

# Meta-Prompting:

## Enhancing Language Models with Task-Agnostic Scaffolding

Mirac Suzgun

Stanford University\*

msuzgun@stanford.edu

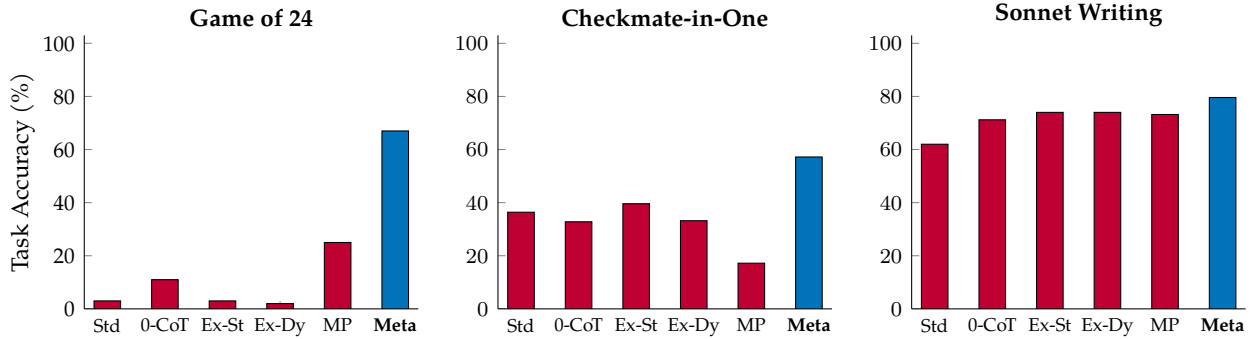
Adam Tauman Kalai

OpenAI\*

adam@kal.ai

### Abstract

We introduce meta-prompting, an effective scaffolding technique designed to enhance the functionality of language models (LMs). This approach transforms a single LM into a multi-faceted conductor, adept at managing and integrating multiple independent LM queries. By employing high-level instructions, meta-prompting guides the LM to break down complex tasks into smaller, more manageable subtasks. These subtasks are then handled by distinct “expert” instances of the same LM, each operating under specific, tailored instructions. Central to this process is the LM itself, in its role as the conductor, which ensures seamless communication and effective integration of the outputs from these expert models. It additionally employs its inherent critical thinking and robust verification processes to refine and authenticate the end result. This collaborative prompting approach empowers a single LM to simultaneously act as a comprehensive orchestrator and a panel of diverse experts, significantly enhancing its performance across a wide array of tasks. The zero-shot, task-agnostic nature of meta-prompting greatly simplifies user interaction by obviating the need for detailed, task-specific instructions. Furthermore, our research demonstrates the seamless integration of external tools, such as a Python interpreter, into the meta-prompting framework, thereby broadening its applicability and utility. Through rigorous experimentation with GPT-4, we establish the superiority of meta-prompting over conventional scaffolding methods: When averaged across all tasks, including the Game of 24, Checkmate-in-One, and Python Programming Puzzles, meta-prompting—augmented with a Python interpreter functionality—surpasses standard prompting by 17.1%, expert (dynamic) prompting by 17.3%, and multipersona prompting by 15.2%.<sup>1</sup>



**Figure 1:** Enhancing GPT-4 with meta-prompting. In this study, we introduce and examine the effectiveness of meta-prompting, contrasting it with a range of zero-shot prompting techniques, including standard zero-shot (Std), zero-shot chain-of-thought (0-CoT; Kojima et al. (2022)), generic and dynamic expert (Ex-St and Ex-Dy; Xu et al. (2023)), and multipersona (MP; Wang et al. (2023)). Our research demonstrates that meta-prompting, particularly when combined with a Python interpreter, significantly improves overall accuracy and robustness in GPT-4 across a variety of tasks.

\*Work done while at Microsoft Research New England.

<sup>1</sup>The data, prompts, and the model outputs are all available at <https://github.com/suzgunmirac/meta-prompting>.

# 1 Introduction

The latest generation of language models (LMs)—notably, GPT-4 (OpenAI, 2023), PaLM (Anil et al., 2023), and LLaMa (Touvron et al., 2023)—have expanded the boundaries of natural-language processing and generation. These large-scale models can tackle a wide spectrum of tasks, ranging from writing Shakespearean sonnets about hedgehogs to summarizing intricate medical reports and solving competition-level programming puzzles. Despite their versatility, these models are not infallible; they sometimes generate responses that are inaccurate, misleading, or conflicting. As the operational costs of these models become more affordable, it becomes natural to ask whether one might use scaffolding systems and leverage multiple LM queries to not only refine but also to enhance the accuracy and robustness of these model outputs.

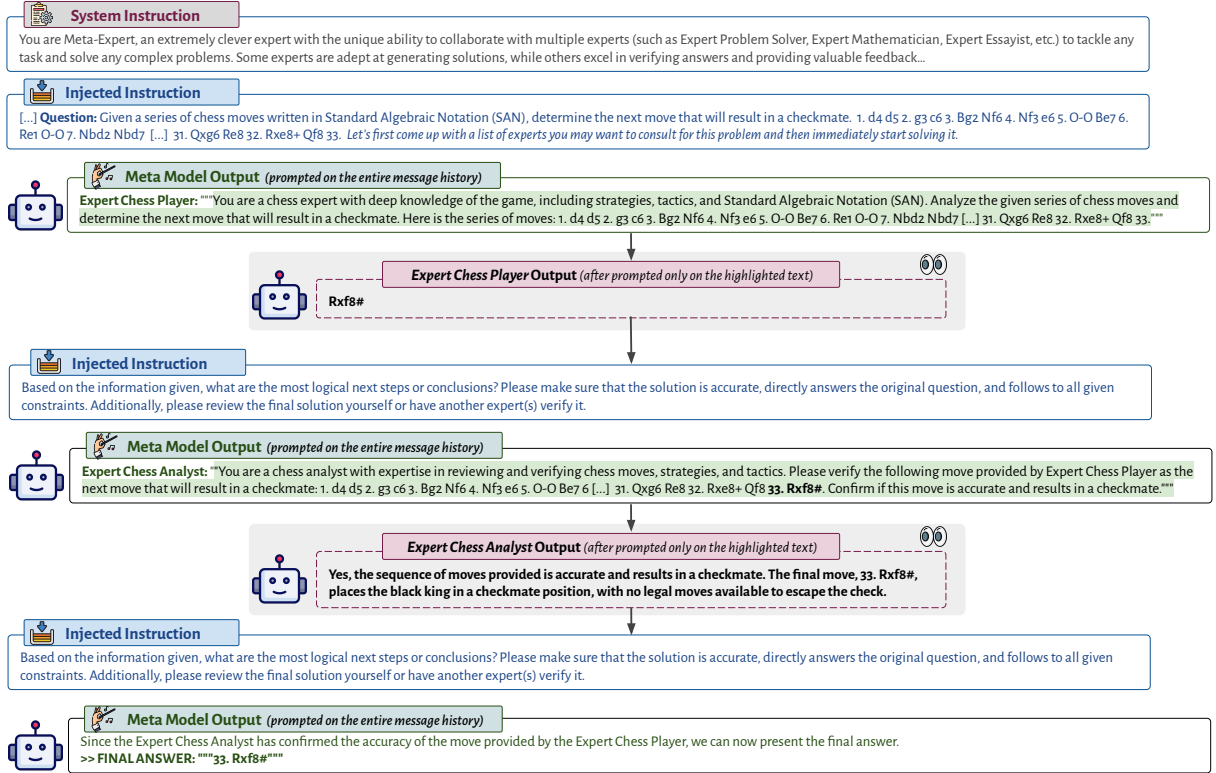
In this work, we introduce a new technique for enhancing the functionality and performance of LMs, called meta-prompting. It involves constructing a high-level “meta” prompt that instructs an LM to: (i) break down complex tasks or problems into smaller, manageable pieces; (ii) assign these pieces to specialized “expert” models with proper and detailed natural-language instructions; (iii) oversee the communication between these expert models; and (iv) apply its own critical thinking, reasoning, and verification skills throughout the process. When presented with a query, the LM, effectively prompted under meta-prompting, serves as a conductor. It produces a message history—a narrative, if you will—comprising the responses from various expert models. The LM is originally responsible for generating the conductor’s portion of this history, which includes the selection of experts and the formulation of specific instructions for them. However, the same LM doubles itself as these independent experts as well, generating outputs based on the expertise and information chosen by the conductor for each particular query.

This approach allows for a single, uniform LM to maintain a coherent line of reasoning while also tapping into a variety of expert roles. The use of dynamically selected contexts for prompting these experts introduces fresh perspectives into the process, while the conductor model retains a bird’s-eye view of the entire history and coordination. This method, therefore, enables a single black-box LM to function effectively as both a central conductor and a diverse panel of experts to produce more accurate, reliable, and coherent responses.

Our proposed meta-prompting technique combines and expands upon various prompting ideas introduced by recent studies—including, *high-level planning and decision-making* (Yao et al., 2023b; Sun et al., 2023; Hao et al., 2023a), *dynamic persona assignment* (Xu et al., 2023; Wang et al., 2023), *multi-agent debating* (Du et al., 2023; Zhuge et al., 2023), *self-debugging and self-reflection* (Schick et al., 2023b; Liu et al., 2023a; Gou et al., 2023; Madaan et al., 2023; Shinn et al., 2023). A key aspect of meta-prompting is its *task-agnostic* nature. Unlike traditional scaffolding methods that require specific instructions or examples tailored to each task, meta-prompting employs the same set of high-level instructions across various tasks and inputs. This universality is particularly beneficial for users who might find it cumbersome to provide detailed examples or specific guidance for every distinct task. For instance, in responding to a one-off request like “Write a Shakespearean sonnet about selfies,” the user would not need to supply examples of high-quality neoclassical poems. The meta-prompting approach elevates the utility of language models by offering a broad, flexible framework without compromising on specificity or relevance. Additionally, to demonstrate the versatility and integration capabilities of meta-prompting, we have enhanced our system with the functionality to invoke a Python interpreter. This allows for an even more dynamic and comprehensive application of the technique, further extending its potential to address a wide array of tasks and queries effectively.

We provide an illustrative visualization of a meta-prompting session in Figure 2. It depicts how the Meta Model—our technical term for the central controlling LM (a.k.a. the conductor)—intersperses its own output with inputs and outputs from various specialized expert models or code executions. Such a configuration makes meta-prompting a nearly universal tool. It allows for the consolidation of various LM interactions and computations into a single, coherent narrative. What sets meta-prompting apart is that it leaves the decision of which prompts to use and which code snippets to execute to the discretion of the LM itself.

In our comprehensive experiments, which primarily utilize GPT-4 as the foundational LM, we compare the efficacy of meta-prompting against other task-agnostic scaffolding methods. Our findings reveal that meta-prompting not only enhances overall performance but often leads to state-of-the-art results across a diverse range of tasks. Its flexibility is noteworthy: The conductor model has the capability to call upon



**Figure 2:** An example meta-prompting history, where the prompts have been shortened for illustrative purposes. The history is initialized by a question provided by a user. Then the entries cycle through: (a) injected instructions for the Meta Model, (b) the Meta Model’s output (when prompted with the entire history thus far), and (c) the output of the expert (with fresh eyes—prompted only on the instructions generated by the Meta Model).

expert models (basically itself, albeit with fresh instructions) for performing a variety of functions. These functions might include critiquing earlier outputs, selecting specific personas for certain tasks, refining generated content, and ensuring that the final outputs meet the desired criteria in both substance and form. This approach shows a marked improvement over several existing methods, as demonstrated in Figure 1.

The core contribution of this work is the introduction of a task-agnostic scaffolding system that leverages a single LM. This LM not only carries forward the thread of the task but also dynamically selects and instructs expert models appropriate for each specific task. The effectiveness of this system is showcased across various benchmarks, including the Game of 24 (Yao et al., 2023a), Checkmate-in-One from the BIG-Bench suite (BIG-Bench authors, 2023), and our novel task of “Shakespearean Sonnet Writing.” Overall, our empirical results underscore the versatility and robustness of meta-prompting in enhancing LM performance.

## 2 Meta Prompting

**Intuition and Abstract Overview.** The modus operandi of meta-prompting is to use a model<sup>2</sup> to coordinate and execute multiple independent inquiries and subsequently synthesize their responses to render a final response. This mechanism, in principle, endorses an ensemble approach, drawing from the strength and diversity of independent specialized models to collaboratively address and tackle multifaceted tasks or problems. We posit that while a single, general-purpose model might deliver valuable and useful insights into generic queries, combining the perspectives and conclusions of multiple domain-specific models (which we also refer to as *experts*) has the potential to yield more comprehensive, robust, and accurate solutions.

<sup>2</sup>Our use of the term model refers to the application of an LM with certain prompt templates to play a specified “role.” We typically only use a single LM (e.g., GPT-4) to implement all the models in an execution.

Central to our meta-prompting strategy is its shallow hierarchical configuration, where a single model—called the “Meta Model”—emerges as the principal entity of authority. This prompting structure is reminiscent of an orchestra, wherein the conductor’s role is mirrored by the Meta Model and each musician corresponds to a distinct domain-specific model. Just as a conductor harmonizes multiple musical elements to craft a beautiful melody, the Meta Model combines solutions and insights from a range of models to provide an accurate and comprehensive answer to an intricate problem or task.

Conceptually, a domain-specific expert within our framework can take diverse forms, such as a finetuned LM tailored to perform a particular task, a specialized API equipped to handle specific domain-related inquiries, or even computational tools like calculators or a Python interpreter that can perform arithmetic calculations or write and execute code. These experts, despite their varying functionalities, are directed and unified under the supervision of the Meta Model.

Under our setup, experts can be called only by the Meta Model. They cannot directly interact or communicate with each other, though the Meta Model can choose to share some text from or combine the insights of various experts when interacting with a new expert. This restriction is made to simplify the communication between the experts and to put the Meta Model at the center of the operation.

**Notation and Terminology.** Before we delve into the specific steps involved in meta-prompting, we establish some notation and terminology. We let  $\mathbb{S}$  denote the set of finite strings, with  $\emptyset$  representing the empty string. We use  $x \in \mathbb{S}$  to refer to a test-time query, which can be a task or a problem described in natural language. A crucial element of meta-prompting is the fixed language model, denoted as LM, which operates from  $\mathbb{S}$  to  $\mathbb{S}$ . This model, like GPT-4, takes an input text (a prompt history that may include a list of previous messages, symbolized by  $\mathcal{H}$ ) and produces a corresponding output (i.e., response). We also introduce specific template functions:  $t_{\text{init}}$ ,  $t_{\text{mid}}$ , and  $t_{\text{exp}}$ , each mapping from  $\mathbb{S}$  to  $\mathbb{S}$ ; each takes a string input and formats it according to a predefined template. Specifically,  $t_{\text{init}}$  and  $t_{\text{mid}}$  are used to format text for the history given to the Meta Model, while  $t_{\text{exp}}$  wraps the output of the Meta Model in a prompt suitable for an expert model. Furthermore, we have two string extractors,  $e_{\text{exp}}$  and  $e_{\text{ret}}$ , each mapping from  $\mathbb{S}$  to  $\mathbb{S}$ . These extractors are designed to retrieve a substring that is enclosed within specific delimiters, returning the first matching segment in cases where multiple segments are present. The symbol  $\oplus$  is used to represent string concatenation. Lastly, we introduce a specific string referred to as error  $\in \mathbb{S}$ , which is designed to denote an error message in the process.

**Algorithmic Procedure.** Algorithm 1 provides pseudocode of our proposed meta-prompting approach. We further provide a conceptual overview of the procedure below:

---

**Algorithm 1** Meta Prompting

---

**Input:**  $\text{LM} : \mathbb{S} \rightarrow \mathbb{S}; x, \text{error} \in \mathbb{S}; T \in \mathbb{N}; t_{\text{init}}, t_{\text{mid}}, t_{\text{exp}}, e_{\text{exp}}, e_{\text{ret}} : \mathbb{S} \rightarrow \mathbb{S}$

---

```

1:  $\mathcal{H}_1 \leftarrow t_{\text{init}}(x)$ 
2: for  $t \in [1, \dots, T]$  do
3:    $y_t \leftarrow \text{LM}(\mathcal{H}_t)$ 
4:   if  $e_{\text{exp}}(y_t) \neq \emptyset$  then                                      $\triangleright$  Meta Model provided expert instructions
5:      $\text{prompt} \leftarrow t_{\text{exp}}(e_{\text{exp}}(y_t))$ 
6:      $z_t \leftarrow \text{LM}(\text{prompt})$ 
7:      $\mathcal{H}_{t+1} \leftarrow \mathcal{H}_t \oplus t_{\text{mid}}(z_t)$ 
8:   else if  $e_{\text{ret}}(y_t) \neq \emptyset$  then                                    $\triangleright$  Meta Model returned a final answer
9:     return  $e_{\text{ret}}(y_t)$ 
10:  else                                                                  $\triangleright$  Meta Model formatting error
11:     $\mathcal{H}_{t+1} \leftarrow \mathcal{H}_t \oplus \text{error}$ 
12:  end if
13: end for

```

---

1. **Transforming the Input:** Using the transformation function  $t_{\text{init}}$ , the raw query is placed in a suitable template followed by initial instructions to the Meta Model.

2. **Loop Iteration:**

- (a) **Prompting the Meta Model:** The current message list, namely  $\mathcal{H}_t$ , guides the Meta Model’s next action—either directly addressing the query or consulting a domain-specific expert.
- (b) **Engaging Domain-Specific Expert Models:** If the Meta Model does not return a result, it can conjure any expert and give it instructions, which are extracted from its output using  $e_{\text{exp}}$ . This process is isolated though: Each expert only sees what the Meta Model chooses to share with them, and responds accordingly. For instance, if a problem pertains to mathematics and history, the Meta Model might consult a mathematics expert for a calculation and a history expert for historical context. The output of the expert is extracted and additional instructions are appended, all using the  $t_{\text{mid}}$  template.
- (c) **Returning the Final Response:** If the Meta Model’s response contains a final answer (highlighted by distinct special markers), the solution is extracted using  $e_{\text{ret}}$  and returned.
- (d) **Error Handling:** In cases where the model response  $y_t$  contains neither a final answer nor a call to an expert model, an error message appended to the message list  $\mathcal{H}_t$ . This ensures that our procedure is robust and can handle unexpected outputs.

**Meta and Expert Model Specifications.** In our setup, we employ the same LM, such as GPT-4, to function in both Meta and Expert capacities. Their roles are distinguished by their respective model instructions in their prompts, with the Meta Model adhering to a set of instructions provided in Figure 3, and the expert models following separate instructions dynamically determined by the Meta Model at inference time .

### 3 Experimental Setup

#### 3.1 Baselines

We compare meta-prompting with the task-agnostic, zero-shot versions of the following prompting methods:

- **Standard prompting:** This represents our most basic baseline wherein an LM is asked to directly yield a response without any specific guiding input-output exemplars or any additional guiding instructions, besides the task description already included in the input query.
- **Zero-shot CoT prompting** (Kojima et al., 2022): Drawing inspirations from the chain-of-thought method of Wei et al. (2022b), this zero-shot prompting approach simply appends “Let’s think step by step” to the input query, encouraging the model to have a more deliberative and iterative cognition before addressing the problem or task at hand.
- **Expert prompting** (Xu et al., 2023): This prompting approach functions through a two-step process: It first crafts an expert identity tailored to align with the specific context of the input query. It then integrates this generated expert profile into the input to generate a well-informed and authoritative response. In our experiments, we consider two versions of expert prompting, namely (a) *static* (i.e., *generic*) and (b) *dynamic* (i.e., *adaptive*); the former uses a fixed and generic expert description, whereas the latter adaptively designs a new expert identity for each input query.
- **Multi-persona prompting** (Du et al., 2023): Also known as solo-performance prompting (SPP), this method instructs an LM to perform the following: (i) Propose a small ensemble of “personas” to address the specific task or problem at hand; (ii) let these personas engage in a collective dialogue, collaboratively generating potential solutions while extending feedback to one another and refining their answers; and (iii) synthesize all the available information and deliver a final response.

#### 3.2 Datasets and Tasks

To evaluate the efficacy of our proposed meta-prompting approach over other zero-shot prompting baselines, we consider a wide range of tasks and datasets that require various degrees of mathematical and algorithmic

reasoning, domain-specific knowledge, and literary creativity. These include:

- (a) The **Game of 24** from (Yao et al., 2023a) where the goal is to form an arithmetic expression whose value is 24 using each of four given numbers exactly once,
- Three BIG-Bench Hard (BBH; Suzgun et al. (2023b)) tasks—namely, (b) **Geometric Shapes**, (c) **Multi-Step Arithmetic Two**, and (d) **Word Sorting**—as well as one reasoning task directly obtained from the BIG-Bench suite (BIG-Bench authors, 2023), that is, (e) **Checkmate-in-One**;
- (f) **Python Programming Puzzles** (P3; Schuster et al. (2021)), a collection of challenging programming puzzles written in Python—with varying difficulty levels;
- (g) **Multilingual Grade School Math** (MGSM; Shi et al. (2023)), a multilingual version of the GSM8K dataset (Cobbe et al., 2021) with translations of a subset of examples into ten typologically diverse languages, including Bengali, Japanese, and Swahili;
- (h) **Shakespearean Sonnet Writing**, a novel task we created where the goal is to write a sonnet with strict rhyme scheme “ABAB CDCD EFEF GG,” containing the three provided words verbatim.<sup>3</sup>

### 3.3 Answer Extraction and Evaluation Protocols

As shown in Figure 3, the system instruction in our proposed meta-prompting method encourages the Meta Model to present its final answer in a specific format. This format, designed for consistent and unambiguous extraction, requires that the final answer is wrapped within triple quotes and preceded by a distinct marker (namely, “>FINAL ANSWER:”).

Once the final answer is extracted from the model and properly post-processed, we also need to evaluate its correctness.<sup>4</sup> Because we consider a wide range of tasks, there is not a single metric that allows us to measure accuracy across all. Depending on the nature and formulation of the task, we measure accuracy using one of the following three metrics:

- *Exact Match* (EM): Under this strict metric, the correctness of an answer is determined by its precise alignment with the ground-truth label(s). An answer is deemed correct only if it is identical to a provided reference.
- *Soft Match* (SM): This metric offers a more lenient approach than EM. For an answer to be deemed correct, it is sufficient for a ground-truth label to be present within the model’s output, regardless of any additional textual content.
- *Functionally Correct* (FC): This metric ascertains whether the answer is functionally correct, meaning that it adheres to task-specific constraints.

We use EM for Geometric Shapes, Multi-Step Arithmetic Two, and Checkmate-in-One; SM for MGSM and Word Sorting; and FC for Game of 24, Python Programming Puzzles, and Shakespearean Sonnet Writing.

### 3.4 Models and Inference

In our main experiments, we concentrate on **GPT-4** (gpt-4-32k), which is accessible through Microsoft’s Azure OpenAI Service. Additionally, in our supplementary experiments, we include **GPT-3.5** (gpt-35-turbo). Both GPT-3.5 and GPT-4 are models fine-tuned for following instructions, though GPT-4 has demonstrated significantly better reasoning and content generation abilities than GPT-3.5.<sup>5</sup>

<sup>3</sup>While all the other tasks and datasets were previously introduced by other studies, we present this task for the first time.

<sup>4</sup>We have developed suitable pipelines for answer extraction and processing tailored to each task. Specific implementation details can be found in our codebase.

<sup>5</sup>In our preliminary experiments, we also tested other OpenAI models such as text-davinci-003 and code-davinci-002, but we discovered that our meta-prompting approach yielded consequential results when applied to GPT-3.5 and GPT-4.



In all of our experiments, we consistently applied the same parameters and system instructions to the Meta Model. We set the temperature value at 0, the top-p value at 0.95, and the maximum token count at 1024.<sup>6</sup>

### Meta Model Instruction

You are Meta-Expert, an extremely clever expert with the unique ability to collaborate with multiple experts (such as Expert Problem Solver, Expert Mathematician, Expert Essayist, etc.) to tackle any task and solve any complex problems. Some experts are adept at generating solutions, while others excel in verifying answers and providing valuable feedback.

Note that you also have special access to Expert Python, which has the unique ability to generate and execute Python code given natural-language instructions. Expert Python is highly capable of crafting code to perform complex calculations when given clear and precise directions. You might therefore want to use it especially for computational tasks.

As Meta-Expert, your role is to oversee the communication between the experts, effectively using their skills to answer a given question while applying your own critical thinking and verification abilities.

To communicate with a expert, type its name (e.g., "Expert Linguist" or "Expert Puzzle Solver"), followed by a colon ":", and then provide a detailed instruction enclosed within triple quotes. For example:

Expert Mathematician:

```
"""
You are a mathematics expert, specializing in the fields of geometry and algebra.
Compute the Euclidean distance between the points (-2, 5) and (3, 7).
"""
```

Ensure that your instructions are clear and unambiguous, and include all necessary information within the triple quotes. You can also assign personas to the experts (e.g., "You are a physicist specialized in...").

Interact with only one expert at a time, and break complex problems into smaller, solvable tasks if needed. Each interaction is treated as an isolated event, so include all relevant details in every call.

If you or an expert finds a mistake in another expert's solution, ask a new expert to review the details, compare both solutions, and give feedback. You can request an expert to redo their calculations or work, using input from other experts. Keep in mind that all experts, except yourself, have no memory! Therefore, always provide complete information in your instructions when contacting them. Since experts can sometimes make errors, seek multiple opinions or independently verify the solution if uncertain. Before providing a final answer, always consult an expert for confirmation. Ideally, obtain or verify the final solution with two independent experts. However, aim to present your final answer within 15 rounds or fewer.

Refrain from repeating the very same questions to experts. Examine their responses carefully and seek clarification if required, keeping in mind they don't recall past interactions.

Present the final answer as follows:

```
>> FINAL ANSWER:
```

```
"""
[final answer]
"""
```

For multiple-choice questions, select only one option. Each question has a unique answer, so analyze the provided information carefully to determine the most accurate and appropriate response. Please present only one solution if you come across multiple options.

**Figure 3:** The instructions given to the Meta Model using the "system message" parameter in the GPT-4 API.

<sup>6</sup>The temperature value, which usually ranges between 0 and 1, controls how much randomness or creativity the model exhibits. Ideally, a temperature of 0 should lead to the model producing the same output when presented with the same input. However, both GPT-3.5 and GPT-4 have shown a tendency to generate varied responses even at this setting. This means that reproducing our exact results might be challenging under identical experimental conditions. To address this issue, we are releasing all model inputs, interactions, and outputs in our GitHub repository.

| Task                   | Basic       |             | Expert |         | SPP           | Meta        |             | $\Delta$<br>(M-S) |
|------------------------|-------------|-------------|--------|---------|---------------|-------------|-------------|-------------------|
|                        | Standard    | 0-CoT       | Static | Dynamic | Multi-Persona | - Python    | + Python    |                   |
| Checkmate-in-One       | <b>36.4</b> | 32.8        | 39.6   | 33.2    | 17.2          | <b>57.2</b> | <b>57.2</b> | +20.8             |
| Game of 24             | <b>3.0</b>  | 11.0        | 3.0    | 2.0     | 25.0          | 11.0        | <b>67.0</b> | +64.0             |
| Geometric Shapes       | <b>56.8</b> | <b>69.2</b> | 55.2   | 53.6    | 57.6          | 58.4        | 59.2        | +2.4              |
| MGSM (avg)             | <b>84.4</b> | 85.5        | 83.0   | 85.0    | <b>85.7</b>   | 85.4        | 84.8        | +0.4              |
| Multi-Step Arithmetic  | <b>84.0</b> | 83.2        | 83.2   | 78.8    | <b>91.6</b>   | 84.8        | 90.0        | +6.0              |
| Python Prog. Puzzles   | <b>31.1</b> | 36.3        | 33.8   | 25.0    | 32.5          | 32.7        | <b>45.8</b> | +14.7             |
| Sonnet Writing         | <b>62.0</b> | 71.2        | 74.0   | 74.0    | 73.2          | 77.6        | <b>79.6</b> | +17.6             |
| Word Sorting           | <b>80.4</b> | 83.6        | 83.2   | 85.2    | 79.2          | 84.0        | <b>99.6</b> | +19.2             |
| <b>Average (macro)</b> | <b>54.8</b> | 59.1        | 56.9   | 54.6    | 57.7          | 61.4        | 72.9        | +18.1             |

**Table 1:** Comparison of baselines with meta-prompting across tasks. Without a Python interpreter, meta-prompting significantly outperforms other methods on the Checkmate-in-One and Sonnet Writing tasks and is on par on most other tasks except Geometric Shapes. Meta-prompting can leverage the Python interpreter in a task-agnostic manner to improve performance significantly across many tasks.

## 4 Main Results and Discussion

The results of our experiments, summarized in Table 1, demonstrate the superior effectiveness of our meta-prompting approach compared to the standard zero-shot prompting methods. When we look at the overall performance across all tasks, there is a notable increase in accuracy with meta-prompting, especially when it is augmented with a Python interpreter. Specifically, meta-prompting outperforms standard prompting by 17.1%, expert (dynamic) prompting by 17.3%, and multipersona prompting by 15.2%. Below, we delve into four key insights that emerged from our empirical analysis.

### 4.1 Overall Performance

The meta-prompting approach, particularly when augmented with a Python interpreter, consistently outperforms conventional zero-shot prompting across various tasks. This approach proves to be especially effective in tackling tasks that are heavily reliant on heuristic or iterative trial-and-error problem-solving strategies. In the Game of 24 challenge, we see an accuracy improvement of over 60% compared to the basic standard prompting method (highlighted in pink), about a 15% gain in Python Programming Puzzles, and close to an 18% increase in accuracy for Sonnet Writing. These tasks require complex, iterative, and heuristic search strategies, where conventional single-shot prompting falls short. Conversely, meta-prompting leverages the collective intelligence of various expert personas to iteratively navigate towards a solution, thus fostering a more dynamic and effective problem-solving ecosystem.

Expanding upon its capabilities, meta-prompting appears to be effective in creative writing tasks as well. In the Shakespearean Sonnet Writing task, for instance, which demands linguistic precision and creative conformity to specific poetic structures, meta-prompting notably enhances performance. While standard prompting methods yield a 62% accuracy rate, meta-prompting achieves 79.6% and 77.6% accuracy, with and without a Python interpreter, respectively.

In MGSM and Geometric Shapes, the benefits of meta-prompting over the other prompting approaches seem minimal based on the first impression. Nonetheless, meta-prompting does provide 4-6% gains in Bengali and Telugu, two underrepresented languages with the lowest baseline performances. In Geometric Shapes,<sup>7</sup> we had expected that GPT-4 to identify the shapes of objects by generating and executing appropriate codes under meta-prompting, but this did not happen. Meta-prompting yielded only a modest 2.4% gain in this

<sup>7</sup>This task involves naming a shape from its SVG path. Note that the LMs we used did not offer visual capabilities at the time.



geometric task. We, however, acknowledge that the zero-shot-CoT baseline performed surprisingly better than all the other methods, outperforming meta-prompting with a 10% accuracy gap.

While the most significant gains were observed using a Python interpreter, we note that for the Checkmate-in-One task, meta-prompting achieved a 20.8% gain even without it. Overall, our results highlight the versatility of meta-prompting and underscore its potential for broad application beyond strictly computational problems.

## 4.2 Zero-Shot Decomposition, Error Detection, and Aggregation

The success of our meta-prompting framework lies partly in its strategic use of specialized knowledge, self-collaboration, and implicit verification loops. This approach, as well as multipersona prompting, encourages multi-turn interactions where different personas collaborate to resolve a problem.

To illustrate how the framework can be beneficial, consider solving multilingual arithmetic problems from the MGSM dataset. GPT-4, under the meta-prompting method, typically follows a three-phase approach: initially translating the problem from the source language (e.g., Bengali) to English, then applying computational expertise (like calling an Expert Mathematician) to find a solution, and finally, conducting an independent or corroborated verification. This unsupervised approach aligns with the multilingual CoT prompting method used by [Shi et al. \(2023\)](#) for MGSM, where the prompt instructs the LM to first translate the problem and then solve it. Meta-prompting performs such translation without explicitly being instructed to do so.

Our structured approach embodies the principle of the wisdom of the crowd ([Suzgun et al., 2023a](#)), which posits that a collective opinion of a diverse set of critical thinkers often surpasses the insights of individual experts. By harnessing an ensemble of specialized expert models under the guidance of the Meta Model, each contributing from different angles of expertise, we achieve more accurate and reliable problem-solving.

## 4.3 Fresh Eyes

The concept of *fresh eyes* helps mitigate the well-known problem of LMs doubling-down on their mistakes and exhibiting overconfidence (see, e.g., [Zhang et al., 2023b](#)). Fresh eyes are a crucial differentiator between meta-prompting and the multipersona prompting, and thus comparing experimental results demonstrates the advantage. In meta-prompting, fresh perspectives are introduced by engaging experts—or personas—to reassess the problem. This approach provides an opportunity for novel insights and the potential discovery of previously unnoticed incorrect solutions.

Grounded in principles from cognitive psychology, fresh perspectives can lead to more creative problem-solving and error detection. When individuals or models approach a problem without preconceived notions, they are more likely to consider alternative solutions and identify errors that might have been overlooked. Fresh eyes may help avoid cognitive biases such as anchoring, confirmation bias, as well as overconfidence.

Consider the following summary of an execution, which illustrates the benefit of “fresh eyes” in practice. Say the task is the 24 game, e.g., to use each of the numbers 6, 11, 12, and 13, exactly once, in an arithmetic expression whose value is 24. The history may look something like the following:

1. The Meta Model proposes consulting experts in mathematics, problem-solving, and Python programming. It emphasizes the need for accuracy and adherence to constraints, suggesting the involvement of another expert for review if needed.
2. An expert proposes a solution, which a second expert identifies to be incorrect, and the Meta Model suggests writing a Python program to find a valid solution.
3. A programming expert is consulted to write a program.
4. Another programming expert identifies an error in the script, modifies it, and then executes the revised script.

5. A mathematics expert is consulted to verify the solution output by the program.
6. After this verification, the Meta Model outputs it as the final answer.

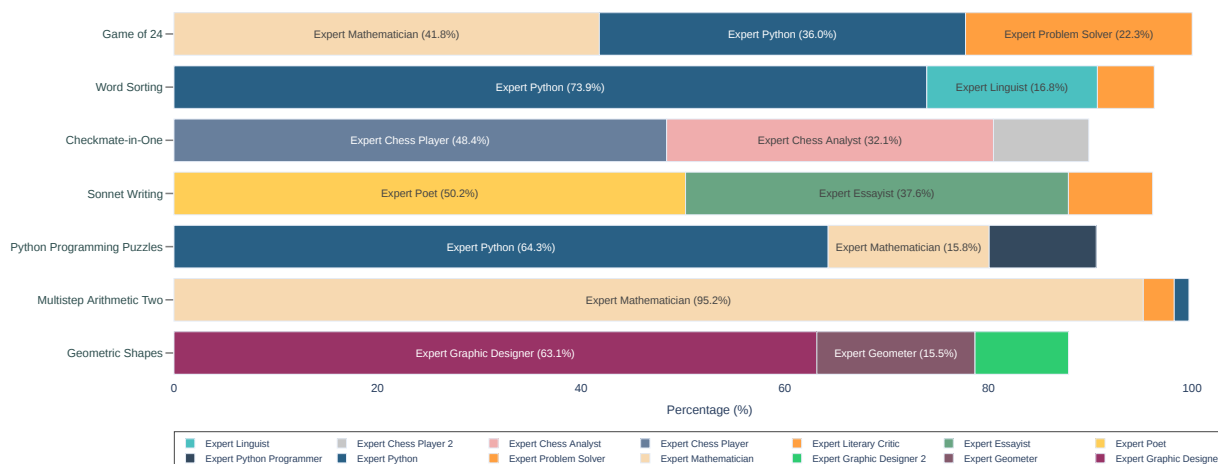
This example underscores how meta-prompting, incorporating fresh perspectives at each step (since the expert’s prompt does not include the whole history), not only finds solutions but also effectively identifies and corrects errors. The diversity of perspectives, ranging from problem-solving strategies to technical execution and verification, demonstrates how different angles of expertise contribute to a more robust and reliable problem-solving process.

## 4.4 Real-Time Code Execution

The introduction of a Python expert for code generation and execution within our meta-prompting framework leads to significant advancement in tackling algorithmic challenges. This enhancement is evident in Python Programming Puzzles, where the integration of the Expert Python into the meta-prompting framework elevates the success rate from 32.7% to 45.8%. This improvement primarily arises from the Meta Model’s ability to use a Python expert for generating and executing code based on natural-language instructions. Real-time code execution enables instant validation and optimization of solutions, substantially improving both the efficiency and precision of problem-solving.

This enhancement is not confined to a single task type, however. In tasks such as the Game of 24 and Word Sorting, accuracy rates increase by 56.0% and 15.6%, respectively, with the integration of a Python interpreter into meta-prompting. (When compared with the baseline standard prompting, the accuracy gains correspond to 64.0% and 19.2%, respectively.) These improvements highlight the significant role of code generation and execution in enhancing the effectiveness of the meta-prompting framework, demonstrating its transformative impact across various computational tasks. Overall, integrating a Python interpreter results in an average performance improvement of an additional 11.5% across different tasks compared to meta-prompting without a Python interpreter.

However, the introduction of real-time code execution also brings essential security considerations. Establishing such a system requires a secure and controlled environment to mitigate risks such as data breaches and system vulnerabilities. Therefore, the deployment of a Python interpreter within the meta-prompting framework should be fortified with a secure sandbox. These measures are crucial to ensure the system’s integrity and the protection of user data, ensuring that the advantages of improved problem-solving efficiency are not compromised in any way by security and privacy concerns, among other issues.



**Figure 4:** Distribution of experts conjured by the Meta Model in experiments *involving* a Python interpreter. The remaining blank space represents a combination of experts that were employed infrequently.

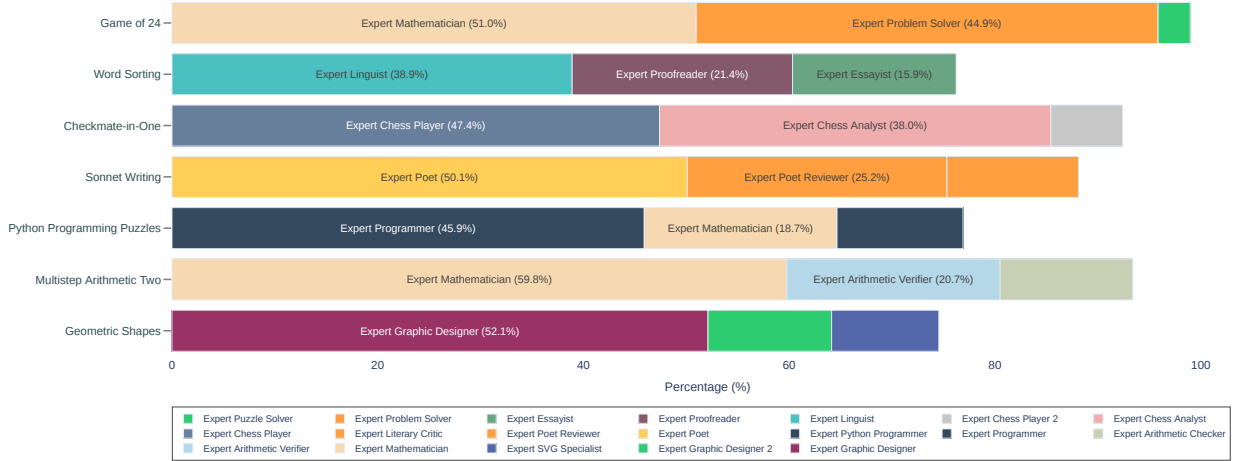


Figure 5: Distribution of experts conjured by the Meta Model in experiments *without* the use of a Python interpreter.

## 5 Further Discussion

### 5.1 Additional Analysis of Meta Prompting

**Analysis of Expert Types Used in Meta Prompting.** The Meta Model’s dynamic selection of expert types distinctly illustrates its adaptability and strategic alignment with specific task requirements. Analyzing tasks with and without a Python interpreter offers insightful contrasts in the model’s expert choices, influenced by the available tools and task characteristics. In scenarios where a Python expert is explicitly mentioned for code generation and execution, there is a noticeable preference for technical and computational expertise. For example, in Python Programming Puzzles, the Meta Model frequently utilizes Expert Python, Expert Mathematician, and several tiers of Expert Python Programmers. This pattern reveals a task-oriented strategy, highlighting a focus on programming and algorithmic problem-solving. Similarly, tasks such as Game of 24 and Word Sorting prominently feature Expert Python, reinforcing the model’s propensity to rely on computational expertise when Python capabilities are accessible.

In contrast, for meta-prompting without a specific Python expert, the spectrum of experts employed is more diverse. Tasks like Geometric Shapes predominantly involve design and geometry experts (e.g., Expert Graphic Designer and Expert Geometer), indicating a pivot towards visual and spatial problem-solving rather than computational approaches. This task illustrates where the Meta Model may have made a poor choice of experts, and in particular it might have been more preferable to use an expert in SVG visualizations. In Sonnet Writing, the Meta Model naturally leans on literary experts, notably Expert Poet and Expert Literary Critic, emphasizing creative and linguistic skills. This pattern demonstrates the Meta Model’s ability to dynamically tailor its expert engagement to the demands of the task, utilizing technical experts for computational challenges and a varied range of non-computational expertise for creative or abstract tasks.

**Number of Rounds Taken to Reach a Solution.** Examining the meta-prompting experiments involving a Python expert reveals that the average number of rounds required to reach a solution in the Meta Model varies significantly across tasks, indicative of their complexity and specific nature. Simpler tasks, such as Word Sorting (3.31 rounds) and Checkmate-in-One (3.48 rounds), typically necessitate fewer rounds, suggesting a more linear and straightforward resolution process, likely due to their clearly defined parameters. Conversely, more algorithmically challenging tasks like Python Programming Puzzles average a higher number of rounds at 6.07, reflecting the nuanced and multifaceted aspects of programming tasks that require extensive interactions for thorough clarification and iterative refinement. The Game of 24 and Multistep Arithmetic Two, with averages around 3.5 rounds, meld computational proficiency with logical reasoning, necessitating additional rounds for accurate and precise solutions. This observed correlation between the number of rounds and the task complexity underscores the Meta Model’s proficiency and adaptability. It efficiently manages simpler tasks with minimal interactions while skillfully handling the complexities of more challenging and

heuristic-based problems, ensuring precision and efficacy in its solutions. This performance characteristic is particularly critical in environments where efficiency and interaction trade-off are key.

**Enhancing Solution Reliability through Systematic Verification.** The Meta Model's systematic verification protocol strengthens the reliability and robustness of its solutions. Fundamental to this approach is the consistent practice of consulting an expert for validation before finalizing responses, a principle applied across diverse tasks. This method is further evidenced by the detailed interaction data. In tasks such as Checkmate in One, for instance, the Meta Model employs a two-step verification strategy. Initially, it consults an Expert Chess Player to come up with a solution, followed by a critical verification from an Expert Chess Analyst, ensuring strategic correctness. A similar approach is adopted in Sonnet Writing too, where an Expert Poet drafts the sonnet, and an Expert Poet Reviewer or Expert Essayist reviews it, making sure that the solution adheres to the strict rhyme scheme. This unsupervised but rigorous verification process extends to complex tasks like Game of 24 and MGSM, involving both external expert consultations and internal reviews. By integrating this dual verification mechanism, the model significantly enhances solution accuracy and reliability, essential for real-world applications where precision is paramount.

**Navigating No-Solution Territories.** Meta-prompting enables the Meta Model to acknowledge the absence or impossibility of a valid solution or its inability to find one more frequently than other prompting methods. In 100 examples of the Game of 24, the model reports no solution 9 times with Expert Python and 15 times without it, compared to the mere 2 instances under standard prompting. In Checkmate, across 250 examples, it admits to no solution 12 times without Expert Python and 10 times with it, a rarity in multipersona and standard prompting. While there were always solutions, it is arguably preferable to abstain from answering rather than provide an incorrect answer. Typically expressed as "No valid solution found" or more explicitly as "There is no solution to the 24 game with these numbers given the constraints," these acknowledgments are likely the result of the model's verification and feedback loop, emphasizing accuracy and confidence over speculative but incorrect responses.

**Setting the Bar High: GPT-4's Zero-Shot Task Solving Capabilities.** Even without the enhanced capabilities of meta-prompting, GPT-4 stands out as an effective zero-shot task solver under standard prompting conditions. Its performance across various tasks, including Python Programming Puzzles and MGSM, is remarkable, particularly when compared to other LMs as highlighted by [OpenAI \(2023\)](#). GPT-4 excels as a task-agnostic solver, capable of processing and responding to diverse queries effectively. A significant attribute of GPT-4 is its proficiency in following instructions. Given clear and unambiguous natural-language instructions, the model demonstrates a high level of compliance and accuracy. This aspect of instruction-following is also a cornerstone of our meta-prompting framework, where we leverage GPT-4's capabilities. Our experiments reinforce that GPT-4 excels in code generation, demonstrates impressive zero-shot reasoning, and engages effectively in role-playing, solidifying its position as a versatile and reliable LM.

**Limited Performance Improvement with GPT-3.5.** In comparison to GPT-4, GPT-3.5 demonstrates a more limited scope of performance enhancement across various tasks. Although it shows notable improvements in specific tasks such as Sonnet Writing and Checkmate-in-One, its capabilities do not consistently surpass baseline standards or zero-shot CoT prompting methods in other tasks, notably Word Sorting and Multiple Arithmetic Two. Our qualitative analysis suggests that GPT-3.5 may not be as effective as GPT-4 in simulating role-playing scenarios or managing extended context windows. This observation leads us to believe that factors such as the scale of the model, the quality and size of the instruction-following corpus may be significantly influencing the efficacy of the meta-prompting approach. Furthermore, it appears that the advantages offered by meta-prompting may even emerge more prominently at larger model scales.

## 5.2 Limitations and Failure Modes of Meta Prompting

The meta-prompting framework, despite its innovative approach, encounters several notable limitations, including cost efficiency, scalability, operational linearity, domain restrictions, information transfer challenges, and response patterns. A primary limitation is the elevated cost associated with multiple model calls. In our setup using GPT-4, the dual role of the Meta Model and the experts, distinguished by unique instructions, incurs substantial costs under the GPT-4 API pricing model. This cost factor diminishes the effectiveness

of meta-prompting in smaller models like ChatGPT, which lack the comprehensive capabilities of GPT-4. Consequently, meta-prompting, though insightful, can become prohibitively expensive due to extensive model interactions and lengthy message histories. However, these costs will decrease as the costs of LMs decrease. Note that recent OpenAI API features announced after the experiments were run, namely the ability to run code in a sandbox directly through the API, could significantly decrease the costs of our system.

Another critical limitation is the requirement for substantial scale and a considerable context window. GPT-4 fits this criterion, but smaller models such as ChatGPT fall short. Meta-prompting’s design, characterized by extensive message histories, demands an LM capable of handling and retaining lengthy textual information, a feature not universally present in all LMs. Operational efficiency is also challenged by the linear (sequential) nature of meta-prompting. The framework, in its current form, processes steps one at a time, relying on the outcome of preceding calls. This dependency constrains the possibility of parallel processing, impacting the speed and efficiency of the system.

Additionally, our research confined meta-prompting within a closed-domain system. Nevertheless, the framework’s potential extends to incorporating external resources such as APIs, specialized finetuned models, search engines, or computational tools. More expansive implementations like AutoAgents (Chen et al., 2023a) and AutoGen (Wu et al., 2023), which include higher-level planning and diverse cooperation mechanisms, offer a glimpse into future directions. In subsequent versions, the Meta Model could benefit from refining or summarizing its history before advancing, optimizing the relevance and efficiency of the process. There is also untapped potential in concurrently summoning multiple experts or utilizing a single expert with varied temperature parameters to synthesize their outputs.

A practical challenge faced is the Meta Model’s occasional oversight in conveying necessary information to experts, forgetting that experts can only access data adhering to a certain format (within triple quotes in our system). This oversight can lead to unintended confusion and underscores the need for improved information management. Lastly, the Meta Model’s response pattern, particularly in tasks with lower performance, often includes apologies, such as “Apologies for the confusion in my previous response” or “I apologize for the previous incorrect solution.” This behavior likely stems from its training on instruction-following data.

## 6 Related Work

This section seeks to contextualize our proposed meta-prompting approach amidst recent advancements in prompting strategies and scaffolding techniques. We provide a brief overview of these developments, highlighting their relevance and connections to our work.

**Enhancing Reasoning in Language Models through Prompting.** Recent efforts in LM scaffolding and prompting methods have significantly boosted the arithmetic and commonsense reasoning capabilities of LMs. The chain-of-thought (CoT) prompting (Wei et al., 2022b) and its variants—including least-to-most (Zhou et al., 2023), zero-shot CoT (Kojima et al., 2022), self-ask (Press et al., 2022), ask-me-anything (Arora et al., 2023), decomposed prompting (Khot et al., 2023), and auto-CoT (Zhang et al., 2023d)—have marked a paradigm shift in how LMs process complex queries. These methods encourage LMs to adopt human-like, sequential thinking processes, breaking down intricate questions into simpler subtasks and systematically solving them before presenting a final answer. Multiple studies (Wei et al., 2022a; Madaan and Yazdanbakhsh, 2022; Shi et al., 2023; Drozdov et al., 2023; Fu et al., 2023b; Suzgun et al., 2023b, *inter alia*) have shown the efficacy of these prompting methods across a broad set of tasks and benchmarks. More recent innovations such Tree-of-Thought (Yao et al., 2023a), Graph-of-Thought (Besta et al., 2023), Program-of-Thought (Chen et al., 2023d), and Skeleton-of-Thought (Ning et al., 2023), have further enriched this domain; these explore dynamic, non-linear reasoning pathways, broadening the computational and heuristic capabilities of LMs. However, they come with increased resource demands and greater time complexity, require multiple manual prompt crafting, and are often specialized for particular types of tasks.

**Iterative Self-Feedback and Refinement Mechanisms.** Recent instruction-following techniques and data-collection efforts have expanded the capabilities of LMs to follow instructions, emulate certain aspects of human behavior, and assist in tasks such as annotation and evaluation (Haluptzok et al., 2022; Aher et al.,



2023). LMs such as GPT-4, PaLM, and Llama are now capable of effectively integrating self-feedback and refinement mechanisms through prompting and can leverage their own natural-language outputs to guide their behaviour and improve decision-making. SayCan (Ahn et al., 2022) and Inner Monologue (Huang et al., 2023) are early examples showcasing the benefits of inner dialogues in a closed-loop system for robotic control and action planning. Reflexion (Shinn et al., 2023) builds upon these studies and focuses on natural-language generation and reasoning tasks. It functions as a policy optimization mechanism through natural language feedback, using self-feedback and self-reflection to influence and correct behaviors in LMs, and has shown considerable success in preliminary experiments. In a more innovative vein, the Self-Taught Reasoner approach (STaR; Zelikman et al., 2022) iteratively trains an LM on its own outputs to refine initial rationales for more accurate solutions, leading to enhanced reasoning skills. Other notable methods such as Critic (Gou et al., 2023), Iterative Refinement (Chen et al., 2023b), RCI (Kim et al., 2023), Re<sup>3</sup> (Yang et al., 2022), Refiner (Paul et al., 2023), Self-Critique (Saunders et al., 2022), Self-Correction (Welleck et al., 2023), Self-Eval, Self-Debug (Chen et al., 2023c), Self-Edit (Zhang et al., 2023a), Self-Evolve (Jiang et al., 2023b), Self-Taught Optimizer (SToP; Zelikman et al., 2023), and so forth, illustrate how verbal feedback, both internal and external, can significantly improve the accuracy, quality, and robustness of model outputs across various tasks and setups.

**Exploring Role-Playing in Language Models.** The integration of role-playing and self-collaboration concepts into LMs, grounded in cognitive psychology and developmental education principles, has emerged as a useful method for augmenting LMs’ problem-solving capabilities and optimizing their internal domain-specific knowledge and expertise. Recent studies (Park et al., 2022, 2023; Li et al., 2023; Xu et al., 2023; Fu et al., 2023a; Deshpande et al., 2023) have shown that endowing instruction-following LMs with “expert” personas or roles enhances the quality and accuracy of their output. In particular, approaches like CAMEL (Li et al., 2023) and Expert Prompting (Xu et al., 2023), which involve dynamically assigning personas to a single LM, have been shown to yield higher quality and more reliable responses than models without designated personas. Further investigations (Chen et al., 2023a,e; Du et al., 2023; Hao et al., 2023b; Liang et al., 2023; Liu et al., 2023b; Jiang et al., 2023a; Xiong et al., 2023; Zhang et al., 2023c) demonstrate that assigning multiple expert identities or roles to a single LM, tailored to specific tasks or problems, and prompting it to conduct multi-round internal dialogues—similar to a team of experts discussing and refining ideas—amplifies the reliability and comprehensiveness of the LM’s analysis; this leads to more well-rounded and thorough solutions. These studies advocate a complementary approach wherein multiple instances of an LM propose, debate, and refine their individual responses and reasoning in successive rounds, culminating in a unified final answer. This role-playing concept has shown to significantly improve mathematical and strategic reasoning across various tasks. Moreover, it improves the factual accuracy of the generated content, thereby reducing erroneous or fabricated responses.

**Autonomous Decision-Making and Execution in Multi-Agent LM Systems.** There has been a growing interest in using LMs for autonomous decision-making and task execution. Open-source projects like Auto-GPT, Agent-GPT, Baby-AGI, and LangChain are notable efforts developing agent protocols that are capable of planning, decision-making, and executing tasks end-to-end, with minimal or no human intervention. These systems highlight the potential and risks of LMs, which go beyond performing predefined tasks to adapting, learning, and autonomously executing decisions in real time. As discussed by Masa (2023), those autonomous models might be exploited by individuals with malicious intents and pose threats to humanity. There is also the dilemma of accountability: who bears responsibility when an LM-driven autonomous agent produces an inappropriate or criminal action? Ensuring safety and security with these agents is crucial, given its potential for mishaps or exploitation by malicious actors, and its vulnerability to cyber-attacks.

**Integration of External Tools and APIs into Language Models.** As LMs continue to evolve, the integration of external tools is becoming increasingly important. This tool-use integration, often achieved through in-context learning (e.g., Cai et al., 2023) or finetuning (e.g., Schick et al., 2023a), allows LMs to effectively engage with real-world scenarios and tackle a diverse range of dynamic tasks. Recent advancements (Cai et al., 2023; Gao et al., 2023; Gou et al., 2023; Hao et al., 2023c; Khattab et al., 2023; Lu et al., 2023; Qiao et al., 2023; Paranjape et al., 2023; Patil et al., 2023; Schick et al., 2023a; Yang et al., 2023; Yuan et al., 2023) have enabled LMs to perform accurate calculations, retrieve up-to-date information from search engines or databases, and interact with APIs, making them crucial for complex, multimodal real-world problems.



OpenAI’s incorporation of predefined APIs and plugins into ChatGPT underscores the importance of external integration in developing a comprehensive LM ecosystem. However, most approaches often limit themselves to a select group of tools or domain-specific resources, posing challenges in adapting to new domains (Lu et al., 2023). Our meta-prompting approach, as detailed in Section 2, treats the LM as an independent tool and expert, available on-demand for specific tasks. Furthermore, incorporating a Python interpreter—through Expert Python—to execute and evaluate model-generated code has been instrumental in enhancing both accuracy and efficiency in various tasks.

## 7 Conclusion

In this work, we have introduced and examined meta-prompting, a simple yet powerful scaffolding technique that enhances the performance of language models in a task-agnostic manner. This approach leverages a language model to act as both a central conductor and a group of expert instances, thereby endowing traditional models with dynamic, multi-functional capabilities. A noteworthy aspect of meta-prompting lies in its proficiency to decompose complex tasks, engage distinct expertise for each component, and then integrate the varied outputs seamlessly. Demonstrating significant, double-digit improvements across a series of tasks, ranging from challenging arithmetic puzzles like the Game of 24 to the creative literary exercise of Shakespearean Sonnet Writing, meta-prompting promises to grow more potent and cost-efficient as language models continue to evolve, offering exciting prospects for future applications.

## Acknowledgements

We would like to thank Federico Bianchi, Annabelle Carrell, Tayfun Gür, Dan Jurafsky, Suproteem Sarkar, Scott Duke Kominers, Lester Mackey, Neil Mallinar, Şule Kahraman, Deniz Keleş, Luke Melas-Kyriazi, Drew Pendergrass, Faiz Surani, Garrett Tanzer, Michael Wornow, and Eric Zelikman for their valuable comments, useful suggestions, and support.

## References

- Gati V Aher, Rosa I. Arriaga, and Adam Tauman Kalai. 2023. Using Large Language Models to Simulate Multiple Humans and Replicate Human Subject Studies. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 337–371. <https://proceedings.mlr.press/v202/aher23a.html>
- Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691* (2022).
- Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. PaLM 2 Technical Report. *arXiv preprint arXiv:2305.10403* (2023). <https://arxiv.org/abs/2305.10403>
- Simran Arora, Avanika Narayan, Mayee F Chen, Laurel Orr, Neel Guha, Kush Bhatia, Ines Chami, and Christopher Re. 2023. Ask Me Anything: A simple strategy for prompting language models. In *The Eleventh International Conference on Learning Representations*. <https://openreview.net/forum?id=bhUPJnS2g0X>
- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, et al. 2023. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687* (2023).
- BIG-Bench authors. 2023. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research* (2023). <https://openreview.net/forum?id=uyTL5Bvosj>
- Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. 2023. Large language models as tool makers. *arXiv preprint arXiv:2305.17126* (2023).
- Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F Karlsson, Jie Fu, and Yemin Shi. 2023a. AutoAgents: A Framework for Automatic Agent Generation. *arXiv preprint arXiv:2309.17288* (2023).
- Pinzhen Chen, Zhicheng Guo, Barry Haddow, and Kenneth Heafield. 2023b. Iterative Translation Refinement with Large Language Models. *arXiv preprint arXiv:2306.03856* (2023).
- Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023d. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks. *Transactions on Machine Learning Research* (2023). <https://openreview.net/forum?id=YfZ4ZPt8zd>
- Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, et al. 2023e. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *arXiv preprint arXiv:2308.10848* (2023).
- Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023c. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128* (2023).
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168* (2021).
- Ameet Deshpande, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, and Karthik Narasimhan. 2023. Toxicity in chatgpt: Analyzing persona-assigned language models. *arXiv preprint arXiv:2304.05335* (2023).
- Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. 2023. Compositional Semantic Parsing with Large Language Models. In *The Eleventh International Conference on Learning Representations*. <https://openreview.net/forum?id=gJW8hSGBys8>

- Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving Factuality and Reasoning in Language Models through Multiagent Debate. *arXiv preprint arXiv:2305.14325* (2023).
- Yao Fu, Hao Peng, Tushar Khot, and Mirella Lapata. 2023a. Improving language model negotiation with self-play and in-context learning from ai feedback. *arXiv preprint arXiv:2305.10142* (2023).
- Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023b. Complexity-Based Prompting for Multi-step Reasoning. In *The Eleventh International Conference on Learning Representations*. <https://openreview.net/forum?id=yf1icZHC-19>
- Difei Gao, Lei Ji, Luowei Zhou, Kevin Qinghong Lin, Joya Chen, Zihan Fan, and Mike Zheng Shou. 2023. AssistGPT: A General Multi-modal Assistant that can Plan, Execute, Inspect, and Learn. *arXiv preprint arXiv:2306.08640* (2023).
- Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2023. CRITIC: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738* (2023).
- Patrick Haluptzok, Matthew Bowers, and Adam Tauman Kalai. 2022. Language Models Can Teach Themselves to Program Better. In *The Eleventh International Conference on Learning Representations*.
- Rui Hao, Linmei Hu, Weijian Qi, Qingliu Wu, Yirui Zhang, and Liqiang Nie. 2023b. ChatLLM Network: More brains, More intelligence. *arXiv preprint arXiv:2304.12998* (2023).
- Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023a. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992* (2023).
- Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023c. ToolkenGPT: Augmenting Frozen Language Models with Massive Tools via Tool Embeddings. *arXiv preprint arXiv:2305.11554* (2023).
- Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. 2023. Inner Monologue: Embodied Reasoning through Planning with Language Models. In *Conference on Robot Learning*. PMLR, 1769–1782.
- Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023a. LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion. *arXiv preprint arXiv:2306.02561* (2023).
- Shuyang Jiang, Yuhao Wang, and Yu Wang. 2023b. SelfEvolve: A Code Evolution Framework via Large Language Models. *arXiv preprint arXiv:2306.02907* (2023).
- Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. 2023. DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines. *arXiv preprint arXiv:2310.03714* (2023).
- Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed Prompting: A Modular Approach for Solving Complex Tasks. In *The Eleventh International Conference on Learning Representations*. [https://openreview.net/forum?id=\\_nGgzQjzaRy](https://openreview.net/forum?id=_nGgzQjzaRy)
- Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. *arXiv preprint arXiv:2303.17491* (2023).
- Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models are Zero-Shot Reasoners. In *Advances in Neural Information Processing Systems*, Vol. 35. 22199–22213.
- Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large scale language model society. *arXiv preprint arXiv:2303.17760* (2023).

- Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. 2023. Encouraging Divergent Thinking in Large Language Models through Multi-Agent Debate. *arXiv preprint arXiv:2305.19118* (2023).
- Zeyi Liu, Arpit Bahety, and Shuran Song. 2023a. Reflect: Summarizing robot experiences for failure explanation and correction. *arXiv preprint arXiv:2306.15724* (2023).
- Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2023b. Dynamic LLM-Agent Network: An LLM-agent Collaboration Framework with Agent Team Optimization. *arXiv preprint arXiv:2310.02170* (2023).
- Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-Play Compositional Reasoning with Large Language Models. *arXiv preprint arXiv:2304.09842* (2023).
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651* (2023).
- Aman Madaan and Amir Yazdanbakhsh. 2022. Text and patterns: For effective chain of thought, it takes two to tango. *arXiv preprint arXiv:2209.07686* (2022).
- Masa. 2023. 7 Challenges and Potential Risks of AutoGPT Technology. *AutoGPT Official* (Apr 2023). <https://autogpt.net/challenges-and-potential-risks-of-autogpt-technology/>
- Xuefei Ning, Zinan Lin, Zixuan Zhou, Huazhong Yang, and Yu Wang. 2023. Skeleton-of-thought: Large language models can do parallel decoding. *arXiv preprint arXiv:2307.15337* (2023).
- OpenAI. 2023. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774* (2023). <https://arxiv.org/abs/2303.08774>
- Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. 2023. ART: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014* (2023).
- Joon Sung Park, Joseph C. O’Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. In *In the 36th Annual ACM Symposium on User Interface Software and Technology (UIST ’23)* (San Francisco, CA, USA) (UIST ’23). Association for Computing Machinery, New York, NY, USA.
- Joon Sung Park, Lindsay Popowski, Carrie Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2022. Social simulacra: Creating populated prototypes for social computing systems. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–18.
- Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334* (2023).
- Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2023. Refiner: Reasoning feedback on intermediate representations. *arXiv preprint arXiv:2304.01904* (2023).
- Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350* (2022).
- Shuofei Qiao, Honghao Gui, Huajun Chen, and Ningyu Zhang. 2023. Making Language Models Better Tool Learners with Execution Feedback. *arXiv preprint arXiv:2305.13068* (2023).
- William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. 2022. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802* (2022).

- Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023a. Toolformer: Language Models Can Teach Themselves to Use Tools.
- Timo Schick, Jane A. Yu, Zhengbao Jiang, Fabio Petroni, Patrick Lewis, Gautier Izacard, Qingfei You, Christoforos Nalmpantis, Edouard Grave, and Sebastian Riedel. 2023b. PEER: A Collaborative Language Model. In *The Eleventh International Conference on Learning Representations*. <https://openreview.net/forum?id=KbYevcLjnc>
- Tal Schuster, Ashwin Kalyan, Alex Polozov, and Adam Tauman Kalai. 2021. Programming Puzzles. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. <https://arxiv.org/abs/2106.05784>
- Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2023. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations*. <https://openreview.net/forum?id=fR3wGCK-IXp>
- Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366* (2023).
- Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. 2023. AdaPlanner: Adaptive Planning from Feedback with Language Models. *arXiv preprint arXiv:2305.16653* (2023).
- Mirac Suzgun, Luke Melas-Kyriazi, and Dan Jurafsky. 2023a. Follow the Wisdom of the Crowd: Effective Text Generation via Minimum Bayes Risk Decoding. In *Findings of the Association for Computational Linguistics: ACL 2023*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 4265–4293. <https://doi.org/10.18653/v1/2023.findings-acl.262>
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023b. Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. In *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, Toronto, Canada, 13003–13051. <https://doi.org/10.18653/v1/2023.findings-acl.824>
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).
- Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. 2023. Unleashing Cognitive Synergy in Large Language Models: A Task-Solving Agent through Multi-Persona Self-Collaboration. *arXiv preprint arXiv:2307.05300* (2023).
- Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research* (2022). <https://openreview.net/forum?id=yzkSU5zdwD> Survey Certification.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022b. Chain of Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). [https://openreview.net/forum?id=\\_VjQ1MeSB\\_J](https://openreview.net/forum?id=_VjQ1MeSB_J)
- Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khoshdel, and Yejin Choi. 2023. Generating Sequences by Learning to Self-Correct. In *The Eleventh International Conference on Learning Representations*. <https://openreview.net/forum?id=hH36JeQZDa0>



- Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework. *arXiv:2308.08155* [cs.AI]
- Kai Xiong, Xiao Ding, Yixin Cao, Ting Liu, and Bing Qin. 2023. Diving into the Inter-Consistency of Large Language Models: An Insightful Analysis through Debate. *arXiv preprint arXiv:2305.11595* (2023).
- Benfeng Xu, An Yang, Junyang Lin, Quan Wang, Chang Zhou, Yongdong Zhang, and Zhendong Mao. 2023. ExpertPrompting: Instructing Large Language Models to be Distinguished Experts. *arXiv preprint arXiv:2305.14688* (2023).
- Kevin Yang, Yuandong Tian, Nanyun Peng, and Dan Klein. 2022. Re3: Generating Longer Stories With Recursive Reprompting and Revision. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 4393–4479. <https://doi.org/10.18653/v1/2022.emnlp-main.296>
- Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. 2023. Gpt4tools: Teaching large language model to use tools via self-instruction. *arXiv preprint arXiv:2305.18752* (2023).
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601* (2023).
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*.
- Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi R Fung, Hao Peng, and Heng Ji. 2023. Craft: Customizing llms by creating and retrieving from specialized toolsets. *arXiv preprint arXiv:2309.17428* (2023).
- Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. 2023. Self-Taught Optimizer (STOP): Recursively Self-Improving Code Generation. *arXiv preprint arXiv:2310.02304* (2023).
- Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. STaR: Bootstrapping Reasoning With Reasoning. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). [https://openreview.net/forum?id=\\_3ELRdg2sgI](https://openreview.net/forum?id=_3ELRdg2sgI)
- Kechi Zhang, Zhuo Li, Jia Li, Ge Li, and Zhi Jin. 2023a. Self-Edit: Fault-Aware Code Editor for Code Generation. *arXiv preprint arXiv:2305.04087* (2023).
- Muru Zhang, Ofir Press, Will Merrill, Alisa Liu, and Noah A. Smith. 2023b. How Language Model Hallucinations Can Snowball. *ArXiv abs/2305.13534* (2023). <https://api.semanticscholar.org/CorpusID:258841857>
- Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew Chi-Chih Yao. 2023c. Cumulative reasoning with large language models. *arXiv preprint arXiv:2308.04371* (2023).
- Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023d. Automatic Chain of Thought Prompting in Large Language Models. In *The Eleventh International Conference on Learning Representations*. <https://openreview.net/forum?id=5NTt8GFjUHkr>
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. In *The Eleventh International Conference on Learning Representations*. <https://openreview.net/forum?id=WZH7099tgfM>
- Mingchen Zhuge, Haozhe Liu, Francesco Faccio, Dylan R Ashley, Róbert Csordás, Anand Gopalakrishnan, Abdullah Hamdi, Hasan Abed Al Kader Hammoud, Vincent Herrmann, Kazuki Irie, et al. 2023. Mindstorms in Natural Language-Based Societies of Mind. *arXiv preprint arXiv:2305.17066* (2023).