Agent-Centric Projection of Prompting Techniques and Implications for Synthetic Training Data for Large Language Models

Dhruv Dhamani[a] and Mary Lou Maher[b]

University of North Carolina, Charlotte, U.S.A.

Keywords: Large Language Models (LLMs), Task-Oriented LLM System, Prompt Engineering, Large Language Model-Based Agent, LLM-Based Multi-Agent System, Synthetic Training Data, Artificial Intelligence in Problem Solving.

Abstract: Recent advances in prompting techniques and multi-agent systems for Large Language Models (LLMs) have produced increasingly complex approaches. However, we lack a framework for characterizing and comparing prompting techniques or understanding their relationship to multi-agent LLM systems. This position paper introduces and explains the concepts of linear contexts (a single, continuous sequence of interactions) and non-linear contexts (branching or multi-path) in LLM systems. These concepts enable the development of an agent-centric projection of prompting techniques, a framework that can reveal deep connections between prompting strategies and multi-agent systems. We propose three conjectures based on this framework: (1) results from non-linear prompting techniques can predict outcomes in equivalent multi-agent systems, (2) multi-agent system architectures can be replicated through single-LLM prompting techniques that simulate equivalent interaction patterns, and (3) these equivalences suggest novel approaches for generating synthetic training data. We argue that this perspective enables systematic cross-pollination of research findings between prompting and multi-agent domains, while providing new directions for improving both the design and training of future LLM systems.

1 INTRODUCTION

Large Language Models (LLMs) are a recent development in Generative Artificial Intelligence that can mimic human-like behavior (Park et al., 2023), especially in conversations (Cai et al., 2023). LLMs have also shown a kind of general intelligence (Radford et al., 2019; Yogatama et al., 2019). Central to harnessing the capabilities of LLMs is the concept of prompting, a strategy that significantly influences task performance by instructing LLMs in specific ways (Chen et al., 2023).

HYPOTHESIS AND GOALS: In this position paper, we hypothesize that viewing prompting techniques through a proposed agent-centric lens can help uncover structural equivalences between single-LLM prompting and multi-agent approaches. Our goal is to (1) introduce a unified framework for comparing these techniques, (2) develop and examine conjectures about their relationship, and (3) outline how this perspective can inform the generation of synthetic training data.

Consider a simple math problem. When we directly prompt an LLM, "What is 13×27?", we might receive a single numeric answer. However, when we ask, "Let's solve this step by step: what is 13×27?", we explicitly prompt for intermediate reasoning plus the final result (Kojima et al., 2023; Yu et al., 2023). While both prompts seek the same final answer, are they both still the same problem if one has a different "correct" answer?

Another approach to improving end-task performance when using LLMs has been to incorporate "reasoning" (OpenAI, 2024). The model outputs arbitrarily long "reasoning traces" before responding to the prompts. These traces are sequences of natural language statements like "Let's first understand the input and output formats". OpenAI o1 is a single large language model or agent.

If we simply added role identifiers before each statement - "Analyst: Let's first understand the input and output formats" - would it suddenly qualify as a multi-agent system?

[a] https://orcid.org/0009-0003-8226-7621
[b] https://orcid.org/0000-0002-4150-0322

1254

What about approaches where an LLM analyzes problems from multiple perspectives in separate conversations before merging all perspectives together in another conversation (Saha et al., 2023)? Is each conversation a different "agent" performing a subtask? Is this a multi-agent system?

We argue that these questions can be systematically addressed by viewing prompting techniques through an agent-centric lens. By developing the concepts of linear and non-linear contexts in LLM systems, we shed light on possible connections between single-LLM prompting techniques and multi-agent systems. We discuss implications for the future of LLM systems, from enabling cross-pollination of research findings between prompting techniques and multi-agent systems to suggesting novel approaches for generating synthetic training data that could enhance capabilities in both domains.

To develop this argument, we first establish foundational definitions and examine previous work on prompting techniques and task-oriented LLM systems (§ 2). Building on these foundations, we present our framework for agent-centric projection and explore its implications for both system design and training (§ 3). We conclude by discussing the larger impact of this perspective on future research in LLM systems (§ 4).

2 DEFINITIONS AND PRIOR WORK

A task-oriented LLM system is a Large Language Model (LLM) system configured to perform specific tasks, rather than open-ended conversations[1]. Such systems have shown promise in complex tasks such as software development (Hong et al., 2023), where the system must manage multiple rounds of interaction, maintain context throughout iterations, and often collaborate with other systems or agents to complete the task.

We begin by defining a minimal task-oriented LLM system (§ 2.1), with particular attention to how such systems manage context across multiple interactions. We then examine prompting techniques (§ 2.2), focusing on how different approaches to prompting lead to different patterns of context creation and management. These patterns form the basis for our novel concepts of linear and non-linear contexts, which enable an agent-centric projection of prompting techniques.

2.1 Minimal Task-Oriented LLM System

A minimal task-oriented LLM system is a minimal LLM system that can be instructed to solve tasks. Thus, we start by defining a minimal LLM system.

Large Language Models are auto-regressive models that accept input tokens and use them as history (often referred to as context), to compute probabilities of all tokens in their vocabulary as the next token. We can sample from this probability distribution using a sampling/decoding algorithm to generate text. This process is then repeated until the LLM predicts a special token, or a special sequence of tokens, that marks the end of the text (Feuerriegel et al., 2023)[2].

We call this a bare-bones LLM system (see Figure 1) as it contains the minimal components needed for text generation, without additional components to help with context management. Every time an LLM is prompted with context C_n, it generates a response R_n that would need to be stored in context C_{n+1} for the next prompt, assuming multiple rounds of instruction and response generation are required.
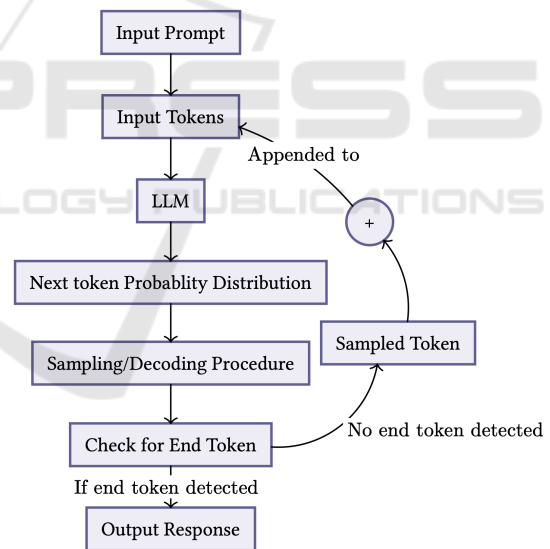

Figure 1: A bare-bones LLM system.

For systems oriented towards solving even moderately complex tasks, context management becomes quickly cumbersome. For example, in (Saha et al., 2023), the authors describe a system in which m branches are created from a LLM response to a

---

[1]In (Xi et al., 2023), the authors describe task-oriented deployments of LLM-based agents, which we generalize to simply task-oriented LLM systems

[2]The Generative AI system description in (Feuerriegel et al., 2023) includes any UI components as part of the Generative AI system, and we use a modified definition that only includes the language model and sampling/decoding procedure here.

prompt C_n, producing a set of m responses $r = \{R_{n_1}, R_{n_2}, \ldots, R_{n_m}\}$. Then, they take all of r, and transform it into a prompt C_{n+1}, in which they instruct the LLM to merge all responses in r into a single response R_{n+1}, which potentially needs to be stored in context C_{n+2} for the next prompt. A similarly complex system is described in (Ning et al., 2023), and we will examine more examples in our discussion of prompting techniques.

If we define a minimal LLM system without describing how context is managed, it would be too difficult to compare different systems and apply learnings from one researched system to another. Because of this, we include a description of a minimal context management subsystem within our definition of a minimal task-oriented LLM system.

Specifically, we include a context store CS. Initially, the context store CS is empty and the first time an LLM is provided with context C_1 to generate R_1, both the prompt and the response are permanently appended to CS. For all future requests, the LLM is first provided with a sliding window of content from CS as context, to which it appends the prompt C_n to generate R_n. Once the response R_n is generated, both the prompt and the response are permanently appended to CS. The model then closely matches messaging: the context store CS acts as chat history, users send messages, and the LLM responds.
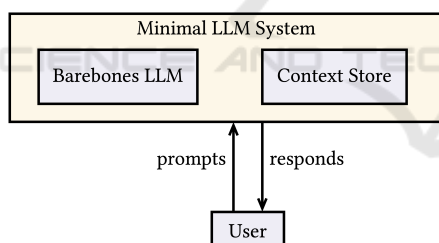

Figure 2: A minimal LLM system that includes a context store.

We discuss the implications of this composition of a minimal LLM system in § 3.2.

2.2 Prompting Techniques

Prompting refers to the act of constructing and providing input text (a prompt) to an LLM. In the context of task-oriented LLM systems, prompt engineering can be defined as iteratively creating and adapting a prompt for a given LLM and task pair.

The way an LLM is prompted significantly affects task performance (Nori et al., 2023; Savage et al., 2023). There are many surprising results in this area, such as letting an LLM know that solving a task "is

very important to my career" can improve task performance (Li et al., 2023).

Such results can be explained by research such as (Hendel et al., 2023), which shows that in-context learning creates task vectors or representations within the LLM that increase the probability of correct task completion. Other research has shown that it is possible to "search" for prompts that are more likely to lead to success, analogous to finding task vectors that are more likely to lead to success. In (Zou et al., 2023), the authors were able to procedurally find adversarial prefixes, which, when added to prompts, result in LLMs breaking their alignment and engaging in unsafe behavior.

All of these are examples of modifying the prompt without changing the actual task/problem definition, to make the successful completion of the intended task more likely. However, researchers are prone to modifying the prompts in a manner that changes the task, rather than modifying the prompts in a manner that improves task performance.

For example, when using Chain-of-Thought prompting (Wei et al., 2023)[3], or when asking LLMs to think step-by-step (Kojima et al., 2023) – the task meaningfully changes. It goes from instructing LLMs to give me an answer now to asking it to first plan out a solution, and then share an answer. This is a different task being solved, even though the final deliverable (the answer) is the same. It should be a given that LLMs have different capabilities for different tasks.

This is not to say that we shouldn't instead solve equivalent tasks that LLMs are more suitable to, but that it is problematic to have prompt modification (that leaves instructions/task definition intact) to instruction modification in the same category. Thus, we make the distinction between prompt engineering and instruction engineering:

- PROMPT ENGINEERING: The act of modifying the prompt without changing the actual task/problem definition or adding relevant knowledge/information, to make the successful completion of the intended task more likely. We restrict the addition of relevant knowledge/information to LLM augmentation to avoid an overlap.

- INSTRUCTION ENGINEERING: The act of modifying the prompt in a manner that changes the task/problem to an equivalent task/problem that the LLMs are more suitable for, such that the final deliverable (the answer) is the same.

---

[3]As described in the paper, one is to also provide in-context examples, but this is unnecessary, as shown in (Kojima et al., 2023)

In (Besta et al., 2023), the authors describe a taxonomy of techniques to improve reliability in task-oriented text generation:

- INPUT-OUTPUT. The LLM is directly being instructed to respond with the result of a prompted task.

- INPUT-OUTPUT WITH ADDITIONAL STEPS. The LLM is instructed to perform additional steps before or after generating a result for a prompted task, such as reflecting on its response and refining it, or creating a plan (Madaan et al., 2023; Wei et al., 2023).

- SINGLE INPUT-MANY OUTPUT.[4] The LLM is passed the same input prompt multiple times, and elaborate mechanisms are used to choose the final answer (Wang et al., 2022).

- INPUT WITH NON-LINEAR INTERMEDIARY STEPS.[5] LLM branches into multiple paths (through variations of an input prompt), generating multiple responses as additional steps, and then merges them into a single response (Saha et al., 2023; Ning et al., 2023).

- TREE OF THOUGHTS. An elaborate method described in (Yao et al., 2023), where many intermediate thought branches are explored, backtracked, and pruned until a final answer is settled on.

- GRAPH OF THOUGHTS. An elaborate method described in (Besta et al., 2023), where intermediate thoughts are modeled as a connected graph, and the LLM traverses the graph to settle on a final answer.

The way these techniques manage context differs significantly, from linear interactions to branching paths of thought. In the next section, we introduce the concepts of Linear and Non-Linear contexts to formalize these differences, and show how this formalization enables an Agent-centric projection of prompting techniques with potential implications for synthetic training data generation.

3 FRAMEWORK AND CONJECTURES

In research and practice, LLM systems exhibit different patterns in how they manage context and generate responses. We argue that these patterns can be understood through a theoretical framework that connects

---

[4]Referred to as Multiple CoTs in (Besta et al., 2023)

[5]This is not described in (Besta et al., 2023)

prompting techniques with multi-agent systems, revealing opportunities for improving both system design and training. In this section, we first introduce a formal categorization of context management patterns in LLM systems (§ 3.1). Building on this foundation, we develop an agent-centric projection of prompting techniques (§ 3.2) that reveals deep connections between seemingly disparate areas. Finally, we explore how this unified perspective suggests novel approaches to synthetic training data generation (§ 3.3), with potentially far-reaching implications for improving LLM capabilities.

3.1 Linear and Non-Linear Context in LLM Systems

To formally characterize how task-oriented LLM systems manage context and generate responses, we develop a framework based on message flow patterns. Building on the minimal task-oriented LLM system concept (§ 2.1), we analyze how the context store maintains sequences of messages $M = \{(C_n, R_n)\}_{n=1}^{N}$, where each response R_n is generated using all previous context-response pairs.

Using this foundation, we propose a method for classifying prompting techniques and their resulting task-oriented LLM systems into two categories based on their context management patterns.

PROMPTING TECHNIQUES WITH LINEAR CONTEXT – where there exists exactly one continuous sequence of messages $M = \{(C_n, R_n)\}_{n=1}^{N}$ that contains all generated messages and input contexts in the correct chronological order.

All Input-Output and Input-Output with additional steps techniques (as described in § 2.2) can be classified as having a linear context, as they all involve a single continuous sequence of messages.

For example, consider Self-Refine (Madaan et al., 2023), where each response is iteratively refined using all previous context-response pairs until a stop condition is met.

PROMPTING TECHNIQUES WITH NON-LINEAR CONTEXT – where there cannot always be one continuous sequence of messages that contains all input context and generated messages in the correct chronological order. Instead, there can be multiple branches of conversation possible, each with its own continuous sequence of messages $\{M_1, M_2, \ldots, M_n\}$.

All single input-many output, input with non-linear intermediary steps, tree of thought, and graph of thought techniques, (as described in § 2.2) can be classified as having a non-linear context, as they all potentially involve sequences of conversation M.
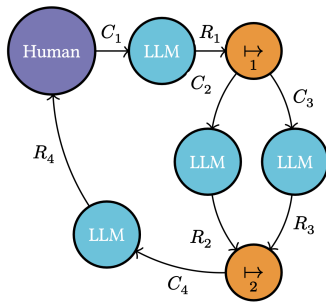
For example, consider a simplified version of

Figure 3: An example of a prompting technique with non-linear context.

BRANCH-SOLVE-MERGE, first described in (Saha et al., 2023), and as visualized in Figure 3. The figure depicts a task-oriented LLM system that helps the user (the human) make decisions. First, the human first instructs the system to make a decision. The system uses the instructions to create an input context for an LLM (context C_1) and uses it to generate a response R_1. R_1 is then used to create two new prompts (via an algorithmic transformation depicted in the figure as $\mapsto_1$), one in which the LLM is tasked with reflecting on the drawbacks of this decision (in context C_2) and another where the LLM is tasked with reflecting on the benefits of this decision (in context C_3). Finally, another prompt is created where both reflections (responses R_2 and R_3) are considered (using another algorithmic transformation $\mapsto_2$) to create a new prompt (context C_4) which is used to generate a final decision within response R_4. R_4 is then reported to the user as the final decision.

As long as any task-oriented system is using LLMs, it will always have one or more continuous streams of messages M as described. This means that all task-oriented LLM systems and all prompting techniques can be classified as having either linear or non-linear contexts.

This fundamental dichotomy between linear and non-linear contexts provides a powerful lens through which to analyze LLM systems. As we will show in the next section, it reveals surprising connections between prompting techniques and multi-agent systems that can inform both system design and training approaches.

3.2 Agent-Centric Projection of Prompting Techniques

In the previous section (§ 3.1), we classify all prompting techniques and all resulting task-oriented LLM systems they bore into either having a linear or non-linear context. This decision and the overall definition have the following implications:

- Research on techniques for reliable, task-oriented text generation that involve linear context can be modeled as a kind of two-agent system (the human instructing the LLM being the second agent, as we also see in (Xi et al., 2023; Wu et al., 2023)).
- Research on techniques for reliable, task-oriented text generation that involves non-linear context can be modeled to be a kind of multi-agent system, where each "branch" of conversation M can be considered to have occurred with a different agent.
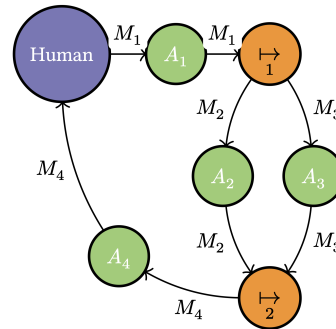

Figure 4: The prompting technique from Figure 3 is modeled as a multi-agent system.

For example, In Figure 4, we show how the prompting technique from Figure 3 can be modeled as a multi-agent system. Each continuous linear sequence of messages $M_1 = \{C_1, R_1\}$, $M_2 = \{C_2, R_2\}$, $M_3 = \{C_3, R_3\}$, and $M_4 = \{C_4, R_4\}$ can be considered to have occurred with a different agent. Using this approach, we can model any prompting technique with non-linear context as a multi-agent system.

It would also help to note that each continuous sequence of messages M_n in Figure 4 is essentially a minimal task-oriented LLM system, as described in § 2.1. This means that we can substitute each such minimal system with a more complicated task-oriented system if needed.

In Figure 5, we show a more realistic example of a multi-agent system, designed to replicate the behavior of the prompting technique in Figure 3. Here, the major changes are that the agents communicate with each other using tools, meaning all communication is bidirectional (say, if an agent wants to ask a clarifying question) and that the algorithmic transformations $\mapsto_1$ and $\mapsto_2$ are now present each as a tool available to Agents A_1 and A_4 respectively. This system may behave exactly like the system in Figure 3 most of the time, but may prove to be more resilient to unexpected circumstances, as each component is more "intelligent".
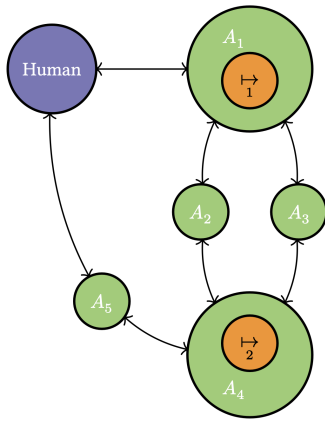
Figure 5: A more realistic projection of the prompting technique from Figure 3 as a multi-agent system.

As all prompting techniques can be projected to such multi-agent systems, we can conjecture that:

Conjecture 1. *Results from prompting techniques involving non-linear context can predict similar results from multi-agent systems designed to replicate the same behavior.*

This projection or view allows us to generalize all such techniques and apply learnings from one technique to another, and even learnings from multi-agent systems. For example, if new LLM-based multi-agent collaboration research shows that "A Process Supervising Agent is all you Need", then we can immediately apply that result to the prompting technique described in BRANCH-SOLVE-MERGE from (Saha et al., 2023) – by viewing the "process supervising agent" work via a non-linear context lens as illustrated in Figure 3, and then adding the BRANCH-SOLVE-MERGE "nodes and connections".

But it does not end there – because of how flexible natural language is, all non-linear contexts can also be projected to linear contexts. For example, say four agents engage in adversarial interaction as described in (Xi et al., 2023) (§ 4.2.2), where they argue about a decision until they reach a consensus. The benefit of this interaction paradigm is that each agent can be instructed to look at the problem from various perspectives.

This interaction can be elicited within a linear context, where the LLM is prompted with the same decision-making problem but with additional instructions to share a turn-by-turn dialogue where four individuals argue about the decision until they reach a consensus. This has been demonstrated in (Wang et al., 2024), where a single LLM instance is prompted to produce a transcript of multiple personas (agents) interacting with each other to solve a task. The authors call this "Solo Per-

formance Prompting". Their results show that this technique—essentially converting non-linear context (multiple agents collaborating) into linear context (a dialogue transcript)—shows performance gains comparable to those achieved by multi-agent systems on other tasks ((Wang et al., 2024) does not directly compare to multi-agent systems).

In (Dong et al., 2024) the authors describe a similar approach minus the "dialogue", where multiple roles (analyst, coder, tester, etc.) are simulated by a single LLM with linear context (Dong et al., 2024) (§2.2, Eq. 1). The paper shows how this approach outperforms baseline and advanced prompting techniques (such as CoT).

Conjecture 2. *Performance improvements achieved through multi-agent system architectures can be at least partially replicated using single-LLM prompting techniques[6] that simulate equivalent multi-agent interaction patterns within a linear context.*

3.3 Implications for Synthetic Training Data

Recent work has demonstrated that synthetic data can effectively enhance model capabilities in various applications, from structured information extraction (Josifoski et al., 2023) to visual question answering (Su et al., 2024).

A key insight stems from an apparent paradox in LLM systems: while all LLMs are trained on "linear context" (sequential text), research and practice show that "non-linear context" approaches—such as advanced prompting techniques and multi-agent interactions—are of significant interest and demonstrate superior task performance (Saha et al., 2023; Ning et al., 2023; Wei et al., 2023; Hong et al., 2023; Wu et al., 2023).

The previous subsection presents an argument for how techniques involving non-linear context can be projected to an equivalent technique utilizing linear context. This can have profound implications when you consider that all LLMs are trained on "linear context", i.e., trained on continuous sequences of text. If intermediate steps from advanced prompting techniques like BRANCH-SOLVE-MERGE are projected to linear contexts similar to Solo Performance Prompting (Wang et al., 2024) and Self-Collaboration (Dong et al., 2024) – then they can also be used as synthetic training data.

Interestingly, a recent approach called Stream of

---

[6]prompting techniques such as Solo Performance Prompting (Wang et al., 2024) and Self-Collaboration (Dong et al., 2024)

Search (SoS) (Gandhi et al., 2024) further underscores our perspective on using non-linear or suboptimal reasoning traces for training. SoS demonstrates that when LLMs are trained on branching, backtracking search trajectories—serialized into a linear textual format—they acquire stronger problem-solving capabilities and can even discover new strategies. These findings support Conjecture 3 below, illustrating how self-generated, "messy" intermediate steps can serve as valuable synthetic data to improve the performance of an LLM.

Conjecture 3. *Synthetically generated "self-collaboration" transcripts of successful task-solving attempts—whether derived from non-linear prompting techniques or multi-agent collaboration—when used as training data, improve LLM performance in both multi-agent systems and advanced prompting techniques targeting similar tasks.*

This idea can be extended further by using existing problems and their real-world deliverables, both intermediate and final, and generating simulated interactions between collaborators as synthetic data. Consider taking the requirements of a completed software project on GitHub, along with pull requests/issue commentary, commit messages, commit diffs in chronological order, and using LLMs to fabricate communication between collaborators – wouldn't the resulting manuscript, perhaps made to resemble a theater play script, be effective training data?

Thus, our proposed framework of linear and non-linear context (§ 3.1) along with the agent-centric projection of prompting techniques (§ 3.2) presents a lens that could lead to significant advancements in synthetic data generation.

4 CONCLUSIONS

4.1 Core Arguments

- PROMPT ENGINEERING AND INSTRUCTION ENGINEERING: Clearly differentiating the adjustment of the prompt without altering the actual task or the definition of the problem (prompt engineering) and modifying the task to an equivalent task[7] more suitable for LLM systems (instruction engineering) is essential to precise communication and understanding of research in this area.

- LINEAR AND NON-LINEAR CONTEXT: Prompting techniques and resulting task-oriented LLM

---

[7]one with the same final deliverable

systems can be classified into having either linear or non-linear context.

- AGENT-CENTRIC PROJECTION OF PROMPTING TECHNIQUES: We demonstrate approaches that allow prompting techniques with non-linear context to be understood as multi-agent systems and vice versa. This projection provides a framework for analyzing, comparing, and improving both prompting techniques and multi-agent system architectures.

4.2 Implications for Future Research

- CROSS-POLLINATION IN PROMPTING AND LLM-BASED MULTI-AGENT SYSTEMS: The agent-centric projection of prompting techniques may allow us to cross-pollinate research findings in these areas.

- SYNTHETIC TRAINING DATA GENERATION: Our core arguments suggest two novel approaches for generating high-quality synthetic training data for LLMs: (1) converting successful non-linear prompting traces into linear training data and (2) augmenting real-world task traces with synthetic agent collaboration artifacts. These approaches could provide structured, high-quality data specifically suited for training LLMs in multi-agent and complex reasoning tasks.

- REAL-WORLD APPLICATIONS AND ETHICAL CONSIDERATIONS: As these systems become more capable, their deployment in real-world scenarios becomes more feasible. With this comes the need for rigorous ethical considerations, especially concerning autonomy, decision making, and human-AI interaction.

By establishing the fundamental distinction between linear and non-linear contexts in prompting techniques, and using this to develop an agent-centric projection that reveals deep connections between prompting techniques and multi-agent systems. This framework leads to three key conjectures about the relationship between prompting techniques and multi-agent systems, suggesting that results from one domain can inform the other. Furthermore, we demonstrate how this unified perspective opens up novel approaches to synthetic training data generation, both through the conversion of non-linear prompting traces and through the augmentation of real-world task traces. Our position highlights the untapped potential in viewing prompting techniques through an agent-centric lens, providing concrete directions for improving both the design and training of future LLM systems.

REFERENCES

Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Gianinazzi, L., Gajda, J., Lehmann, T., Podstawski, M., Niewiadomski, H., Nyczyk, P., and Hoefler, T. (2023). Graph of Thoughts: Solving Elaborate Problems with Large Language Models. arXiv:2308.09687 [cs].

Cai, Z. G., Haslett, D. A., Duan, X., Wang, S., and Pickering, M. J. (2023). Does ChatGPT resemble humans in language use? arXiv:2303.08014 [cs].

Chen, B., Zhang, Z., Langren'e, N., and Zhu, S. (2023). Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review. *arXiv.org*.

Dong, Y., Jiang, X., Jin, Z., and Li, G. (2024). Self-Collaboration Code Generation via ChatGPT. *ACM Trans. Softw. Eng. Methodol.*, 33(7):189:1–189:38.

Feuerriegel, S., Hartmann, J., Janiesch, C., and Zschech, P. (2023). Generative AI. *Business & Information Systems Engineering*. arXiv:2309.07930 [cs].

Gandhi, K., Lee, D., Grand, G., Liu, M., Cheng, W., Sharma, A., and Goodman, N. D. (2024). Stream of Search (SoS): Learning to Search in Language. arXiv:2404.03683.

Hendel, R., Geva, M., and Globerson, A. (2023). In-Context Learning Creates Task Vectors. arXiv:2310.15916 [cs].

Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Zhang, C., Wang, J., Wang, Z., Yau, S. K. S., Lin, Z., Zhou, L., Ran, C., Xiao, L., Wu, C., and Schmidhuber, J. (2023). MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. arXiv:2308.00352 [cs].

Josifoski, M., Sakota, M., Peyrard, M., and West, R. (2023). Exploiting Asymmetry for Synthetic Training Data Generation: SynthIE and the Case of Information Extraction. arXiv:2303.04132.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. (2023). Large Language Models are Zero-Shot Reasoners. arXiv:2205.11916 [cs].

Li, C., Wang, J., Zhang, Y., Zhu, K., Hou, W., Lian, J., Luo, F., Yang, Q., and Xie, X. (2023). Large Language Models Understand and Can be Enhanced by Emotional Stimuli. arXiv:2307.11760 [cs].

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Welleck, S., Majumder, B. P., Gupta, S., Yazdanbakhsh, A., and Clark, P. (2023). Self-Refine: Iterative Refinement with Self-Feedback. *arXiv.org*.

Ning, X., Lin, Z., Zhou, Z., Wang, Z., Yang, H., and Wang, Y. (2023). Skeleton-of-Thought: Large Language Models Can Do Parallel Decoding. arXiv:2307.15337 [cs].

Nori, H., Lee, Y. T., Zhang, S., Carignan, D., Edgar, R., Fusi, N., King, N., Larson, J., Li, Y., Liu, W., Luo, R., McKinney, S. M., Ness, R. O., Poon, H., Qin, T., Usuyama, N., White, C., and Horvitz, E. (2023). Can Generalist Foundation Models Outcompete Special-Purpose Tuning? Case Study in Medicine. arXiv:2311.16452 [cs].

OpenAI (2024). Learning to Reason with LLMs.

Park, J., O'Brien, J. C., Cai, C. J., Morris, M., Liang, P., and Bernstein, M. S. (2023). Generative Agents: Interactive Simulacra of Human Behavior. *arXiv.org*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners.

Saha, S., Levy, O., Celikyilmaz, A., Bansal, M., Weston, J., and Li, X. (2023). Branch-Solve-Merge Improves Large Language Model Evaluation and Generation. arXiv:2310.15123 [cs].

Savage, T., Nayak, A., Gallo, R., Rangan, E., and Chen, J. H. (2023). Diagnostic Reasoning Prompts Reveal the Potential for Large Language Model Interpretability in Medicine. arXiv:2308.06834 [cs].

Su, X., Luo, M., Pan, K. W., Chou, T. P., Lal, V., and Howard, P. (2024). SK-VQA: Synthetic Knowledge Generation at Scale for Training Context-Augmented Multimodal LLMs. arXiv:2406.19593.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., and Zhou, D. (2022). Self-Consistency Improves Chain of Thought Reasoning in Language Models. *ArXiv*.

Wang, Z., Mao, S., Wu, W., Ge, T., Wei, F., and Ji, H. (2024). Unleashing the Emergent Cognitive Synergy in Large Language Models: A Task-Solving Agent through Multi-Persona Self-Collaboration. arXiv:2307.05300.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. (2023). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A. H., White, R. W., Burger, D., and Wang, C. (2023). AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. arXiv:2308.08155 [cs].

Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., Zheng, R., Fan, X., Wang, X., Xiong, L., Zhou, Y., Wang, W., Jiang, C., Zou, Y., Liu, X., Yin, Z., Dou, S., Weng, R., Cheng, W., Zhang, Q., Qin, W., Zheng, Y., Qiu, X., Huang, X., and Gui, T. (2023). The Rise and Potential of Large Language Model Based Agents: A Survey. arXiv:2309.07864 [cs].

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. (2023). Tree of Thoughts: Deliberate Problem Solving with Large Language Models. *arXiv.org*.

Yogatama, D., d'Autume, C. d. M., Connor, J., Kocisky, T., Chrzanowski, M., Kong, L., Lazaridou, A., Ling, W., Yu, L., Dyer, C., and Blunsom, P. (2019). Learning and Evaluating General Linguistic Intelligence. arXiv:1901.11373 [cs, stat].

Yu, Z., He, L., Wu, Z., Dai, X., and Chen, J. (2023). Towards Better Chain-of-Thought Prompting Strategies: A Survey. *arXiv.org*.

Zou, A., Wang, Z., Kolter, J. Z., and Fredrikson, M. (2023). Universal and Transferable Adversarial Attacks on Aligned Language Models. arXiv:2307.15043 [cs].