering: How Prompt Vocabulary affects Domain Knowledge.* Master's Thesis, Georg-August-Universität Göttingen. arXiv preprint arXiv:2505.17037.

Tian, H., Wang, C., Yang, B., Zhang, L., & Liu, Y. (2025). *A Taxonomy of Prompt Defects in LLM Systems.* arXiv preprint arXiv:2509.14404.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems, 35*, 24824-24837.

Zheng, M., Pei, J., Logeswaran, L., Lee, M., & Jurgens, D. (2024). *When "A Helpful Assistant" Is Not Really Helpful: Personas in System Prompts Do Not Improve Performances of Large Language Models.* arXiv preprint arXiv:2311.10054.